Die vom Verfasser erstellten Avenueprogramme in der Anlage zur Dissertation „Computergestützte Auswertung, Modellierung und Visualisierung der quartären Mittelterrassen und Niederterrassen in der südlichen Niederrheinischen Bucht durch Programmierung von ArcView“

```
'achshrst.ave
'Zur Herstellung der Achsen-, der Skalenlinien und der Koordinaten
'für einen Profilschnitt mit einer Überhöhung als zwei FThemen.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("achshrst")
myScript.SetNumberFormat( "d.dddd") ' script default

XAchsMinStr="2558600.00"
XAchsMaxStr="2582450.00"
XKKStr=MsgBox.Input("die kleinste Koord. von X-Achse",
     "Eingabe der Koord. der Achse", XAchsMinStr)
XGKStr=MsgBox.Input("die größte Koord. von X-Achse",
     "Eingabe der Koord. der Achse", XAchsMaxStr)
XKK=XKKStr.AsNumber
XGK=XGKStr.AsNumber
YAchsnHStr="0.00"
YminHStr=MsgBox.Input("die kleinste Koord. von Y-Achse [m]",
       "Eingabe der Koord. der Achse", YAchsnHStr)
YminH=YminHStr.AsNumber
YAchsmHStr="210.00"
YmaxHStr=MsgBox.Input("die größte Koord. von Y-Achse [m]",
       "Eingabe der Koord. der Achse", YAchsmHStr)
YmaxH=YmaxHStr.AsNumber
YFaktStr="50"
YFtStr=MsgBox.Input("zur Überhöhung von Y_Achse",
     "Eingabe eines Faktors", YFaktStr)
YFt=YFtStr.AsNumber
YminF=YminH*YFt
YmaxF=YmaxH*YFt
ListofAchs={}   'Liste für Achsen- und Skalenlinien
ListofPL1={}    'Liste für Linien
ListofPL1.Add(XKK@YminF) ' x-Achse
ListofPL1.Add(XGK@YminF)
ListofAchs.Add(ListofPL1)
ListofPL1={}
ListofPL1.Add(XKK@YminF) ' linke y-Achse
ListofPL1.Add(XKK@YmaxF)
ListofAchs.Add(ListofPL1)
ListofPL1={}
ListofPL1.Add(XGK@YminF) ' rechte y-Achse
ListofPL1.Add(XGK@YmaxF)
ListofAchs.Add(ListofPL1)

aSkLD=(((XGK-XKK)/23850)*10).AsString
aSkLStr=MsgBox.Input("Länge der langen Skalenlinien [m]",
      "Eingabe der Skalenlinien", aSkLD)
aSkL=aSkLStr.AsNumber

aSkFkt = aSkL * 0.7
SkLgr = aSkFkt
```

```
SkLkl = (aSkFkt/10)*5
SkLk2 = (aSkFkt/10)*4
SkLk3 = (aSkFkt/10)*2
SkLk4 = (aSkFkt/10)*1
SkLk5 = (aSkFkt/10)*0.5

ListofxSkd = {5000.00, 1000.00, 500.00, 100.00, 50.00, 10.00}
ListofxSkLd = {SkLgr,SkLkl,SkLk2,SkLk3,SkLk4,SkLk5}
AnzXSkLStr = MsgBox.Input("Anzahl der Sorte der Skalenlinien auf x-Achse",
        "Eingabe der Anzahl der Skalenlinien","6")
AnzXSkL = AnzXSkLStr.AsNumber
IdxXSkL = AnzXSkL - 1

ListofxSk = {}
ListofxSkL = {}
maxxSk = 0
for each i in 0..IdxXSkL
  NrStr = (i + 1).SetFormat("d").AsString
  aSk = MsgBox.ListAsString(ListofxSkd,
      "für"++NrStr+"."++"Sorte auf x-Achse [m]"+NL+
      "(erst größte, dann immer kleinere)",
      "Eingabe der Skalenlinien (Abstand)")
  ListofxSk.Add(aSk)
  aSkIdx = ListofxSkd.FindByValue(aSk)
  aSkL = ListofxSkLd.Get(aSkIdx)
  ListofxSkL.Add(aSkL)
  if (aSk > maxxSk) then
    maxxSk = aSk
  end
end

axSkL = aSkL * 2.5
ySkLgr = axSkL
ySkLkl = (axSkL/10)*7
ySkLk2 = (axSkL/10)*4
ySkLk3 = (axSkL/10)*2
ySkLk4 = (axSkL/10)*1

ListofySkd = {100.00, 50.00, 10.00, 5.00, 1.00}
ListofySkLd = {ySkLgr,ySkLkl,ySkLk2,ySkLk3,ySkLk4}
AnzYSkLStr = MsgBox.Input("Anzahl der Sorte der Skalenlinien auf y-Achse [m]",
        "Eingabe der Anzahl der Skalenlinien","5")
AnzYSkL = AnzYSkLStr.AsNumber
IdxYSkL = AnzYSkL - 1

ListofySk = {}
ListofySkL = {}
maxySk = 0
for each i in 0..IdxYSkL
  NrStr = (i + 1).SetFormat("d").AsString
  aSk = MsgBox.ListAsString(ListofySkd,
      "für"++NrStr+"."++"Sorte auf y-Achse [m]"+NL+
      "(erst größte, dann immer kleinere)",
      "Eingabe der Skalenlinien (Abstand)")
  ListofySk.Add(aSk)
  aSkIdx = ListofySkd.FindByValue(aSk)
  aSkL = ListofySkLd.Get(aSkIdx)
  ListofySkL.Add(aSkL)
  if (aSk > maxySk) then
    maxySk = aSk
  end
end
```

```
max2ySk = 0
for each i in 0..IdxYSkL
  aSk = ListofySk.Get(i)
  if (aSk <> maxySk) then
    if (aSk > max2ySk) then
      max2ySk = aSk
    end
  end
end

'Herstellung der x-Skalenlinien
SFkt = (XGK - XKK)/23850  'Faktor für Position der Schrift
ListofPts={}
ListofSchrift={}
ytextpt1=YminF-(aSkL*YFt*10)  'y-Koord. für Schrift der x-Skalenlinie
ListofxSkKrd = {}

for each i in 0..IdxXSkL
   xSk = ListofxSk.Get(i)
   xM = XKK Mod xSk
   if (xM = 0) then
     xKrd = XKK
   elseif (xM <> 0) then
     xKrd = ((XKK / xSk).Ceiling) * xSk
   end
   xSkL = ListofxSkL.Get(i)
   yKrd = YminF-(xSkL*YFt)
   if (xSk = maxxSk) then
    if (xKrd <> XKK) then
      ListofPL1={}
      ListofPL1.Add(XKK@YminF)
      ListofPL1.Add(XKK@yKrd)
      ListofAchs.Add(ListofPL1)
      ListofPts.Add((XKK-(1000*SFkt))@ytextpt1)
      xNrStr=(XKK.SetFormat("").SetFormat("d")).AsString
      ListofSchrift.Add(xNrStr)
      ListofxSkKrd.Add(XKK)
    end
   end
   XGKP = XGK + 0.01
   while (xKrd <  XGKP)
     aTest = ListofxSkKrd.FindByValue(xKrd)
     if (aTest = -1) then
       ListofPL1={}
       ListofPL1.Add(xKrd@YminF)
       ListofPL1.Add(xKrd@yKrd)
       ListofAchs.Add(ListofPL1)
       if (xSk = maxxSk) then
         ListofPts.Add((xKrd-(1000*SFkt))@ytextpt1)
         xNrStr=(xKrd.SetFormat("").SetFormat("d")).AsString
         ListofSchrift.Add(xNrStr)
       end
       ListofxSkKrd.Add(xKrd)
     end
     xKrd = xKrd + xSk ' Abstand der x-Skalenlinien
   end
   if (xSk = maxxSk) then
    aTest = ListofxSkKrd.FindByValue(XGK)
    if (aTest = -1) then
      ListofPL1={}
      ListofPL1.Add(XGK@YminF)
```

```
        ListofPL1.Add(XGK@yKrd)
        ListofAchs.Add(ListofPL1)
        ListofPts.Add((XGK-(1000*SFkt))@ytextpt1)
        xNrStr=(XGK.SetFormat("").SetFormat("d")).AsString
        ListofSchrift.Add(xNrStr)
      end
    end
end


'Herstellung einer Schrift der y-Skalenlinie
ListofySkKrd = {}

for each i in 0..IdxYSkL
    ySk = ListofySk.Get(i)
    yM = YminH Mod ySk
    if (yM = 0) then
      yKrd = YminH
    elseif (yM <> 0) then
      yKrd = ((YminH / ySk).Ceiling) * ySk
    end
    ySkL = ListofySkL.Get(i)
    xKrd = XKK-(ySkL*YFt)
    xKrdr = XGK+(ySkL*YFt)
    if (ySk = maxySk) Then
      if (yKrd <> YminH) then
        yKrdF = yKrd*YFt
        ListofPL1={}
        ListofPL1.Add(XKK@yKrdF)
        ListofPL1.Add(xKrd@yKrdF)
        ListofAchs.Add(ListofPL1)
        ListofPL1={}
        ListofPL1.Add(XGK@yKrdF)
        ListofPL1.Add(xKrdr@yKrdF)
        ListofySkKrd.Add(yKrd)
        ListofAchs.Add(ListofPL1)
        if ((ySk = maxySk) or (ySk = max2ySk)) then
          yNrStr=(yKrd.SetFormat("").SetFormat("d")).AsString
          AnzStr=yNrStr.Count
          if (AnzStr = 1) then
            xtextpt1=XKK-(axSkL*YFt*3)
          elseif (AnzStr = 2) then
            xtextpt1=XKK-(axSkL*YFt*3.3)
          elseif (AnzStr > 2) then
            xtextpt1=XKK-(axSkL*YFt*3.7)
          end
          ListofPts.Add(xtextpt1@yKrdF)
          ListofSchrift.Add(yNrStr)
        end
      end
    end
    YmaxHP = YmaxH + 0.01
    while (yKrd <  YmaxHP)
        aTest = ListofySkKrd.FindByValue(yKrd)
        if (aTest = -1) then
          yKrdF = yKrd*YFt
          ListofPL1={}
          ListofPL1.Add(XKK@yKrdF)
          ListofPL1.Add(xKrd@yKrdF)
          ListofAchs.Add(ListofPL1)
          ListofPL1={}
          ListofPL1.Add(XGK@yKrdF)
```

```
          ListofPL1.Add(xKrdr@yKrdF)
          ListofAchs.Add(ListofPL1)
          ListofySkKrd.Add(yKrd)
          if ((ySk = maxySk) or (ySk = max2ySk)) then
            yNrStr=(yKrd.SetFormat("").SetFormat("d")).AsString
            AnzStr=yNrStr.Count
            if (AnzStr = 1) then
              xtextpt1=XKK-(axSkL*YFt*3)
            elseif (AnzStr = 2) then
              xtextpt1=XKK-(axSkL*YFt*3.3)
            elseif (AnzStr > 2) then
              xtextpt1=XKK-(axSkL*YFt*3.7)
            end
            ListofPts.Add(xtextpt1@yKrdF)
            ListofSchrift.Add(yNrStr)
          end
        end
        yKrd = yKrd + ySk ' Abstand der y-Skalenlinie
    end
end

'Ein Feature-Shape-File für Achsenlinlien (Polyline) wird hergestellt
av.ShowMsg("Herstellung des neuen Themas für Achsen im Profilschnitt ...")
WDStr=theProject.GetWorkDir.AsString
fnStr0="Achsqpl1"
fnStr=FileName.Make(WDStr).MakeTmp(fnStr0,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output Shape File (Achsenlinien)")
if (fName=nil) then exit end
fName.SetExtension("shp")
G1tFTab=FTab.MakeNew(fName, Polyline)
theIDField=Field.Make("ID", #FIELD_SHORT, 2, 0)
G1tFTab.AddFields({theIDField})
theShapeField=G1tFTab.FindField("shape")
AnzAchsPL=ListofAchs.Count
IdxAchsPL=AnzAchsPL-1
G1tFTab.SetEditable(false)
G1tFTab.SetEditable(true)
for each AL in 0..IdxAchsPL
  ListofPt=ListofAchs.Get(AL)
  ListofListofPt={}
  ListofListofPt.Add(ListofPt)
  AchsPolyLg=PolyLine.Make(ListofListofPt)
  G1tFTab.AddRecord
  G1tFTab.SetValue(theShapeField, AL, AchsPolyLg)
  G1tFTab.SetValue(theIDField, AL, AL)
end
G1tFTab.SetEditable(false)
thmNew=FTheme.Make(G1tFTab)
theView.AddTheme(thmNew)

'Ein Feature-Shape-File für AchsenTicksLabel (Punkt) im Querprofilschnittsview wird hergestellt
fnStr0="Achsqps1"
fnStr=FileName.Make(WDStr).MakeTmp(fnStr0,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (AchsenTicksLabel)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PtFTab=FTab.MakeNew(fName, Point)
ShapeField=PtFTab.FindField("shape")
IDField=Field.Make("ID", #Field_Short, 4, 0)
XField=Field.Make("x-Krd", #Field_FLOAT, 10, 2)
YField=Field.Make("y-Krd", #Field_FLOAT, 10, 2)
LbField=Field.Make("Beschrift", #FIELD_CHAR, 15, 0)
```

```
ListofATLFlds={IDField, XField, YField, LbField}
PtFTab.AddFields(ListofATLFlds)
ListofPts.Add((XKK-(30*SFkt))@(YmaxF+(20*YFt*SFkt)))
ListofPts.Add((((XGK-XKK)/2)+XKK-(2000*SFkt))@(YminF-(50*YFt*SFkt)))
ListofSchrift.Add("[m ü NN]")
ListofSchrift.Add("Entfernung [m]")
AnzPts=ListofPts.Count
IdxPts=AnzPts-1
PtFTab.SetEditable(false)
PtFTab.SetEditable(true)
for each aPt in 0..IdxPts
    theShp=ListofPts.Get(aPt)
    ax=theShp.Getx.SetFormat("").SetFormat("d.dd")
    ay=theShp.Gety.SetFormat("").SetFormat("d.dd")
    thePtTxt=ListofSchrift.Get(aPt)
    PtFTab.AddRecord
    PtFTab.SetValue(ShapeField, aPt, theShp)
    PtFTab.SetValue(IDField, aPt, aPt)
    PtFTab.SetValue(XField, aPt, ax)
    PtFTab.SetValue(YField, aPt, ay)
    PtFTab.SetValue(LbField, aPt, thePtTxt)
end
PtFTab.SetEditable(false)
thmNew=FTheme.Make(PtFTab)
theView.AddTheme(thmNew)
theDisplay=theView.GetDisplay
theGraphicList=theView.GetGraphics
for each aPt in 0..IdxPts
  theGString=ListofSchrift.Get(aPt)
  theGPoint=ListofPts.Get(aPt)
  theGText=GraphicText.Make(theGString, theGPoint)
  theGText.SetAlignment(#TEXTCOMPOSER_JUST_CENTER)
  theTextSymbol=theGText.ReturnSymbols.Get(0)
  theTextSymbol.SetSize(12)
  newFont=Font.Make("Arial", "normal")
  theTextSymbol.SetFont(newFont)
  theTextSymbol.SetColor(Color.GetBlack)
  theGraphicList.Add(theGText)
end
theDisplay.Invalidate(true)
```

```
'arcvwxls.ave
'Daten eines Punkt-Themas in ArcView werden gelesen
'und in Excel geschrieben. Der Dezimalpunkt der Excel-Datei ist ",".
'Dieses Skript wird als ein Menü oder als eine Schaltfläche
'in einem aktiven View zum Anklicken benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc
ListofThemen=theView.GetThemes
ListofPtThms = {}
for each aT in ListofThemen
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
       ListofPtThms.Add(aT)
     end
   end
```

```
end

theTheme=MsgBox.ChoiceAsString(ListofPtThms,
      "Name eines Themas im View"++theView.AsString,
      "Auswahl eines Punkt-Themas")
theFTab=theTheme.GetFTab
theFields=theFTab.GetFields
AnzFlds=theFields.Count
IdxFlds=AnzFlds-1

'EgD2=MsgBox.Input("Eingabe der im Hintergrund laufenden Excel-Datei",
' "Dynamic-Data-Exchange", "tokbops3.xls")
systemClient=DDEClient.Make("Excel", "System")
if (systemClient.HasError) then
  MsgBox.error(systemClient.GetErrorMsg, "Error-Message")
  exit
end
systemClient.Execute("[NEW(1,0,false)]")
selection=systemClient.Request("selection")
spreadsheet=selection.Left(selection.IndexOf("!"))
systemClient.Execute("[workspace(,,true)]")
systemClient.Close
theClient=DDEClient.Make("Excel", spreadsheet)

'Feststellung der Anzahl der Datensätze im ArcView-Thema
AnzPt = 0
for each arec in theFTab
  AnzPt = AnzPt + 1
end
if (AnzPt <= 64000) then
  recNr=-1   'Anfangs-Index-Nummer der Datensätze - 1
  Anzdat=AnzPt
elseif (AnzPt > 64000) then
  aFakt = AnzPt / 64000
  aDateiZ = aFakt.Ceiling
  MsgBox.Report(
  "Anzahl der Datensätze im Thema"++theTheme.AsString+":"+NL+
   AnzPt.AsString+NL+"Die Anzahl der Datensätze"+NL+
  "ist größer als 64000. Deshalb sind die Daten"+NL+
  "mindestens in"++aDateiZ.AsString++"Excel-Dateien"+NL+
  "geteilt zu schreiben.", "Infomation")
  aAnfRecNr = MsgBox.Input("Anfangs-Index-Nummer der Datensätze"+NL+
                      "des Themas"++theTheme.AsString,
                      "Eingabe der Datensätze", "0")
  aEndRecNr = MsgBox.Input("Ende-Index-Nummer der Datensätze"+NL+
                      "des Themas"++theTheme.AsString,
                      "Eingabe der Datensätze", "15295")
  ListofEndsch = {"sicher", "nicht sicher"}
  aKrit = MsgBox.ListAsString(ListofEndsch,
        "Die Index-Nummer ist",
        "Bestätigung der Index-Nummer")
  if (aKrit = "sicher") then
     recNr = aAnfRecNr.AsNumber - 1
     AnzDat = aEndRecNr.AsNumber - aAnfRecNr.AsNumber + 1
  elseif (aKrit = "nicht sicher") then
    MsgBox.Info("Die Index-Nummer ist nicht sicher."+NL+
              "Deshalb wird das Programm jetzt abgebrochen!",
              "Information")
    exit
  end
end
```

```
AnfIdx=1   'Anfangs-Reihennummer der Excel-Datei - 1
EndIndex=AnfIdx+Anzdat-1  'Ende-Reihennummer der Excel-Datei - 1
Nr=0
av.ShowMsg("Daten im Thema"++theTheme.AsString
       ++"werden gelesen und in Excel geschrieben ...")
av.ShowStopButton
zstring=1.AsString
for each x in 1..IdxFlds
   sstring=x.AsString
   FeldN=theFields.Get(x)
   theClient.Poke("z"+zstring+"s"+sstring, FeldN)
end

for each i in AnfIdx..EndIndex
   zstring=(i+1).AsString
   recNr=recNr+1
   for each j in 1..IdxFlds
      aFld=theFields.Get(j)
      aFldType=aFld.GetType
      Value1=theFTab.ReturnValue(aFld, recNr)
      sstring=(j).AsString
      if (aFld.IsTypeNumber) then
        Value1=Value1.SetFormat("d.dd")
        vString1=Value1.AsString
        ListofDt=vString1.AsTokens(".")
        'MsgBox.ListAsString(ListofDt, "List of Tokens", "Daten")
        DZ=ListofDt.Count
        DtNr1=ListofDt.Get(0)
        DtNr2=ListofDt.Get(1)
        DtNr12=DtNr1+","+DtNr2
        'MsgBox.Info(DtNr12.AsString, "DatenNr")
        theClient.Poke("z"+zstring+"s"+sstring, DtNr12)
      elseif (aFld.IsTypeString) then
        theClient.Poke("z"+zstring+"s"+sstring, Value1)
      end
   end
   'Show percentage complete with enabled stop button
   more=av.SetStatus((i)/Anzdat*100)
   if (not more) then
     break
   end
end
theClient.Close
```

```
'bohrhypo.ave
'Ein Polygon-Thema für eine hypothetische Bohrung mit einer
'Überhöhung wird an der Stelle eines Punktes hergestellt.
'Dabei wird ein Punkt durch einen Maus-Klick auf dem Bildschirm
'oder durch eine Tastatureingabe eingegeben.
'Ein Schichtenmodell wird auch eingegeben.
'Dieses Script wird als Werkzeug (Tool) im aktiven View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc

aMousePt=theView.GetDisplay.ReturnUserPoint
ListofThms=theView.GetThemes
```

```
ListofPtAusw={"Der Punkt des Maus-Klickens",
            "Der Punkt durch Tastatur-Eingabe"}

'Auswahl eines Punktes zur Bestimmung der Stelle
'einer hypothetischen Bohrung
myPt=MsgBox.ChoiceAsString(ListofPtAusw, "zur Bestimmung der Stelle"
                    +NL+"einer hypothetischen Bohrung",
                    "Auswahl eines Punktes im Karten-View")
if (myPt = "Der Punkt des Maus-Klickens") then
  thePtx=aMousePt.Getx
  thePty=aMousePt.Gety
elseif (myPt = "Der Punkt durch Tastatur-Eingabe") then
  aMPtxStr=aMousePt.Getx.SetFormat("d.dd").AsString
  aMPtyStr=aMousePt.Gety.SetFormat("d.dd").AsString
  aTastPtx=MsgBox.Input("X-Koordinate des Punktes", "Eingabe des Punktes", aMPtxStr)
  aTastPty=MsgBox.Input("Y-Koordinate des Punktes", "Eingabe des Punktes", aMPtyStr)
  thePtx=aTastPtx.AsNumber
  thePty=aTastPty.AsNumber
end
thePt=Point.Make(thePtx, thePty)
theKr=Circle.Make(thePt, 10)
thePg=theKr.AsPolygon

'Eingabe einer Tabelle für ein Schichtenmodell des Gebietes
aTable=MsgBox.Input("für ein Schichtenmodell des Gebietes",
      "Eingabe einer Tabelle in Projekt", "Schmgebt.dbf")
aVTab = theProject.FindDoc(aTable).GetVTab
ListofFlds=aVTab.GetFields
aNmAbkFld=MsgBox.ChoiceAsString(ListofFlds, "das die Abkürzung des Namens"
       +NL+"der Schichten enthält",
       "Auswahl eines Feldes im:"++aTable.AsString)
aNamenFld=MsgBox.ChoiceAsString(ListofFlds, "das den Namen der Schichten enthält",
       "Auswahl eines Feldes im:"++aTable.AsString)
aSchmFld=MsgBox.ChoiceAsString(ListofFlds,
      "welches das Schichtenmodell der Schichten enthält",
      "Auswahl eines Feldes im:"++aTable.AsString)
aVerbFld=MsgBox.ChoiceAsString(ListofFlds,
      "welches die Verbreitungsgrenze der Schichten enthält",
      "Auswahl eines Feldes im:"++aTable.AsString)

'Bestimmung der Anzahl der Schichten im Schichtenmodell des Gebietes
AnzSch=0
for each rec in aVTab
   AnzSch=AnzSch+1
end
IdxSch=AnzSch-1
MsgBox.Report("im Schichtenmodell"++aTable.AsString+":"++AnzSch.AsString,
      "Anzahl der Schichten")

'Suche nach den Schichten, in deren Verbreitungsgrenze
'der eingegebene Punkt sich befindet.
ListofSchm={}
ListofAbks={}
ListofSNm={}
for each arec in 0..IdxSch
   aPgStr=aVTab.ReturnValue(aVerbFld, arec)
   aPgThm=theView.FindTheme(aPgStr)
   aPgFTab=aPgThm.GetFTab
   aPgVgrSchFld=aPgFTab.FindField("Shape")
   aPgVgr=aPgFTab.ReturnValue(aPgVgrSchFld, 0)
   Qt1=aPgVgr.Contains(thePt)
   if (Qt1) then
```

```
      aSchm=aVTab.ReturnValue(aSchmFld, arec)
      ListofSchm.Add(aSchm)
      aAbk=aVTab.ReturnValue(aNmAbkFld, arec)
      ListofAbks.Add(aAbk)
      aSNm=aVTab.ReturnValue(aNamenFld, arec)
      ListofSNm.Add(aSNm)
    end
end

'Anzahl der Schichten, die den eingegebenen Punkt enthalten.
AnzSchm=ListofSchm.Count
IdxSchm=AnzSchm-1
'Bestimmung der Ober- und Unterkante der Schichten am Punkt
ListofOKH={}
ListofUKH={}
ListofNmAbk={}
ListofSNm2={}
aPrj=Prj.MakeNull
av.ShowMsg("Bestimmung der Höhen für Bohrungen ...")
av.ShowStopButton
for each aS in 0..IdxSchm
  Ng=aS+1
  aSchm=ListofSchm.Get(aS)
  aAbk=ListofAbks.Get(aS)
  aSNm=ListofSNm.Get(aS)
  'Abfrage, ob es einen Punkt innerhalb der Entfernung
  'von 10 m vom Maus-Klicken gibt.
  theSchm=theView.FindTheme(aSchm)
  SchmFTab=theSchm.GetFTab
  theShpFld=SchmFTab.FindField("Shape")
  SchmFTab.SelectByPolygon(thePg, #VTAB_SELTYPE_NEW)
  AnzSel=SchmFTab.GetSelection.Count
  if (SchmFTab.GetSelection.Count <> 0) then
    'Für den Fall mit einem Punkt innerhalb der Entfernung von 10 m
    minAbst=1000
    ListofFTFlds=SchmFTab.GetFields
    aOKFld=MsgBox.ChoiceAsString(ListofFTFlds,
        "das die Höhen der Oberkante enthält"+NL+
        "(für Schicht:"++aSNm+")",
        "Auswahl des Feldes im Thema"++SchmFTab.AsString)
    aUKFld=MsgBox.ChoiceAsString(ListofFTFlds,
        "das die Höhen der Unterkante enthält"+NL+
        "(für Schicht:"++aSNm+")",
        "Auswahl des Feldes im Thema"++SchmFTab.AsString)
    for each rec in SchmFTab.GetSelection
      theShp=SchmFTab.ReturnValue(theShpFld, rec)
      theOkH=SchmFTab.ReturnValue(aOKFld, rec)
      theUkH=SchmFTab.ReturnValue(aUKFld, rec)
      theRW=theShp.Getx
      theHW=theShp.Gety
      Abstx=(theRW-thePtx)*(theRW-thePtx)
      Absty=(theHW-thePty)*(theHW-thePty)
      Abst=(Abstx+Absty).Sqrt.Abs
      if (Abst < minAbst) then
        minAbst=Abst
        myOKH=theOkH
        myUKH=theUkH
      end
    end
    ListofOKH.Add(myOKH)
    ListofUKH.Add(myUKH)
    ListofNmAbk.Add(aAbk)
```

```
  ListofSNm2.Add(aSNm)
elseif (SchmFTab.GetSelection.Count = 0) then
  'Für den Fall ohne einen Punkt innerhalb der Entfernung von 10 m
  ListofFlaeche={"Oberkante", "Unterkante"}
  for each aFL in 0..1
   aNr=aFL+1
   if (aFL = 0) then
     theFL="Oberkante"
   elseif (aFL = 1) then
     theFL="Unterkante"
   end
   if ((aS < IdxSchm) or (theFL = "Oberkante")) then
    'Eingabe der interpolierten Datei mit den Höhenwerten (Grid oder Tin)
    done=False
    While (not done)
     surfaceList = {}
     for each t2 in ListofThms
       if (t2.Is(GTheme) or t2.Is(STheme)) then
         surfaceList.Add(t2)
       end
     end
     if (surfaceList.Count = 0) then
       aSurfFN = SourceManager.GetDataSet({Grid,Tin},
              "Surface vom:" ++ theSchm.GetName++ theFL)
       if (aSurfFN = NIL) then
         continue
       end
       aSrcName = Grid.MakeSrcName(aSurfFN.AsString)
       if (aSrcName <> NIL) then
         theSurface = Grid.Make(aSrcName)
         surfTheme = GTheme.Make(theSurface)
       else
         aSrcName = SrcName.Make(aSurfFN.AsString)
         theSurface = Tin.Make(aSrcName)
         surfTheme = STheme.Make(theSurface)
       end
       theView.AddTheme(surfTheme)
     else
       surfTheme = MsgBox.ListAsString(surfaceList,
           "Auswahl des Themas, um als surface zu benutzen:",
           "Surface für:"++ theSchm.GetName ++ theFL)
       if (surfTheme = NIL) then
         continue
       end
       if (surfTheme.Is(GTheme)) then
         theSurface = surfTheme.GetGrid
       elseif (surfTheme.Is(STheme)) then
         theSurface = surfTheme.GetSurface
       else
         continue
       end
     end
     done=True
    end
    if (surfTheme.Is(GTheme)) then
      theZValue=theSurface.PointValue(thePt, aPrj)
      'MsgBox.Report("Das Schichtenmodell:"++theSchm.AsString+NL+
      '             "Fläche:"++theFL+NL+"Höhe [m ü NN]:"
      '              ++theZValue.AsString, "Kontrolle 1")
    elseif (surfTheme.Is(STheme)) then
      theZValue=theSurface.Elevation(thePt)
      'MsgBox.Report("Das Schichtenmodell:"++theSchm.AsString+NL+
```

```
          '              "Fläche:"++theFL+NL+"Höhe [m ü NN]:"
          '               ++theZValue.AsString, "Kontrolle 2")
        end
        if (theFL = "Oberkante") then
          ListofOKH.Add(theZValue)
        elseif (theFL = "Unterkante") then
          ListofUKH.Add(theZValue)
        end
      elseif ((aS = IdxSchm) and (theFL = "Unterkante")) then
        ListofUKH.Add(0)
        theZValue=0
        'MsgBox.Report("Das Schichtenmodell:"++theSchm.AsString+NL+
        '              "Fläche:"++theFL+NL+"Höhe [m ü NN]:"
        '               ++theZValue.AsString, "Kontrolle 3")
      end
    end
    ListofNmAbk.Add(aAbk)
    ListofSNm2.Add(aSNm)
  end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzSchm*100)
  if (not more) then
    break
  end
  theSchm.ClearSelection
end
'Auswahl eines Polygon-Themas zur Speicherung der Bohrungen
Qt33=MsgBox.YesNo("Gibt es schon ein Polygon-Thema"+NL+
                  "für Bohrungen im View"++theView.AsString,
                  "Speicherung der Bohrung", true)
if (Not Qt33) then
  WDStr=theProject.GetWorkDir.AsString
  fnStr=FileName.Make(WDStr).MakeTmp("Bohrbl00","shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Polygon)")
  if (fName=nil) then exit end
  fName.SetExtension("shp")
  PgFTab=FTab.MakeNew(fName, Polygon)
  PgShpFld=PgFTab.FindField("shape")
  PggPgIdFld=Field.Make("Pg_ID", #Field_Short, 5, 0)
  PgBIDFld=Field.Make("Bohr_ID", #Field_Short, 3, 0)
  PgSIDFld=Field.Make("Scht_ID", #Field_Short, 3, 0)
  PgRWFld=Field.Make("RW", #Field_Float, 8, 0)
  PgHWFld=Field.Make("HW", #Field_Float, 8, 0)
  PgNmAbkFld=Field.Make("Namen_Abk", #Field_Char, 10, 0)
  PgNmFld=Field.Make("Namen", #Field_Char, 30, 0)
  PgHmFld=Field.Make("Hoehe_m", #Field_Float, 8, 2)
  PgTmFld=Field.Make("Tiefe_m", #Field_Float, 8, 2)
  ListofPgFlds={PggPgIdFld, PgBIDFld, PgSIDFld, PgRWFld, PgHWFld,
                PgNmAbkFld, PgNmFld, PgHmFld, PgTmFld}
  PgFTab.AddFields(ListofPgFlds)
  recNr=-1
  BIDv=-1
elseif (Qt33) then
  ListofFThm = {}
  for each aT in ListofThms
    if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Polygon)) then
        ListofFThm.Add(aT)
      end
    end
  end
```

```
  theBThm=MsgBox.ChoiceAsString(ListofFThm,
  "das die Polygone der Bohrungen enthält", "Auswahl eines Themas")
  PgFTab=theBThm.GetFTab
  PgShpFld=PgFTab.FindField("shape")
  PggPgIdFld=PgFTab.FindField("Pg_ID")
  PgBIDFld=PgFTab.FindField("Bohr_ID")
  PgSIDFld=PgFTab.FindField("Scht_ID")
  PgRWFld=PgFTab.FindField("RW")
  PgHWFld=PgFTab.FindField("HW")
  PgNmAbkFld=PgFTab.FindField("Namen_Abk")
  PgNmFld=PgFTab.FindField("Namen")
  PgHmFld=PgFTab.FindField("Hoehe_m")
  PgTmFld=PgFTab.FindField("Tiefe_m")
  'Anzahl der vorhandenen Polygone
  AnzgPg=0
  for each aPgrec in PgFTab
    AnzgPg=AnzgPg+1
  end
  IdxgPg=AnzgPg-1
  recNr=IdxgPg
  BIDv=PgFTab.ReturnValue(PgBIDFld, IdxgPg)
end

'Herstellung der Bohrung
av.ShowMsg("Herstellung von Polygonen für Bohrungen ...")
ListofZeichn={"auf einem Karten-View", "auf einem Bohrungsdiagramm"}
Ausw11=MsgBox.ChoiceAsString(ListofZeichn,
    "wo die Bohrungen gezeichnet werden", "Auswahl einer Stelle")
'Anzahl der Oberkante
AnzObk=ListofOKH.Count
IdxObk=AnzObk-1
ListofPg={}
ListofSNAbk={}
ListofSNmPg={}
ListofHoehe={}
ListofTiefe={}
Tiefem=0
for each i in 0..IdxObk
  aObkm=ListofOKH.Get(i)
  aUkm=ListofUKH.Get(i)
  aAbk=ListofNmAbk.Get(i)
  aSNmPg=ListofSNm2.Get(i)
  if (Ausw11 = "auf einem Karten-View") then
    aObk=aObkm*100+thePty
    aUk=aUkm*100+thePty
    aRW=thePtx
  elseif (Ausw11 = "auf einem Bohrungsdiagramm") then
    aObk=aObkm*100
    aUk=aUkm*100
    aRW=5000+((BIDv+1)*10000)
  end
  aRWl=aRW-1000
  aRWr=aRW+1000
  aSPg=Polygon.Make({{aRWl@aObk, aRWr@aObk, aRWr@aUk, aRWl@aUk}})
  Qt22=aSPg.IsNull
  if (Not Qt22) then
    ListofPg.Add(aSPg)
    ListofSNAbk.Add(aAbk)
    ListofSNmPg.Add(aSNmPg)
    ListofHoehe.Add(aObkm)
    ListofTiefe.Add(Tiefem)
    Dicke=aObkm-aUkm
```

```
      Tiefem=Tiefem-Dicke
    end
end

AnzBohr=ListofPg.Count
IdxBohr=AnzBohr-1
av.ShowMsg("Speicherung der Bohrungen ...")
av.ShowStopButton
PgFTab.SetEditable(false)
PgFTab.SetEditable(true)
for each j in 0..IdxBohr
    Ng=j+1
    recNr=recNr+1
    aBPg=ListofPg.Get(j)
    BIDj=BIDv+1
    theRW=thePtx.SetFormat("d")
    theHW=thePty.SetFormat("d")
    aNmAbk=ListofSNAbk.Get(j)
    aSNmPg=ListofSNmPg.Get(j)
    theHoehe=ListofHoehe.Get(j)
    theTiefe=ListofTiefe.Get(j)
    PgFTab.AddRecord
    PgFTab.SetValue(PgShpFld, recNr, aBPg)
    PgFTab.SetValue(PggPgIdFld, recNr, recNr)
    PgFTab.SetValue(PgBIDFld, recNr, BIDj)
    PgFTab.SetValue(PgSIDFld, recNr, j)
    PgFTab.SetValue(PgRWFld, recNr, theRW)
    PgFTab.SetValue(PgHWFld, recNr, theHW)
    PgFTab.SetValue(PgNmAbkFld, recNr, aNmAbk)
    PgFTab.SetValue(PgNmFld, recNr, aSNmPg)
    PgFTab.SetValue(PgHmFld, recNr, theHoehe)
    PgFTab.SetValue(PgTmFld, recNr, theTiefe)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzBohr*100)
    if (not more) then
      break
    end
end
PgFTab.SetEditable(false)

if (Not Qt33) then
  NewThm=FTheme.Make(PgFTab)
  theView.AddTheme(NewThm)
  theTheme=NewThm
elseif (Qt33) then
  theBThm.UpdateLegend
  theTheme=theBThm
end

'Legende der Bohrungen als Stapel-Arbeit
av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette
FTabP=theTheme.GetFTab
AnzRecP=0
for each aRecP in FTabP
   AnzRecP=AnzRecP+1
end
if (AnzRecP <> 0) then
  theLegend=theTheme.GetLegend
  theLegend.SetLegendType(#Legend_Type_Unique)
  theLegend.Unique(theTheme, "Namen_Abk")
  ListofKlasse=theLegend.GetClassifications
```

```
  AnzKlasse=ListofKlasse.Count
  IdxKlasse=AnzKlasse-2
  'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
  aListofNr={}
  for each i in 0..IdxKlasse
     theKlasseLb=ListofKlasse.Get(i).GetLabel
     'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
     if (theKlasseLb="Deck") then
       theColorNr=53
     elseif (theKlasseLb="NT") then
       theColorNr=15
     elseif (theKlasseLb="lMTn") then
       theColorNr=27
     elseif (theKlasseLb="lMTs") then
       theColorNr=50
     elseif (theKlasseLb="rMTn") then
       theColorNr=21
     elseif (theKlasseLb="rMTm") then
       theColorNr=57
     elseif (theKlasseLb="rMTs1") then
       theColorNr=35
     elseif (theKlasseLb="rMTs2") then
       theColorNr=11
     else
       theColorNr=3
     end
     aListofNr.Add(theColorNr)
  end

  aListofSymbol=theLegend.GetSymbols
  AnzSymb=aListofSymbol.Count
  'MsgBox.Info(AnzSymb.AsString, "Anzahl der Symbole")
  AnzSymbIdx=AnzSymb-2
  aListofColor={}
  for each Nmb in 0..IdxKlasse
    aNumb=aListofNr.Get(Nmb)
    theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
    theRgbList=theColor.GetRgbList
    aColor=Color.Make
    aColor.SetRgbList(theRgbList)
    aListofColor.Add(aColor)
  end
  for each symb in 0..AnzSymbIdx
     aListofSymbol.Get(symb).SetColor(aListofColor.Get(symb))
  end
  theTheme.UpdateLegend
end
```

```
'fadenkbk.ave
'Ein Schnittpunkt zwischen einem Balken eines Fadenkreuzes und einem
'Profilschnitt wird berechnet und zur Bildung eines neuen Balkens
'im Fadenkreuz benutzt. Der Abschnitt des aktiven Profilschnittes wird
'durch zweimalige Klicken der Maus auf den Profilschnitt bestimmt.
'Dieses Script wird als ein Werkzeug (Tool) im aktiven View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc      'Ein aktives Profilschnitt-View
myScript=theProject.FindScript("fadenkbk")
```

```
myScript.SetNumberFormat( "d.dd") ' script default

aEingPolyL=theView.GetDisplay.ReturnUserPolyLine
theTheme=theView.GetActiveThemes.Get(0) 'Auswahl des aktiven Profilschnitt-Themas
ThStr=theTheme.AsString

Frg11=MsgBox.YesNo("Ist das aktive Thema"++theTheme.AsString
    ++"richtig?", "Feststellung des Themas", true)

if (Not Frg11) then
  MsgBox.Error("Das aktive Thema"++theTheme.AsString++"ist falsch!"+NL+
            "Das aktive Thema ist neu auszuwählen.", "")
  exit
end

ListofListofEingPt=aEingPolyL.AsList
ListofEingPt=ListofListofEingPt.Get(0)
AnzE=ListofEingPt.Count
AnzEIdx=AnzE-1
for each EPt in 0..AnzEIdx
   if (Ept = 0) then
     aMousePt=ListofEingPt.Get(EPt)
   elseif (Ept = AnzEIdx) then
     aMousePt2=ListofEingPt.Get(EPt)
   end
end
aMPtAnfx=aMousePt.Getx
aMPtAnfy=aMousePt.Gety
aMPtEndx=aMousePt2.Getx
aMPtEndy=aMousePt2.Gety
MsgBox.Report("Der Name des Themas: "++theTheme.AsString+NL+NL+
           "Der Anfang des Abschnittes am Maus-Klick"+NL+
           "Die X-Koordinate: "++aMPtAnfx.AsString+NL+
           "Die Y-Koordinate: "++aMPtAnfy.AsString+NL+NL+
           "Das Ende des Abschnittes am Maus-Klick"+NL+
           "Die X-Koordinate: "++aMPtEndx.AsString+NL+
           "Die Y-Koordinate: "++aMPtEndy.AsString,
           "Koordinaten an Maus-Klicken")
ListofMPt = {}
ListofMPt.Add(aMPtAnfx@aMPtAnfy)
ListofMPt.Add(aMPtEndx@aMPtEndy)

FTab1=theTheme.GetFTab
ShpFld1=FTab1.FindField("Shape")
ListofFlds = FTab1.GetFields
FLFld1 = MsgBox.ChoiceAsString(ListofFlds,
       "das den geologischen Namen enthält",
       "Auswahl eines Feldes im Thema"++theTheme.AsString)
IdxofFLFld1 = ListofFlds.FindByValue(FLFld1)

'MsgBox.Info(theTheme.AsString, "Der Name des Profilschnittes")

'Feststellung der Anzahl der 2D-PolyLine in dem Thema
Anz2DPL=0
for each rec in FTab1
   Anz2DPL=Anz2DPL+1
end
Anz2DPLIdx=Anz2DPL-1

ListofFlaeche = {}
for each aF in 0..Anz2DPLIdx
   aFlaeche = FTab1.ReturnValue(FLFld1, aF)
```

```
   ListofFlaeche.Add(aFlaeche)
end

aGFL = MsgBox.ListAsString(ListofFlaeche,
     "welche den gesuchten Punkt für ein Fadenkreuz enthält",
     "Auswahl einer Fläche")
aIdx = ListofFlaeche.FindByValue(aGFL)
thePlShp = FTab1.ReturnValue(ShpFld1, aIdx) 'Auswahl einer Fläche

minx = 3000000
maxx = 0

av.ShowMsg("Feststellung der Koordinaten des 2D-Profilschnittes im View"
     ++(theView.AsString)++"...")

theProfilList1={}
theProfilList1.Add({thePlShp})

ListofFLx = {}     'Liste der Vertices der geologischen Fläche
ListofFLy = {}
ListofFLPt = {}
minxkrd = 100000000
minykrd = 100000000
maxxkrd = 0
maxykrd = 0

if (theProfilList1 <> 0) then
  for each q in theProfilList1
     theLines=q.Get(0).AsList
     for each m in theLines
        for each ptx in m
           myx=ptx.Getx
           myy=ptx.Gety
           ListofFLx.Add(myx)
           ListofFLy.Add(myy)
           ListofFLPt.Add(myx@myy)
           if (myx < minxkrd) then
             minxkrd = myx
           end
           if (myx > maxxkrd) then
             maxxkrd = myx
           end
           if (myy < minykrd) then
             minykrd = myy
           end
           if (myy > maxykrd) then
             maxykrd = myy
           end
        end
     end
  end
end

PtAnz= ListofFLx.Count  'Anzahl der Stützpunkte der geologischen Fläche
PtAnzIdx= PtAnz-1

'Bestimmung des Abschnittes der geologischen Fläche als Index der
'Stützpunkte mit dem kleinsten Abstand von den Maus-Klicken
ListofGrPt = {}       'Liste für Grenz-Punkte für den Abschnitt
ListofListofGrIdx = {}  'Liste für Index der Grenz-Punkte
                    'für den Abschnitt
```

```
for each aM in 0..1   ' Anfang der Schleife für einen Abschnitt

minDist=1000
minIdx=-1

aMPt = ListofMPt.Get(aM)
aMPtx = aMPt.Getx
aMPty = aMPt.Gety

if (aMPtx < minxkrd) then
  aMPtx = minxkrd
elseif (aMPtx > maxxkrd) then
  aMPtx = maxxkrd
end
if (aMPty < minykrd) then
  aMPty = minykrd
elseif (aMPty > maxykrd) then
  aMPty = maxykrd
end

for each i in 0..PtAnzIdx
  xkrd= ListofFLx.Get(i)
  ykrd= ListofFLy.Get(i)
  xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
  yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
  xyAbst = ((xAbst  + yAbst).sqrt) .Abs
  if (xyAbst < minDist) then
    minDist = xyAbst
    minIdx = i  'Index des nähesten Punktes des Maus-Klickens
  end
end

'Bestimmung des Abschnittes der geologischen Fläche
'als x-, y-Koordinaten

if (minIdx = 0) then
   vIdx = 0     'ein Vertex auf der Linie vor dem Maus-Klicken
   nIdx = 1     'ein Vertex auf der Linie nach dem Maus-Klicken
elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
   xkrdv = ListofFLx.Get((minIdx -1))   'x-Koord. des letzten Punktes
   xkrd  = ListofFLx.Get(minIdx)        'x-Koord. des nähesten Punktes
   xkrdn = ListofFLx.Get((minIdx +1))  'x-Koord. des nächsten Punktes

   ykrdv = ListofFLy.Get((minIdx -1))   'y-Koord. des letzten Punktes
   ykrd = ListofFLy.Get(minIdx)         'y-Koord. des nähesten Punktes
   ykrdn = ListofFLy.Get((minIdx +1))   'y-Koord. des nächsten Punktes

   xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
   yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
   xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten Punkt
                                  'vom nähesten Punkt
   xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
   yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
   xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten Punkt
                                  'vom nähesten Punkte
   xML = (xkrd - aMPtx) * (xkrd - aMPtx)
   yML = (ykrd - aMPty) * (ykrd - aMPty)
   xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Maus-Klicken
                                  'vom nähesten Punkt
   xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
   yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
   xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Maus-Klicken
```

```
                                'vom letzten Punkt
    xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
    yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
    xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Maus-Klicken
                                'vom nächsten Punkt
    vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken
    vLpML = xyAv + xyML        'und dem letzten Punkt

    nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken
    nLpML = xyAn + xyML        'und dem nächsten Punkt

    vM = (vLmML - xyMLv).Abs   'Differenz zwischen der theorischen und der
    vP = (vLpML - xyMLv).Abs   'tatsächlichen Länge im Bezug auf den letzten
                          'Punkt
    nM = (nLmML - xyMLn).Abs   'Differenz zwischen der theorischen und der
    nP = (nLpML - xyMLn).Abs   'tatsächlichen Länge im Bezug auf den
                           'nächsten Punkt
    ListofLg = {vM, vP, nM, nP}   'Suche nach der kleinsten Länge
    MinLg = 10000
    TheLgIdx = -1
    For each aLg in 0..3
        theLg = ListofLg.Get(aLg)
        if (theLg < MinLg) then
            MinLg = theLg
            TheLgIdx = aLg
        end
    end
    if (xyML < 10) then
      vIdx = minIdx - 1
      nIdx = minIdx + 1
    end
    if (xyML >= 10) then
        if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
            vIdx = minIdx - 1  'ein Vertex auf der Linie vor dem Maus-Klicken
            nIdx= minIdx       'ein Vertex auf der Linie nach dem Maus-Klicken
        elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
            vIdx = minIdx      'ein Vertex auf der Linie vor dem Maus-Klicken
            nIdx = minIdx + 1  'ein Vertex auf der Linie nach dem Maus-Klicken
        else
            vIdx = minIdx    'ein Vertex auf der Linie vor dem Maus-Klicken
            nIdx = minIdx + 1 'ein Vertex auf der Linie nach dem Maus-Klicken
        end
    end
elseif (minIdx = PtAnzIdx)   then
     vIdx = minIdx - 1
     nIdx = minIdx
end

'Bestimmung der Koordinaten des Maus-Klickens auf dem Profilschnitt
vPtx = ListofFLx.Get(vIdx)
vPty = ListofFLy.Get(vIdx)
nPtx = ListofFLx.Get(nIdx)
nPty = ListofFLy.Get(nIdx)

if ((minDist = 0) or (minDist < 10)) then 'Der näheste Punkt liegt innerhalb
   Ptx = ListofFLx.Get(minIdx)     '10 m vom dem Maus-Klicken auf der Linie
   Pty = ListofFLy.Get(minIdx)
   if ( minIdx = 0) then
     vGrIdx = minIdx
     nGrIdx = minIdx + 1
   elseif (( minIdx > 0) and ( minIdx < PtAnzIdx)) then
     vGrIdx = minIdx -1
```

```
    nGrIdx = minIdx + 1
  elseif ( minIdx = PtAnzIdx) then
    vGrIdx = minIdx -1
    nGrIdx = minIdx
  end
elseif (minDist >= 10) then  'Berechnung der Koordinaten des Punktes
     vGrIdx = vIdx      'auf der Linie als Projektion des Maus-Klickens
     nGrIdx = nIdx          'auf die Linie (Profilschnitt)
     if  (vPtx  = aMPtx) then
         Ptx = vPtx
         Pty = vPty
     elseif ((vPtx  <> aMPtx) and (aMPtx <> nPtx)) then
         if (vPty = nPty) then
               Ptx = aMPtx
               Pty = vPty
         elseif (vPty < nPty) then
               if (nPtx = vPtx) then
                     Ptx = nPtx
                     Pty = nPty
               elseif (nPtx <> vPtx) then
                     xnvDiff = nPtx - vPtx
                     ynvDiff = nPty - vPty
                     xMvDiff = aMPtx - vPtx
                     Ptx = aMPtx
                     Pty  = (ynvDiff / xnvDiff) * xMvDiff + vPty
               end
         elseif (vPty > nPty) then
               if (nPtx = vPtx) then
                     Ptx = nPtx
                     Pty = nPty
               elseif (nPtx <> vPtx) then
                     xnvDiff = nPtx - vPtx
                     yvnDiff = vPty - nPty
                     xnMDiff = nPtx - aMPtx
                     Ptx = aMPtx
                     Pty  = (yvnDiff / xnvDiff) * xnMDiff + nPty
               end
         end
     elseif (nPtx  = aMPtx)  then
         Ptx = nPtx
         Pty = nPty
     end
end              ' Ende von (if ((minDist = 0) or (minDist < 10)) then)
ListofGrPt.Add(Ptx@Pty)       'Ein Grenz-Punkt für den Abschnitt

ListofGrIdx = {}
ListofGrIdx.Add(vGrIdx)
ListofGrIdx.Add(nGrIdx)
ListofListofGrIdx.Add(ListofGrIdx)

end                           ' Ende der Schleife für Abschnitt

PtAnz = ListofFLx.Count  'Anzahl der Vertices der geologischen Fläche
PtAnzIdx = PtAnz-1

aGrVIdx = ListofListofGrIdx.Get(0).Get(0)
aGrNIdx = ListofListofGrIdx.Get(1).Get(1)

aGrPtV = ListofGrPt.Get(0)
aGrPtN = ListofGrPt.Get(1)

aGrPtVx = aGrPtV.Getx
```

```
aGrPtVy = aGrPtV.Gety
aGrPtNx = aGrPtN.Getx
aGrPtNy = aGrPtN.Gety

ListofSPfx = {}
ListofSPfy = {}
ListofSPfx.Add(aGrPtVx)
ListofSPfy.Add(aGrPtVy)

for each i in 0..PtAnzIdx
  ax = ListofFLx.Get(i)
  ay = ListofFLy.Get(i)
  if ((i > aGrVIdx) and (i < aGrNIdx)) then
    ListofSPfx.Add(ax)
    ListofSPfy.Add(ay)
  end
end
ListofSPfx.Add(aGrPtNx)
ListofSPfy.Add(aGrPtNy)

'Eingabe einer Fadenkrez-Polyline
ListofThms=theView.GetThemes
ListofPLThms = {}

for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     aClNm = aFTab.GetShapeClass.GetClassName
     if (aClNm = "PolyLine") then
       ListofPLThms.Add(aT)
     end
   end
end

KrTheme=MsgBox.ChoiceAsString(ListofPLThms,
       "um einen Schnittpunkt mit dem Profilschnitt zu finden",
       "Eingabe eines Fadenkreuz-Themas ")

if (KrTheme = nil) then
   MsgBox.Error("Es gibt kein Fadenkreuz-Thema!"+NL+
            "Das Programm wird abgebrochen!", "")
   Exit
end

if (KrTheme <> nil) then
  KrFTab=KrTheme.GetFTab
  KrShpFld=KrFTab.FindField("Shape")
  KrIdFld=KrFTab.FindField("Id")
  ListofKrPtx={}
  ListofKrPty={}
  for each i in 0..1
    aPL=KrFTab.ReturnValue(KrShpFld, i)
    ListofKrPL={}
    ListofKrPL.Add({aPL})
    for each Lst in ListofKrPL
      theLs=Lst.Get(0).AsList
      for each L in theLs
          for each ptx in L
             myx = ptx.Getx
             myy = ptx.Gety
             ListofKrPtx.Add(myx)
             ListofKrPty.Add(myy)
```

```
      end
    end
  end
end
Ptx0 = ListofKrPtx.Get(0)
Anzx0 = 0
for each i in 0..3
  axkrd = ListofKrPtx.Get(i)
  if (axkrd = Ptx0) then
    Anzx0 = Anzx0 + 1
  end
end

Ptx1 = ListofKrPtx.Get(1)
if (Anzx0 = 1) then
  recNrwBk = 0
  xw0 = Ptx0
  xw1 = Ptx1
  yw0 = ListofKrPty.Get(0)
  yw1 = ListofKrPty.Get(1)
  wBkPt0 = Point.Make(xw0, yw0)
  wBkPt1 = Point.Make(xw1, yw1)
  ListofwBkPt={}
  ListofwBkPt.Add(wBkPt0)
  ListofwBkPt.Add(wBkPt1)
  ListofListofBkPt={}
  ListofListofBkPt.Add(ListofwBkPt)
  theWBPolyL=PolyLine.Make(ListofListofBkPt)  'Waagerechter Balken

  recNrsBk = 1
  xs2 = ListofKrPtx.Get(2)
  ys2 = ListofKrPty.Get(2)
  xs3 = ListofKrPtx.Get(3)
  ys3 = ListofKrPty.Get(3)
  sBkPt2 = Point.Make(xs2, ys2)
  sBkPt3 = Point.Make(xs3, ys3)
  ListofsBkPt={}
  ListofsBkPt.Add(sBkPt2)
  ListofsBkPt.Add(sBkPt3)
  ListofListofBkPt={}
  ListofListofBkPt.Add(ListofsBkPt)
  theSBPolyL=PolyLine.Make(ListofListofBkPt) 'Senkrechter Balken

elseif (Anzx0 > 1) then
  recNrsBk = 0
  xs0 = Ptx0
  xs1 = Ptx1
  ys0 = ListofKrPty.Get(0)
  ys1 = ListofKrPty.Get(1)
  sBkPt0 = Point.Make(xs0, ys0)
  sBkPt1 = Point.Make(xs1, ys1)
  ListofsBkPt={}
  ListofsBkPt.Add(sBkPt0)
  ListofsBkPt.Add(sBkPt1)
  ListofListofBkPt={}
  ListofListofBkPt.Add(ListofsBkPt)
  theSBPolyL=PolyLine.Make(ListofListofBkPt)  'Senkrechter Balken

  recNrwBk = 1
  xw2 = ListofKrPtx.Get(2)
  yw2 = ListofKrPty.Get(2)
  xw3 = ListofKrPtx.Get(3)
```

```
    yw3 = ListofKrPty.Get(3)
    wBkPt2 = Point.Make(xw2, yw2)
    wBkPt3 = Point.Make(xw3, yw3)
    ListofwBkPt={}
    ListofwBkPt.Add(wBkPt2)
    ListofwBkPt.Add(wBkPt3)
    ListofListofBkPt={}
    ListofListofBkPt.Add(ListofwBkPt)
    theWBPolyL=PolyLine.Make(ListofListofBkPt) 'Waagerechter Balken

  end

  MsgBox.Report("des Themas"++KrTheme.AsString+NL+NL+
            "Der eingegebene, waagerechte Balken:"+NL+
             theWBPolyL.AsString+NL+
             "Der eingegebene, senkrechte Balken:"+NL+
             theSBPolyL.AsString, "Der Balken als PolyLine")

  AW1=MsgBox.YesNo("Sind die Daten richtig? ",
    "Kontrolle der Daten", TRUE)
  if (Not AW1) then
    MsgBox.Error("Die Daten sind nicht richtig!"+NL+
             "Das Fadenkreuz soll neu eingegeben werden!", "")
    Exit
  end
end

'Auswahl des Balkens eines Fadenkreuzes zur Herstellung
ListofBalkens = {"der senkrechter Balken", "der waagerechter Balken"}

aBalk = MsgBox.ListAsString(ListofBalkens,
 "der aus einem Schnittpunkt zwischen dem Profilschnitt und"
 ++"einem anderen Balken hergestellt wird",
 "Auswahl eines Balkens")

AnzSPfx=ListofSPfx.Count
SPfxIdx=AnzSPfx-1

if (aBalk = "der senkrechter Balken") then
  awPt0 = ListofwBkPt.Get(0)
  yli = awPt0.Gety
  For each BkPt in 0..SPfxIdx
    Prfx1=ListofSPfx.Get(BkPt)
    Prfy1=ListofSPfy.Get(BkPt)
    if (BkPt < SPfxIdx) then
      Prfx2=ListofSPfx.Get(BkPt+1)
      Prfy2=ListofSPfy.Get(BkPt+1)
      if (Prfx1 > Prfx2) then
        Prfgx=Prfx1
        Prfkx=Prfx2
      elseif (Prfx2 > Prfx1) then
        Prfgx=Prfx2
        Prfkx=Prfx1
      end
      if (Prfy1 > Prfy2) then
        Prfgy=Prfy1
        Prfky=Prfy2
      elseif (Prfy2 > Prfy1) then
        Prfgy=Prfy2
        Prfky=Prfy1
      end
      if ((yli > Prfky) and (yli < Prfgy)) then
```

```
        PtSBy=yli
        if (Prfx1 <> Prfx2) then
          if (Prfy2 > Prfy1) then
            PtSBx=((((yli-Prfky).Abs)/((Prfgy-Prfky).Abs))
                 *(Prfgx-Prfkx).Abs)+Prfkx
          elseif (Prfy1 > Prfy2) then
            PtSBx=Prfgx-(((((yli-Prfky).Abs)/((Prfgy-Prfky).Abs))
                 *(Prfgx-Prfkx).Abs))
          end
        elseif (Prfx1 = Prfx2) then
          PtSBx=Prfx1
        end
      elseif (Prfy1 = yli) then
        PtSBx=Prfx1
        PtSBy=yli
      end
    elseif (BkPt = SPfxIdx) then
      if (Prfy1 = yli) then
        PtSBx=Prfx1
        PtSBy=yli
      end
    end
  end


  MsgBox.Report("Der Schnittpunkt zwischen"+NL+
          "dem waagerechten Balken und dem Profilschnitt"+NL+NL+
          "Die X-Koordinate: "++PtSBx.AsString+NL+
          "Die Y-Koordinate: "++PtSBy.AsString,
          "Der Punkt am Fadenkreuz")
  AW2=MsgBox.YesNo("Sind die Daten richtig? ",
     "Kontrolle der Daten", TRUE)
  if (Not AW2) then
    MsgBox.Error("Die Daten sind nicht richtig!"+NL+
           "Die Daten sollen neu eingegeben werden!", "")
    Exit
  end

  'MsgBox.Info("Speicherung des berechneten, senkrechten Balkens",
  '       "Der nächste Schritt")
  theYob=10000.00
  theYun=500.00
  ListofBkPt={}
  ListofBkPt.Add(PtSBx@theYob)
  ListofBkPt.Add(PtSBx@theYun)
  ListofListofBkPt={}
  ListofListofBkPt.Add(ListofBkPt)
  theSBPolyL=PolyLine.Make(ListofListofBkPt)

  recNr = recNrsBk
  KrFTab.SetEditable(false)
  KrFTab.SetEditable(true)
  KrFTab.SetValue(KrShpFld, recNr, theSBPolyL)
  KrFTab.SetEditable(false)

elseif (aBalk = "der waagerechter Balken") then
  asPt0 = ListofsBkPt.Get(0)
  xob = asPt0.Getx
  For each BkPt in 0..SPfxIdx
    Prfx1=ListofSPfx.Get(BkPt)
    Prfy1=ListofSPfy.Get(BkPt)
    if (BkPt < SPfxIdx) then
```

```
    Prfx2=ListofSPfx.Get(BkPt+1)
    Prfy2=ListofSPfy.Get(BkPt+1)
    if (Prfx1 > Prfx2) then
      Prfgx=Prfx1
      Prfkx=Prfx2
    elseif (Prfx2 > Prfx1) then
      Prfgx=Prfx2
      Prfkx=Prfx1
    end
    if (Prfy1 > Prfy2) then
      Prfgy=Prfy1
      Prfky=Prfy2
    elseif (Prfy2 > Prfy1) then
      Prfgy=Prfy2
      Prfky=Prfy1
    end
    if ((xob > Prfkx) and (xob < Prfgx)) then
      PtWBx=xob
      if (Prfy1 <> Prfy2) then
        if (Prfy2 > Prfy1) then
          PtWBy=((((xob-Prfkx).Abs)/((Prfgx-Prfkx).Abs))
              *(Prfgy-Prfky).Abs)+Prfky
        elseif (Prfy1 > Prfy2) then
          PtWBy=((((Prfgx-xob).Abs)/((Prfgx-Prfkx).Abs))
              *(Prfgy-Prfky).Abs)+Prfky
        end
      elseif (Prfy1 = Prfy2) then
        PtWBy=Prfy1
      end
    elseif (Prfx1 = xob) then
      PtWBx=Prfx1
      PtWBy=Prfy1
    end
  elseif (BkPt = SPfxIdx) then
    if (Prfx1 = xob) then
      PtWBx=Prfx1
      PtWBy=Prfy1
    end
  end
end
MsgBox.Report("Der Schnittpunkt zwischen"+NL+
        "dem senkrechten Balken und dem Profilschnitt"+NL+NL+
        "Die X-Koordinate: "++PtWBx.AsString+NL+
        "Die Y-Koordinate: "++PtWBy.AsString,
        "Der Punkt am Fadenkreuz")
AW2=MsgBox.YesNo("Sind die Daten richtig? ",
        "Kontrolle der Daten", TRUE)
if (Not AW2) then
  MsgBox.Error("Die Daten sind nicht richtig!"+NL+
          "Die Daten sollen neu eingegeben werden!", "")
  Exit
end

'MsgBox.Info("Speicherung des berechneten, waagerechten Balkens",
'      "Der nächste Schritt")

theXl=PtWBx-10000.00
theXr=PtWBx+10000.00
ListofBkPt={}
ListofBkPt.Add(theXl@PtWBy)
ListofBkPt.Add(theXr@PtWBy)
ListofListofBkPt={}
```

```
  ListofListofBkPt.Add(ListofBkPt)
  theWBPolyL=PolyLine.Make(ListofListofBkPt)

  recNr = recNrwBk
  KrFTab.SetEditable(false)
  KrFTab.SetEditable(true)
  KrFTab.SetValue(KrShpFld, recNr, theWBPolyL)
  KrFTab.SetEditable(false)
end
KrTheme.UpdateLegend



'flherst1.ave
'Eine Fläche aus den Rasterpunkten wird hergestellt.
'Die Höhen der Rasterpunkte werden aus den Höhen der Endpunkte
'und dem Gefälle in der Nord-Süd Richtung bestimmt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc 'Karten-View, wo ein neues FThema entsteht.

WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("Flmmtbs1","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName=nil) then exit end
fName.SetExtension("shp")

FLFTab=FTab.MakeNew(fName, Point)
FLShpFld=FLFTab.FindField("shape")
FLRWFld=Field.Make("RW", #Field_Float, 10, 2)
FLHWFld=Field.Make("HW", #Field_Float, 10, 2)
FLHmFld=Field.Make("Hoehem", #Field_Float, 6, 2)
ListofFLFld={FLRWFld, FLHWFld, FLHmFld}
FLFTab.AddFields(ListofFLFld)

'Feature-Shape-File für eine Fläche (Punkte) wird hergestellt.
av.ShowMsg("Herstellung der Punkte für eine Fläche ...")
av.ShowStopButton
'Eingabe der Gauß-Krüger-Koordinaten
RWAnfStr=MsgBox.Input("Der kleinste RW", "Eingabe von RW", "2558600.00")
RWEndStr=MsgBox.Input("Der größte RW", "Eingabe von RW", "2582450.00")
HWAnfStr=MsgBox.Input("Der kleinste HW", "Eingabe von HW", "5618450.00")
HWEndStr=MsgBox.Input("Der größte HW", "Eingabe von HW", "5641050.00")
AbstStr=MsgBox.Input(" der Punkte zwischen RW und HW",
      "Eingabe vom Raster-Abstand [m]", "50.00")
AnfHoehemStr=MsgBox.Input("des Punktes am Anfang von HW"++HWAnfStr,
      "Eingabe der Hoehe [m]", "37.07")
EndHoehemStr=MsgBox.Input("des Punktes am Ende von HW"++HWEndStr,
      "Eingabe der Hoehe [m]", "20.80")
GfStr=MsgBox.Input("in Grad", "Eingabe eines Gefälles", "0.04063587")
RWAnf=RWAnfStr.AsNumber.SetFormat("d.dd")
RWEnd=RWEndStr.AsNumber.SetFormat("d.dd")
HWAnf=HWAnfStr.AsNumber.SetFormat("d.dd")
HWEnd=HWEndStr.AsNumber.SetFormat("d.dd")
Abst=AbstStr.AsNumber.SetFormat("d.dd")
AnfHm=AnfHoehemStr.AsNumber.SetFormat("d.ddddddddd")
EndHm=EndHoehemStr.AsNumber.SetFormat("d.ddddddddd")
Gf=GfStr.AsNumber.SetFormat("d.ddddddddd")
GfRad=Gf.AsRadians
```

```
TangentHm=GfRad.Tan

'Anzahl der Punkte
AbstRWGz=RWEnd-RWAnf
AbstHWGz=HWEnd-HWAnf
AnzRWPt=((AbstRWGz/Abst)+1).SetFormat("d")
AnzHWPt=((AbstHWGz/Abst)+1).SetFormat("d")
AnzPt=(AnzRWPt*AnzHWPt).SetFormat("d")
IdxRWPt=AnzRWPt-1
IdxHWPt=AnzHWPt-1
if (AnfHm > EndHm) then
  Vorz=-1
elseif (AnfHm <= EndHm) then
  Vorz=1
end

FLFTab.SetEditable(false)
FLFTab.SetEditable(true)
Ng=0
recNr=-1
for each j in 0..IdxHWPt
  theHW=HWAnf+(j*Abst)
  dHm=((Abst*j)*TangentHm*Vorz).SetFormat("d.dddddddd")
  nHm=(AnfHm+dHm).SetFormat("d.dddddddd")
  for each i in 0..IdxRWPt
    Ng=Ng+1
    theRW=RWAnf+(i*Abst)
    thePt=Point.Make(theRW, theHW)
    theRW2=theRW.SetFormat("d.dd")
    theHW2=theHW.SetFormat("d.dd")
    theHm2=nHm.SetFormat("d.dd")
    recNr=recNr+1
    FLFTab.AddRecord
    FLFTab.SetValue(FLShpFld, recNr, thePt)
    FLFTab.SetValue(FLRWFld, recNr, theRW2)
    FLFTab.SetValue(FLHWFld, recNr, theHW2)
    FLFTab.SetValue(FLHmFld, recNr, theHm2)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzPt*100)
    if (not more) then
      break
    end
  end
end
FLFTab.SetEditable(false)
thmNew=FTheme.Make(FLFTab)
theView.AddTheme(thmNew)



'flvergl1.ave
'Die Höhen der Rasterpunkte der zwei Flächen werden miteinander verglichen und
'dann wird eine Fläche aus den Rasterpunkten hergestellt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc       'Karten-View, wo ein neues FThema entsteht.

ListofThms=theView.GetThemes
ListofFThm = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
```

```
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofFThm.Add(aT)
      end
    end
end
QbTheme=MsgBox.ChoiceAsString(ListofFThm, "für die 1. Fläche zum Vergleich",
      "Auswahl eines Themas im View"++theView.AsString)
QbFTab=QbTheme.GetFTab
LofQbFlds=QbFTab.GetFields
QbShpFld=MsgBox.ChoiceAsString(LofQbFlds, "Feld für Punkt-Shape",
                    "Eingabe vom Feld in"++QbTheme.AsString)
QbRWFld=MsgBox.ChoiceAsString(LofQbFlds, "Feld für RW",
                    "Eingabe vom Feld in"++QbTheme.AsString)
QbHWFld=MsgBox.ChoiceAsString(LofQbFlds, "Feld für HW",
                    "Eingabe vom Feld in"++QbTheme.AsString)
QbHmFld=MsgBox.ChoiceAsString(LofQbFlds, "Feld für Höhe [m]",
                    "Eingabe vom Feld in"++QbTheme.AsString)

FLMTTheme=MsgBox.ChoiceAsString(ListofFThm, "für die 2. Fläche zum Vergleich",
      "Auswahl eines Themas im View"++theView.AsString)
MTFTab=FLMTTheme.GetFTab
LofMTFlds=MTFTab.GetFields
MTShpFld=MsgBox.ChoiceAsString(LofMTFlds, "Feld für Punkt-Shape",
                    "Eingabe vom Feld in"++FLMTTheme.AsString)
MTRWFld=MsgBox.ChoiceAsString(LofMTFlds, "Feld für RW",
                    "Eingabe vom Feld in"++FLMTTheme.AsString)
MTHWFld=MsgBox.ChoiceAsString(LofMTFlds, "Feld für HW",
                    "Eingabe vom Feld in"++FLMTTheme.AsString)
MTHmFld=MsgBox.ChoiceAsString(LofMTFlds, "Feld für Höhe [m]",
                    "Eingabe vom Feld in"++FLMTTheme.AsString)

WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("Vgmmtbs1","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName=nil) then exit end
fName.SetExtension("shp")
FLFTab=FTab.MakeNew(fName, Point)
FLShpFld=FLFTab.FindField("shape")
FLRWFld=Field.Make("RW", #Field_Float, 10, 2)
FLHWFld=Field.Make("HW", #Field_Float, 10, 2)
FLHmFld=Field.Make("Hoehe_m", #Field_Float, 6, 2)
ListofFLFld={FLRWFld, FLHWFld, FLHmFld}
FLFTab.AddFields(ListofFLFld)
'Feature-Shape-File für eine Fläche (Punkte) wird hergestellt
av.ShowMsg("Vergleich der Punkte der beiden Flächen ...")
av.ShowStopButton
'Eingabe der Gauß-Krüger-Koordinaten
RWAnfStr=MsgBox.Input("Der kleinste RW", "Eingabe von RW", "2558600.00")
RWEndStr=MsgBox.Input("Der größte RW", "Eingabe von RW", "2582450.00")
HWAnfStr=MsgBox.Input("Der kleinste HW", "Eingabe von HW", "5618450.00")
HWEndStr=MsgBox.Input("Der größte HW", "Eingabe von HW", "5641050.00")
AbstStr=MsgBox.Input(" der Punkte zwischen RW und HW", "Eingabe von Abstand [m]",
"50.00")
RWAnf=RWAnfStr.AsNumber.SetFormat("d.dd")
RWEnd=RWEndStr.AsNumber.SetFormat("d.dd")
HWAnf=HWAnfStr.AsNumber.SetFormat("d.dd")
HWEnd=HWEndStr.AsNumber.SetFormat("d.dd")
Abst=AbstStr.AsNumber.SetFormat("d.dd")
'Anzahl der Punkte
AbstRWGz=RWEnd-RWAnf
AbstHWGz=HWEnd-HWAnf
```

```
AnzRWPt=((AbstRWGz/Abst)+1).SetFormat("d")
AnzHWPt=((AbstHWGz/Abst)+1).SetFormat("d")
AnzPt=(AnzRWPt*AnzHWPt).SetFormat("d")
IdxRWPt=AnzRWPt-1
IdxHWPt=AnzHWPt-1
FLFTab.SetEditable(false)
FLFTab.SetEditable(true)
Ng=0
recNr=-1
for each j in 0..IdxHWPt
  theHW=HWAnf+(j*Abst)
  for each i in 0..IdxRWPt
    Ng=Ng+1
    recNr=recNr+1
    theRW=RWAnf+(i*Abst)
    thePt=Point.Make(theRW, theHW)
    theRW2=theRW.SetFormat("d.dd")
    theHW2=theHW.SetFormat("d.dd")
    QbRWR=QbFTab.ReturnValue(QbRWFld, recNr)
    QbHWR=QbFTab.ReturnValue(QbHWFld, recNr)
    QbHmR=QbFTab.ReturnValue(QbHmFld, recNr)
    MTRWR=MTFTab.ReturnValue(MTRWFld, recNr)
    MTHWR=MTFTab.ReturnValue(MTHWFld, recNr)
    MTHmR=MTFTab.ReturnValue(MTHmFld, recNr)
    QbRW=QbRWR.SetFormat("d.dd")
    QbHW=QbHWR.SetFormat("d.dd")
    QbHm=QbHmR.SetFormat("d.dd")
    MTRW=MTRWR.SetFormat("d.dd")
    MTHW=MTHWR.SetFormat("d.dd")
    MTHm=MTHmR.SetFormat("d.dd")
    if ((QbRW = MTRW) and (QbHW = MTHW)) then
      theHm2=(QbHm-MTHm).SetFormat("d.dd")  'Quartärbasis-Höhe einer Fläche
    else
      theHm2=(-99).SetFormat("d.dd")
    end
    FLFTab.AddRecord
    FLFTab.SetValue(FLShpFld, recNr, thePt)
    FLFTab.SetValue(FLRWFld, recNr, theRW2)
    FLFTab.SetValue(FLHWFld, recNr, theHW2)
    FLFTab.SetValue(FLHmFld, recNr, theHm2)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzPt*100)
    if (not more) then
      break
    end
  end
end
FLFTab.SetEditable(false)
thmNew=FTheme.Make(FLFTab)
theView.AddTheme(thmNew)
```

```
'gitt2dkt.ave
'2D-Gitterlinien werden für eine Karte in einem
'bestimmten Abstand von RW und HW als ein Thema hergestellt.
'Dieses Script wird als ein Menü zum Anklicken
'in einem aktiven Karten-View benutzt.

theProject=av.GetProject
```

```
theView=av.GetActiveDoc  'ein aktives Karten-View
myScript=theProject.FindScript("gitt2dkt")
myScript.SetNumberFormat( "d.dddd") ' script default

av.ShowMsg("Bestimmung der Koordinaten des Karten-Rahmens ...")
ListofRW = {"2558600.0000", "2582450.0000",
          "2563000.0000", "2568000.0000",
          "2577000.0000", "2580500.0000"}

ListofHW = {"5618450.0000", "5641050.0000",
          "5637000.0000", "5641000.0000",
          "5637900.0000", "5641000.0000"}

akRWStr = MsgBox.ChoiceAsString(ListofRW, "der kleinste RW",
      "Auswahl der Koordinaten des Rahmens")
agRWStr = MsgBox.ChoiceAsString(ListofRW, "der größte RW",
      "Auswahl der Koordinaten des Rahmens")
akHWStr = MsgBox.ChoiceAsString(ListofHW, "der kleinste HW",
      "Auswahl der Koordinaten des Rahmens")
agHWStr = MsgBox.ChoiceAsString(ListofHW, "der größte HW",
      "Auswahl der Koordinaten des Rahmens")

'Veränderungsmöglichkeit
kRWStr = MsgBox.Input("der kleinste RW (einen anderen Wert?)",
    "Veränderungsmöglichkeit des Karten-Rahmens", akRWStr)
gRWStr = MsgBox.Input("der größte RW (einen anderen Wert?)",
    "Veränderungsmöglichkeit des Karten-Rahmens", agRWStr)
kHWStr = MsgBox.Input("der kleinste HW (einen anderen Wert?)",
    "Veränderungsmöglichkeit des Karten-Rahmens", akHWStr)
gHWStr = MsgBox.Input("der größte HW (einen anderen Wert?)",
    "Veränderungsmöglichkeit des Karten-Rahmens", agHWStr)

kRW = kRWStr.AsNumber
gRW = gRWStr.AsNumber
kHW = kHWStr.AsNumber
gHW = gHWStr.AsNumber

ListofRW = {"2558650.0000","2558700.0000","2558800.0000",
          "2559000.0000","2560000.0000",
          "2563050.0000","2563100.0000","2563200.0000",
          "2563500.0000","2564000.0000",
          "2577050.0000","2577100.0000","2577200.0000",
          "2577500.0000","2578000.0000"}

ListofHW = {"5618500.0000","5618600.0000","5619000.0000",
          "5620000.0000",
          "5637050.0000","5637100.0000","5637200.0000",
          "5637500.0000","5638000.0000",
          "5637950.0000","5638000.0000"}

av.ShowMsg("Bestimmung der Koordinaten der Gitterlinien ...")
akRWStr = MsgBox.ChoiceAsString(ListofRW, "der kleinste RW"
  +NL+"(der kleinste RW des Karten-Rahmens:"++kRW.AsString+")",
  "Auswahl der Koordinaten der RW-Gitterlinien")
akHWStr = MsgBox.ChoiceAsString(ListofHW, "der kleinste HW"
  +NL+"(der kleinste HW des Karten-Rahmens:"++kHW.AsString+")",
  "Auswahl der Koordinaten der HW-Gitterlinien")

'Veränderungsmöglichkeit
kRWStr = MsgBox.Input("der kleinste RW (einen anderen Wert?)"
  +NL+"(der kleinste RW des Karten-Rahmens:"++kRW.AsString+")",
  "Veränderungsmöglichkeit der RW-Gitterlinien", akRWStr)
```

```
kHWStr = MsgBox.Input("der kleinste HW (einen anderen Wert?)"
  +NL+"(der kleinste HW des Karten-Rahmens:"++kHW.AsString+")",
     "Veränderungsmöglichkeit der HW-Gitterlinien", akHWStr)

kRWGt = kRWStr.AsNumber
kHWGt = kHWStr.AsNumber

'Bestimmung eines Abstandes der Gitterlinien
ListofGAbst = {"50.0000", "100.0000", "200.0000", "400.0000",
               "500.0000", "1000.0000", "2000.0000"}

aRWGabsStr = MsgBox.ChoiceAsString(ListofGAbst,
      "für die RW-Gitterlinien",
      "Auswahl eines Abstandes")
aHWGabsStr = MsgBox.ChoiceAsString(ListofGAbst,
      "für die HW-Gitterlinien",
      "Auswahl eines Abstandes")

'Veränderungsmöglichkeit
RWGabsStr = MsgBox.Input("die RW-Gitterlinien (einen anderen Wert?)",
      "Veränderungsmöglichkeit des Gitter-Abstandes", aRWGabsStr)
HWGabsStr = MsgBox.Input("die HW-Gitterlinien (einen anderen Wert?)",
      "Veränderungsmöglichkeit des Gitter-Abstandes", aHWGabsStr)

RWGabs = RWGabsStr.AsNumber
HWGabs = HWGabsStr.AsNumber

'Bestimmung der Gitterlinien
ListofGPL={}
ListofGZahl={}
'RW-Gitterlinien für eine Karte
xpt=kRWGt

while (xpt < gRW )
   ListofPL1={}
   ListofPL1.Add(xpt@kHW)
   ListofPL1.Add(xpt@gHW)
   ListofGPL.Add(ListofPL1)
   ListofGZahl.Add(xpt)
   xpt=xpt+RWGabs
end

'HW-Gitterlinien für eine Karte
ypt=kHWGt

while (ypt < gHW )
   ListofPL1={}
   ListofPL1.Add(kRW@ypt)
   ListofPL1.Add(gRW@ypt)
   ListofGPL.Add(ListofPL1)
   ListofGZahl.Add(ypt)
   ypt=ypt+HWGabs
end

AnzGtL = ListofGPL.Count
IdxGtL = (AnzGtL - 1).SetFormat("").SetFormat("d")

'Ein Feature-Shape-File für Gitterlinien (Polyline) wird hergestellt.
aWDStr=av.GetProject.GetWorkDir.AsString
fnStr=FileName.Make(aWDStr).MakeTmp("Gittktg1","shp")
fName=FileDialog.Put(fnStr, "*.shp",
      "Output Shape File (Gitterlinien der Karte)")
```

```
if (fName=nil) then exit end
fName.SetExtension("shp")
GtFTab=FTab.MakeNew(fName, Polyline)
GtIDFld=Field.Make("ID", #FIELD_SHORT, 4, 0)
GtKdFld=Field.Make("RW, HW", #FIELD_Float, 8, 0)
GtFTab.AddFields({GtIDFld, GtKdFld})
GtShpFld=GtFTab.FindField("shape")

GtFTab.SetEditable(false)
GtFTab.SetEditable(true)
for each rec in 0..IdxGtL
  ListofListofaGtL = {}
  ListofaGtL = ListofGPL.Get(rec)
  ListofListofaGtL.Add(ListofaGtL)
  aPolyL=PolyLine.Make(ListofListofaGtL)
  aZahl = ListofGZahl.Get(rec).SetFormat("").SetFormat("d")
  GtFTab.AddRecord
  GtFTab.SetValue(GtShpFld, rec, aPolyL)
  GtFTab.SetValue(GtIDFld, rec, rec)
  GtFTab.SetValue(GtKdFld, rec, aZahl)
end
GtFTab.SetEditable(false)

thmGtNew=FTheme.Make(GtFTab)
theView.AddTheme(thmGtNew)
```

```
'gittz2t1.ave
'3D-Gitterlinien für ein Schichtenmodell (Punkte) in einem horizontalen
'Abstand von 2000 m und in einem Höhen-Abstand von 20 m werden
'mit Zwischengitterlinien für das ganze Gebiet als ein Fthema hergestellt.
'Ein Menü oder eine Schaltfläche in einer aktiven Szene zum Anklicken.

theProject=av.GetProject
theSzene = av.GetActiveDoc
myScript=theProject.FindScript("gittz2t1")
myScript.SetNumberFormat( "d.dd") ' script default
ListofThms=theSzene.GetThemes
ListofFThms = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
       ListofFThms.Add(aT)
     end
   end
end
av.ShowMsg("Einagebe eines Punkt-Themas und"++
  "Feststellung der Anzahl der Punkte in dem Thema ...")
PtTheme=MsgBox.ChoiceAsString(ListofFThms,
     "um in einer 3D-Szene ein 3D-Gitter herzustellen",
     "Auswahl eines Punkt-Themas (Schichtenmodell)")
PtFTab=PtTheme.GetFTab
ListofFlds=PtFTab.GetFields
PtShpFld=PtFTab.FindField("Shape")
PtOkHFld=MsgBox.ListAsString(ListofFlds, "für Höhen der Oberkante",
      "Auswahl eines Feldes im Thema"++PtTheme.GetName)
PtUkHFld=MsgBox.ListAsString(ListofFlds, "für Höhen der Unterkante",
      "Auswahl eines Feldes im Thema"++PtTheme.GetName)
```

```
AnzPt=0
for each rec in PtFTab
   AnzPt=AnzPt+1
end
PtIdx=AnzPt-1
av.ShowMsg("Die kleinsten und größten Koordinaten und Höhen"
       ++"des Gitters werden berechnet ...")
av.ShowStopButton
minRW=2585000
maxRW=2550000
minHW=5645000
maxHW=5610000
minH=5000
maxH=-1000
for each i in 0..PtIdx
  Ng=i+1
  aPt=PtFTab.ReturnValue(PtShpFld, i)
  aRW=aPt.Getx
  aHW=aPt.Gety
  aOkH=PtFTab.ReturnValue(PtOkHFld, i)
  aUkH=PtFTab.ReturnValue(PtUkHFld, i)
  if (aRW < minRW) then
    minRW=aRW
  end
  if (aRW > maxRW) then
    maxRW=aRW
  end
  if (aHW < minHW) then
    minHW=aHW
  end
  if (aHW > maxHW) then
    maxHW=aHW
  end
  if (aUkH < minH) then
    minH=aUkH
  end
  if (aOkH > maxH) then
    maxH=aOkH
  end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzPt*100)
  if (Not more) then
    break
  end
end
minRWStr=MsgBox.Input("der kleinste RW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", minRW.AsString)
maxRWStr=MsgBox.Input("der größte RW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", maxRW.AsString)
minHWStr=MsgBox.Input("der kleinste HW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", minHW.AsString)
maxHWStr=MsgBox.Input("der größte HW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", maxHW.AsString)
minHStr=MsgBox.Input("die kleinste Höhe [m] des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", minH.AsString)
maxHStr=MsgBox.Input("die größte Höhe [m] des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", maxH.AsString)
minRW=minRWStr.AsNumber
maxRW=maxRWStr.AsNumber
minHW=minHWStr.AsNumber
maxHW=maxHWStr.AsNumber
minH=minHStr.AsNumber
```

```
maxH=maxHStr.AsNumber
ListofRW={minRW, maxRW}
ListofHW={minHW, maxHW}
ListofH={minH, maxH}
ListofListofPLZ={}  'Gitterlinien für das ganze Gebiet
ListofFL={}
for each j in 0..1
  theH=ListofH.Get(j)
  for each i in 0..1
    theHW=ListofHW.Get(i)
    ListofPLZ={}
    ListofPLZ.Add(minRW@theHW@theH)
    ListofPLZ.Add(maxRW@theHW@theH)
    ListofListofPLZ.Add(ListofPLZ)
    ListofFL.Add("G")
  end
end
for each j in 0..1
  theH=ListofH.Get(j)
  for each i in 0..1
    theRW=ListofRW.Get(i)
    ListofPLZ={}
    ListofPLZ.Add(theRW@minHW@theH)
    ListofPLZ.Add(theRW@maxHW@theH)
    ListofListofPLZ.Add(ListofPLZ)
    ListofFL.Add("G")
  end
end

for each j in 0..1
  theHW=ListofHW.Get(j)
  for each i in 0..1
    theRW=ListofRW.Get(i)
    ListofPLZ={}
    ListofPLZ.Add(theRW@theHW@minH)
    ListofPLZ.Add(theRW@theHW@maxH)
    ListofListofPLZ.Add(ListofPLZ)
    ListofFL.Add("G")
  end
end
Intv=2000.00 'der Gitterabstand für RW und HW
HIntv=20.00  'der Gitterabstand für Höhe
'HW-Gitter-Linien
Krt11=((minHW/1000) mod 2)
if (Krt11 = 0) then
  yPt=minHW
elseif (Krt11 <> 0) then
  Krt22=((minHW/1000).Ceiling)
  Krt33=(Krt22 mod 2)
  if (Krt33 = 0) then
    yPt=Krt22*1000
  elseif (Krt33 <> 0) then
    yPt=(Krt22+1)*1000
  end
end
AnfY=yPt
for each i in 0..1
  theH=ListofH.Get(i)
  while (ypt < maxHW)
    ListofPLZ={}
    ListofPLZ.Add(minRW@ypt@theH)
    ListofPLZ.Add(maxRW@ypt@theH)
```

```
      ListofListofPLZ.Add(ListofPLZ)
      if (i = 0) then
        ListofFL.Add("B")
      elseif (i = 1) then
        ListofFL.Add("T")
      end
      ypt=ypt+Intv
    end
    ypt=AnfY
  end
  for each j in 0..1
    theRW=ListofRW.Get(j)
    while (ypt < maxHW)
      ListofPLZ={}
      for each i in 0..1
        theH=ListofH.Get(i)
        ListofPLZ.Add(theRW@ypt@theH)
      end
      ListofListofPLZ.Add(ListofPLZ)
      if (j = 0) then
        ListofFL.Add("W")
      elseif (j = 1) then
        ListofFL.Add("O")
      end
      ypt=ypt+Intv
    end
    ypt=AnfY
  end
  Krt11=((minRW/1000) mod 2) 'RW-Gitter-Linien
  if (Krt11 = 0) then
    xPt=minRW
  elseif (Krt11 <> 0) then
    Krt22=((minRW/1000).Ceiling)
    Krt33=(Krt22 mod 2)
    if (Krt33 = 0) then
      xPt=Krt22*1000
    elseif (Krt33 <> 0) then
      xPt=(Krt22+1)*1000
    end
  end
  AnfX=xPt
  for each i in 0..1
    theH=ListofH.Get(i)
    while (xpt < maxRW )
      ListofPLZ={}
      ListofPLZ.Add(xpt@minHW@theH)
      ListofPLZ.Add(xpt@maxHW@theH)
      ListofListofPLZ.Add(ListofPLZ)
      if (i = 0) then
        ListofFL.Add("B")
      elseif (i = 1) then
        ListofFL.Add("T")
      end
      xpt=xpt+Intv
    end
    xPt=AnfX
  end
  for each j in 0..1
    theHW=ListofHW.Get(j)
    while (xpt < maxRW)
      ListofPLZ={}
      for each i in 0..1
```

```
        theH=ListofH.Get(i)
        ListofPLZ.Add(xPt@theHW@theH)
      end
      ListofListofPLZ.Add(ListofPLZ)
      if (j = 0) then
        ListofFL.Add("S")
      elseif (j = 1) then
        ListofFL.Add("N")
      end
      xpt=xpt+Intv
    end
    xPt=AnfX
  end
  if (minH = 0) then  'Höhen-Gitter-Linien
    theH=minH
  elseif (minH <> 0) then
    Krt11=((minH/10) mod 2)
    if (Krt11 = 0) then
      theH=minH
    elseif (Krt11 <> 0) then
      Krt22=((minH/10).Ceiling)
      Krt33=(Krt22 mod 2)
      if (Krt33 = 0) then
        theH=Krt22*10
      elseif (Krt33 <> 0) then
        theH=(Krt22+1)*10
      end
    end
  end
  AnfH=theH
  for each j in 0..1
    theHW=ListofHW.Get(j)
    while (theH < maxH)
      ListofPLZ={}
      ListofPLZ.Add(minRW@theHW@theH)
      ListofPLZ.Add(maxRW@theHW@theH)
      ListofListofPLZ.Add(ListofPLZ)
      if (j = 0) then
        ListofFL.Add("S")
      elseif (j = 1) then
        ListofFL.Add("N")
      end
      theH=theH+20
    end
    theH=AnfH
  end
  for each j in 0..1
    theRW=ListofRW.Get(j)
    while (theH < maxH)
      ListofPLZ={}
      ListofPLZ.Add(theRW@minHW@theH)
      ListofPLZ.Add(theRW@maxHW@theH)
      ListofListofPLZ.Add(ListofPLZ)
      if (j = 0) then
        ListofFL.Add("W")
      elseif (j = 1) then
        ListofFL.Add("O")
      end
      theH=theH+20
    end
    theH=AnfH
  end
```

```
av.ShowMsg("Speicherung der Gitterlinien in einem Thema ...")
aWDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(aWDStr).MakeTmp("Gitter2t","shp")
fName=FileDialog.Put(fnStr, "*.shp","Output Shape File (3D Gitter)")
if (fName=nil) then exit end
fName.SetExtension("shp")
GtFTab=FTab.MakeNew(fName, PolylineZ)
GtIDFld=Field.Make("ID", #FIELD_SHORT, 4, 0)
GtFLFld=Field.Make("Flag", #FIELD_Char, 2, 0)
GtFTab.AddFields({GtIDFld, GtFLFld})
GtShpFld=GtFTab.FindField("shape")
AnzPLZ=ListofListofPLZ.Count
IdxPLZ=AnzPLZ-1
GtFTab.SetEditable(false)
GtFTab.SetEditable(true)
for each i in 0..IdxPLZ
  theListofPLZ=ListofListofPLZ.Get(i)
  aListofListofPLZ={}
  aListofListofPLZ.Add(theListofPLZ)
  thePolyLZ=PolyLineZ.Make(aListofListofPLZ)
  theFL=ListofFL.Get(i)
  GtFTab.AddRecord
  GtFTab.SetValue(GtShpFld, i, thePolyLZ)
  GtFTab.SetValue(GtIDFld, i, i)
  GtFTab.SetValue(GtFLFld, i, theFL)
end
GtFTab.SetEditable(false)
thmNew=FTheme.Make(GtFTab)
theSzene.AddTheme(thmNew)
```

```
'gittz2tr.ave
'3D-Randlinien werden für ein Gebiet eines Schichtenmodells
'als ein FThema hergestellt.
'Ein Menü oder eine Schaltfläche in einer aktiven Szene zum Anklicken.

theProject=av.GetProject
theSzene=av.GetActiveDoc
myScript=theProject.FindScript("gittz2tr")
myScript.SetNumberFormat( "d.dd") ' script default

ListofThms=theSzene.GetThemes
ListofFThms = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofFThms.Add(aT)
      end
   end
end
av.ShowMsg("Einagebe eines Punkt-Themas und"++
  "Feststellung der Anzahl der Punkte in dem Thema ...")
PtTheme=MsgBox.ChoiceAsString(ListofFThms,
      "um in einer 3D-Szene 3D-Ränder herzustellen",
      "Auswahl eines Punkt-Themas (Schichtenmodell)")
PtFTab=PtTheme.GetFTab
ListofFlds=PtFTab.GetFields
```

```
PtShpFld=PtFTab.FindField("Shape")
PtOkHFld=MsgBox.ListAsString(ListofFlds, "für Höhen der Oberkante",
       "Auswahl eines Feldes im Thema"++PtTheme.GetName)
PtUkHFld=MsgBox.ListAsString(ListofFlds, "für Höhen der Unterkante",
       "Auswahl eines Feldes im Thema"++PtTheme.GetName)
AnzPt=0
for each rec in PtFTab
   AnzPt=AnzPt+1
end
PtIdx=AnzPt-1
AI = AnzPt.SetFormat("").SetFormat("d").AsString
MsgBox.Info(AI,"Die Anzahl der Punkte im Thema"++PtTheme.AsString)
av.ShowMsg("Die kleinsten und größten Koordinaten und Höhen"
        ++"der Ränder werden berechnet ...")
av.ShowStopButton
minRW=2585000
maxRW=2550000
minHW=5645000
maxHW=5610000
minH=5000
maxH=-1000
for each i in 0..PtIdx
  Ng=i+1
  aPt=PtFTab.ReturnValue(PtShpFld, i)
  aRW=aPt.Getx
  aHW=aPt.Gety
  aOkH=PtFTab.ReturnValue(PtOkHFld, i)
  aUkH=PtFTab.ReturnValue(PtUkHFld, i)
  if (aRW < minRW) then
    minRW=aRW
  end
  if (aRW > maxRW) then
    maxRW=aRW
  end
  if (aHW < minHW) then
    minHW=aHW
  end
  if (aHW > maxHW) then
    maxHW=aHW
  end
  if (aUkH < minH) then
    minH=aUkH
  end
  if (aOkH > maxH) then
    maxH=aOkH
  end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzPt*100)
  if (Not more) then
    break
  end
end
minRWStr=MsgBox.Input("der kleinste RW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", minRW.AsString)
maxRWStr=MsgBox.Input("der größte RW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", maxRW.AsString)
minHWStr=MsgBox.Input("der kleinste HW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", minHW.AsString)
maxHWStr=MsgBox.Input("der größte HW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", maxHW.AsString)
minHStr=MsgBox.Input("die kleinste Höhe [m] des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", minH.AsString)
```

```
maxHStr=MsgBox.Input("die größte Höhe [m] des Gitters",
  "Veränderungsmöglichkeit der Koordinaten", maxH.AsString)
minRW=minRWStr.AsNumber
maxRW=maxRWStr.AsNumber
minHW=minHWStr.AsNumber
maxHW=maxHWStr.AsNumber
minH=minHStr.AsNumber
maxH=maxHStr.AsNumber
ListofRW={minRW, maxRW}
ListofHW={minHW, maxHW}
ListofH={minH, maxH}

ListofListofPLZ={}     'Gitterlinien für das ganze Gebiet
ListofFL={}
for each j in 0..1
  theHW=ListofHW.Get(j)
  for each i in 0..1
    theH=ListofH.Get(i)
    ListofPLZ={}
    ListofPLZ.Add(minRW@theHW@theH)
    ListofPLZ.Add(maxRW@theHW@theH)
    ListofListofPLZ.Add(ListofPLZ)
    if (j = 0) then
      ListofFL.Add("S")
    elseif (j = 1) then
      ListofFL.Add("N")
    end
  end
end

for each j in 0..1
  theRW=ListofRW.Get(j)
  for each i in 0..1
    theH=ListofH.Get(i)
    ListofPLZ={}
    ListofPLZ.Add(theRW@minHW@theH)
    ListofPLZ.Add(theRW@maxHW@theH)
    ListofListofPLZ.Add(ListofPLZ)
    if (j = 0) then
      ListofFL.Add("W")
    elseif (j = 1) then
      ListofFL.Add("O")
    end
  end
end

for each j in 0..1
  theHW=ListofHW.Get(j)
  for each i in 0..1
    theRW=ListofRW.Get(i)
    ListofPLZ={}
    ListofPLZ.Add(theRW@theHW@minH)
    ListofPLZ.Add(theRW@theHW@maxH)
    ListofListofPLZ.Add(ListofPLZ)
    if (j = 0) then
      ListofFL.Add("S")
    elseif (j = 1) then
      ListofFL.Add("N")
    end
  end
end
av.ShowMsg("Speicherung der Gitterlinien in einem Thema ...")
```

```
aWDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(aWDStr).MakeTmp("Gitter2t","shp")
fName=FileDialog.Put(fnStr, "*.shp","Output Shape File (3D Gitter)")
if (fName=nil) then exit end
fName.SetExtension("shp")
GtFTab=FTab.MakeNew(fName, PolylineZ)
GtIDFld=Field.Make("ID", #FIELD_SHORT, 4, 0)
GtFLFld=Field.Make("Flag", #FIELD_Char, 2, 0)
GtFTab.AddFields({GtIDFld, GtFLFld})
GtShpFld=GtFTab.FindField("shape")

AnzPLZ=ListofListofPLZ.Count
IdxPLZ=AnzPLZ-1
GtFTab.SetEditable(false)
GtFTab.SetEditable(true)
for each i in 0..IdxPLZ
  theListofPLZ=ListofListofPLZ.Get(i)
  aListofListofPLZ={}
  aListofListofPLZ.Add(theListofPLZ)
  thePolyLZ=PolyLineZ.Make(aListofListofPLZ)
  theFL=ListofFL.Get(i)
  GtFTab.AddRecord
  GtFTab.SetValue(GtShpFld, i, thePolyLZ)
  GtFTab.SetValue(GtIDFld, i, i)
  GtFTab.SetValue(GtFLFld, i, theFL)
end
GtFTab.SetEditable(false)
thmNew=FTheme.Make(GtFTab)
theSzene.AddTheme(thmNew)
```

```
'gittzof1.ave
'3D-Gitterlinien in einem horizontalen Abstand von 2000 m
'werden für eine Schicht-Oberfläche hergestellt.
'Ein Menü oder eine Schaltfläche in einer aktiven Szene zum Anklicken.
theProject=av.GetProject
theSzene=av.GetActiveDoc
myScript=theProject.FindScript("gittzof1")
myScript.SetNumberFormat( "d.dd") ' script default
ListofThms=theSzene.GetThemes
ListofFThms = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
       ListofFThms.Add(aT)
     end
   end
end
av.ShowMsg("Einagebe eines Punkt-Themas und"++
  "Feststellung der Anzahl der Punkte in dem Thema ...")
PtTheme=MsgBox.ChoiceAsString(ListofFThms,
     "um in einer Szene ein 3D-Gitter herzustellen",
     "Auswahl eines Punkt-Themas (Schichtenmodell)")
t=PtTheme
PtFTab=PtTheme.GetFTab
ListofFlds=PtFTab.GetFields
PtShpFld=PtFTab.FindField("Shape")
PtOkHFld=MsgBox.ListAsString(ListofFlds, "für Höhen der Oberkane",
```

```
      "Auswahl eines Feldes im Thema"++PtTheme.GetName)
PtUkHFld=MsgBox.ListAsString(ListofFlds, "für Höhen der Unterkante",
      "Auswahl eines Feldes im Thema"++PtTheme.GetName)
AnzPt=0
for each rec in PtFTab
   AnzPt=AnzPt+1
end
PtIdx=AnzPt-1
av.ShowMsg("Die kleinsten und größten Koordinaten und Höhen"
       ++"des Gitters werden berechnet ...")
av.ShowStopButton
minRW=2585000
maxRW=2550000
minHW=5645000
maxHW=5610000
minH=5000
maxH=-1000
for each i in 0..PtIdx
  Ng=i+1
  aPt=PtFTab.ReturnValue(PtShpFld, i)
  aRW=aPt.Getx
  aHW=aPt.Gety
  aOkH=PtFTab.ReturnValue(PtOkHFld, i)
  aUkH=PtFTab.ReturnValue(PtUkHFld, i)
  if (aRW < minRW) then
    minRW=aRW
  end
  if (aRW > maxRW) then
    maxRW=aRW
  end
  if (aHW < minHW) then
    minHW=aHW
  end
  if (aHW > maxHW) then
    maxHW=aHW
  end
  if (aUkH < minH) then
    minH=aUkH
  end
  if (aOkH > maxH) then
    maxH=aOkH
  end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzPt*100)
  if (Not more) then
    break
  end
end
minRWStr=MsgBox.Input("der kleinste RW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", minRW.AsString)
maxRWStr=MsgBox.Input("der größte RW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", maxRW.AsString)
minHWStr=MsgBox.Input("der kleinste HW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", minHW.AsString)
maxHWStr=MsgBox.Input("der größte HW des Gitters",
 "Veränderungsmöglichkeit der Koordinaten", maxHW.AsString)
minRW=minRWStr.AsNumber
maxRW=maxRWStr.AsNumber
minHW=minHWStr.AsNumber
maxHW=maxHWStr.AsNumber
ListofRW={minRW, maxRW}
ListofHW={minHW, maxHW}
```

```
ListofListofPt={}    'Gitterlinien für das ganze Gebiet
Intv=2000.00 'der Gitterabstand für RW und HW
AnzRW=((maxRW-minRW)/50)+1
IdxRW=AnzRW-1
for each j in 0..1
  theHW=ListofHW.Get(j)
  ListofPt={}
  for each i in 0..IdxRW
    theRW=minRW+(i*50)
    ListofPt.Add(theRW@theHW)
  end
  ListofListofPt.Add(ListofPt)
end
AnzHW=((maxHW-minHW)/50)+1
IdxHW=AnzHW-1
for each j in 0..1
  theRW=ListofRW.Get(j)
  ListofPt={}
  for each i in 0..IdxHW
    theHW=minHW+(i*50)
    ListofPt.Add(theRW@theHW)
  end
  ListofListofPt.Add(ListofPt)
end
Krt11=((minHW/1000) mod 2) 'HW-Gitter-Linien
if (Krt11 = 0) then
  yPt=minHW
elseif (Krt11 <> 0) then
  Krt22=((minHW/1000).Ceiling)
  Krt33=(Krt22 mod 2)
  if (Krt33 = 0) then
    yPt=Krt22*1000
  elseif (Krt33 <> 0) then
    yPt=(Krt22+1)*1000
  end
end
while (ypt < maxHW)
  ListofPt={}
  for each i in 0..IdxRW
    theRW=minRW+(i*50)
    ListofPt.Add(theRW@ypt)
  end
  ListofListofPt.Add(ListofPt)
  ypt=ypt+Intv
end
Krt11=((minRW/1000) mod 2) 'RW-Gitter-Linien
if (Krt11 = 0) then
  xPt=minRW
elseif (Krt11 <> 0) then
  Krt22=((minRW/1000).Ceiling)
  Krt33=(Krt22 mod 2)
  if (Krt33 = 0) then
    xPt=Krt22*1000
  elseif (Krt33 <> 0) then
    xPt=(Krt22+1)*1000
  end
end
while (xpt < maxRW )
  ListofPt={}
  for each i in 0..IdxHW
    theHW=minHW+(i*50)
    ListofPt.Add(xpt@theHW)
```

```
    end
    ListofListofPt.Add(ListofPt)
    xpt=xpt+Intv
  end
  done=False      'Eingabe einer interpolierten Datei mit Höhenwerten (Grid oder Tin)
  While (not done)
   surfaceList = {}
   for each t2 in theSzene.GetThemes
     if (t2.Is(GTheme) or t2.Is(STheme)) then
       surfaceList.Add(t2)
     end
   end
   if (surfaceList.Count = 0) then
     aSurfFN = SourceManager.GetDataSet({Grid,Tin},"Select Surface :" ++ t.GetName)
     if (aSurfFN = NIL) then
       continue
     end
     aSrcName = Grid.MakeSrcName(aSurfFN.AsString)
     if (aSrcName <> NIL) then
       theSurface = Grid.Make(aSrcName)
       surfTheme = GTheme.Make(theSurface)
     else
       aSrcName = SrcName.Make(aSurfFN.AsString)
       theSurface = Tin.Make(aSrcName)
       surfTheme = STheme.Make(theSurface)
     end
     theSzene.AddTheme(surfTheme)
   else
     surfTheme = MsgBox.ListAsString(surfaceList,"Choose theme to use as surface:",
             "Select Surface :" ++ t.GetName)
     if (surfTheme = NIL) then
       continue
     end
     if (surfTheme.Is(GTheme)) then
       theSurface = surfTheme.GetGrid
     elseif (surfTheme.Is(STheme)) then
       theSurface = surfTheme.GetSurface
     else
       continue
     end
   end
   done=True
  end
  av.ShowMsg("Zuweisung der Z Werten zu den Punkten ...")
  av.ShowStopButton
  AnzPL=ListofListofPt.Count
  IdxPL=AnzPL-1
  aPrj=Prj.MakeNull
  ListofListofPtZ={}
  Ng=0
  for each aListofPt in ListofListofPt
    Ng=Ng+1
    ListofPtZ={}
    for each aPt in aListofPt
      if (surfTheme.Is(GTheme)) then
        theZValue=theSurface.PointValue(aPt, aPrj)
      elseif (surfTheme.Is(STheme)) then
        theZValue=theSurface.Elevation(aPt)
      end
      ListofPtZ.Add(aPt@theZValue)
    end
    ListofListofPtZ.Add(ListofPtZ)
```

```
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzPL*100)
  if (not more) then
    break
  end
end
av.ShowMsg("Speicherung der Gitterlinien in einem Thema ...")
aWDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(aWDStr).MakeTmp("Gitter2t","shp")
fName=FileDialog.Put(fnStr, "*.shp","Output Shape File (3D-Gitter)")
if (fName=nil) then exit end
fName.SetExtension("shp")
GtFTab=FTab.MakeNew(fName, PolylineZ)
GtIDFld=Field.Make("ID", #FIELD_SHORT, 4, 0)
GtFTab.AddFields({GtIDFld})
GtShpFld=GtFTab.FindField("shape")
AnzPLZ=ListofListofPtZ.Count
IdxPLZ=AnzPLZ-1
GtFTab.SetEditable(false)
GtFTab.SetEditable(true)
for each i in 0..IdxPLZ
  theListofPLZ=ListofListofPtZ.Get(i)
  aListofListofPLZ={}
  aListofListofPLZ.Add(theListofPLZ)
  thePolyLZ=PolyLineZ.Make(aListofListofPLZ)
  GtFTab.AddRecord
  GtFTab.SetValue(GtShpFld, i, thePolyLZ)
  GtFTab.SetValue(GtIDFld, i, i)
end
GtFTab.SetEditable(false)
thmNew=FTheme.Make(GtFTab)
theSzene.AddTheme(thmNew)
```

```
'gittzsn2.ave
'3D-Gitterlinien (PolylineZ) werden durch ein Polygon ausgeschnitten
'und in einem Thema für Gitterlinien gespeichert.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("gittzsn2")
myScript.SetNumberFormat("d.dd")
ListofThms=theView.GetThemes
ListofPLThm={}
ListofPgThm = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(PolylineZ)) then
       ListofPLThm.Add(aT)
     elseif (aFTab.GetShapeClass.IsSubclassOf(PolygonZ)) then
       ListofPgThm.Add(aT)
     end
   end
end
av.ShowMsg("Eingabe des PolylineZ-Themas und"
     ++"Feststellung der Anzahl der Datensätze im Thema ...")
PLThm=MsgBox.ChoiceAsString(ListofPLThm,
```

```
       "das die PolyLine für Gitter enthält,",
       "Eingabe eines Themas im View"++theView.AsString)
PLFTab=PLThm.GetFTab
PLShpFld=PLFTab.FindField("Shape")
ListofPLFlds = PLFTab.GetFields
PLFlagFld=MsgBox.ListAsString(ListofPLFlds, "für eine Kennzeichnung (Flag)",
        "Auswahl eines Feldes im Thema"++PLThm.AsString)
'Feststellung der Anzahl der PolyLine in dem Thema

AnzPL=0
for each rec in PLFTab
   AnzPL=AnzPL+1
end
IdxPL=AnzPL-1
AnzPLStr = AnzPL.SetFormat("").SetFormat("d").AsString
MsgBox.Info(AnzPLStr, "Anzahl der PolyLine im Thema"++PLThm.AsString)

'Eingabe eines PolygonZ-Themas, um die Gitterlinien abzuschneiden
PgThm=MsgBox.ChoiceAsString(ListofPgThm, "das die 3D-Polygone enthält",
       "Eingabe eines Themas im View"++theView.AsString)
PgFTab=PgThm.GetFTab
PgShpFld=PgFTab.FindField("Shape")
ListofPgFlds = PgFTab.GetFields
PgNameFld=MsgBox.ListAsString(ListofPgFlds,
        "für eine Kennzeichnung des Polygons",
        "Auswahl eines Feldes im Thema"++PgThm.AsString)
AnzPgZ=0
for each rec in PgFTab
  AnzPgZ=AnzPgZ+1
end
IdxPgZ=AnzPgZ-1

'Feststellung der Datensätze im PolygonZ-Thema
AnzPgFld=ListofPgFlds.Count
IdxPgFld=AnzPgFld-1
ListofDtStr={}

for each j in 0..IdxPgZ
  DtStr=""
  for each i in 0..IdxPgFld
    aFld=ListofPgFlds.Get(i)
    aFldStr=aFld.GetName
    if (aFldStr <> "Shape") then
      aValue=PgFTab.ReturnValue(aFld, j)
      DtStr=DtStr+aValue.AsString+"; "
    end
  end
  ListofDtStr.Add(DtStr)
end
'Auswahl eines Polygons
ListofPgFL = {"N", "S", "O", "W", "T", "B"}
aDtS = MsgBox.ListAsString(ListofDtStr,
    "um die Gitterlinien abzuschneiden",
    "Auswahl eines Polygons im Thema"++PgThm.AsString)
aDtSIdx = ListofDtStr.FindByValue(aDtS)
the3dPg=PgFTab.ReturnValue(PgShpFld, aDtSIdx)
Nr = aDtSIdx
AnzPgZStr = AnzPgZ.SetFormat("").SetFormat("d").AsString
theFL = MsgBox.ListAsString(ListofPgFL, "für den"
       ++Nr.SetFormat("d").AsString+"."++"Datensatz"+NL+
       "("+aDts+")",
       "Auswahl eines Kennwortes von"++AnzPgZStr++"Pg")
```

```
minRW = 0.00
maxRW = 0.00
minHW = 0.00
maxHW = 0.00
minZ = 0.00
maxZ = 200.00

Listof2DPgPt={}
ListofListofPgPt=the3DPg.AsList
for each aListofPgPt in ListofListofPgPt
  for each aPgPt in aListofPgPt
    aX=aPgPt.Getx
    aY=aPgPt.Gety
    aZ=aPgPt.Getz
    if (theFL = "S") then
      Listof2DPgPt.Add(aX@aZ)
      minHW = aY
    elseif (theFL = "N") then
      Listof2DPgPt.Add(aX@aZ)
      maxHW = aY
    elseif (theFL = "W") then
      Listof2DPgPt.Add(aY@aZ)
      minRW = aX
    elseif (theFL = "O") then
      Listof2DPgPt.Add(aY@aZ)
      maxRW = aX
    elseif (theFL = "B") then
      Listof2DPgPt.Add(aX@aY)
      minZ = aZ
    elseif (theFL = "T") then
      Listof2DPgPt.Add(aX@aY)
      maxZ = aZ
    end
  end
end
ListofListof2DPgPt={}
ListofListof2DPgPt.Add(Listof2DPgPt)
the2DPg=Polygon.Make(ListofListof2DPgPt)

ListoffestW = {minHW,maxHW,minRW,maxRW,minZ,maxZ}

av.ShowMsg("Die Gitterlinien werden durch Polygon ausgeschnitten ...")
av.ShowStopButton
Listof3DGL={}
Listof3DFL={}

ListofagwPL = {}  'Auswahl der PolyLineZ
for each a3PL in 0..IdxPL
  the3PL = PLFTab.ReturnValue(PLShpFld, a3PL)
  if (the3PL.Intersects(the3dPg)) then
    ListofagwPL.Add(the3PL)
  end
end
AnzagwPL = ListofagwPL.Count
IdxagwPL = AnzagwPL - 1
Listof2DGL = {}  'Bildung der 2D-PolyLine
for each agw in 0..IdxagwPL
   agwPL = ListofagwPL.Get(agw)
   Listof2DGLPt={}
   ListofListofGLPt=agwPL.AsList
   for each aListofGLPt in ListofListofGLPt
```

```
      for each aGLPt in aListofGLPt
        aX=aGLPt.Getx
        aY=aGLPt.Gety
        aZ=aGLPt.Getz
        if (theFL = "S") then
          Listof2DGLPt.Add(aX@aZ)
        elseif (theFL = "N") then
          Listof2DGLPt.Add(aX@aZ)
        elseif (theFL = "W") then
          Listof2DGLPt.Add(aY@aZ)
        elseif (theFL = "O") then
          Listof2DGLPt.Add(aY@aZ)
        elseif (theFL = "B") then
          Listof2DGLPt.Add(aX@aY)
        elseif (theFL = "T") then
          Listof2DGLPt.Add(aX@aY)
        end
      end
    end
    ListofListof2DGLPt={}
    ListofListof2DGLPt.Add(Listof2DGLPt)
    the2DGL=Polyline.Make(ListofListof2DGLPt)
    Listof2DGL.Add(the2DGL)
end

AnzGL=Listof2DGL.Count
IdxGL=AnzGL-1

for each i in 0..IdxGL 'Schneiden der PolyLine
  Ng=i+1
  thePLZ=Listof2DGL.Get(i)

  ListofNewPolyL={}
  if (thePLZ.Intersects(the2DPg)) then
    NewPolyL=thePLZ.LineIntersection(the2DPg)
    ListofCheck1=NewPolyL.AsList
    Check1=ListofCheck1.Count
    if (Check1 <> 0) then
      ListofTeile=NewPolyL.Explode
      AnzderTeile=ListofTeile.Count
      AnzdTIdx=AnzderTeile-1
      for each j in 0..AnzdTIdx
        NewPolyLT=ListofTeile.Get(j)
        ListofNewPolyL.Add(NewPolyLT)
      end
    end
  end
  AnzNewPL=ListofNewPolyL.Count
  if (AnzNewPL <> 0) then
    IdxNewPL=AnzNewPL-1
    for each j in 0..IdxNewPL
      aNPL=ListofNewPolyL.Get(j)
      aNListofListofPt=aNPL.AsList
      Listof3DPt={}
      for each aNListofPt in aNListofListofPt
        for each aNPt in aNListofPt
          aX=aNPt.Getx
          aY=aNPt.Gety
          if (theFL = "S") then
            minHW = ListoffestW.Get(0)
            Listof3DPt.Add(aX@minHW@aY)
          elseif (theFL = "N") then
```

```
            maxHW = ListoffestW.Get(1)
            Listof3DPt.Add(aX@maxHW@aY)
          elseif (theFL = "W") then
            minRW = ListoffestW.Get(2)
            Listof3DPt.Add(minRW@aX@aY)
          elseif (theFL = "O") then
            maxRW = ListoffestW.Get(3)
            Listof3DPt.Add(maxRW@aX@aY)
          elseif (theFL = "B") then
            minZ = ListoffestW.Get(4)
            Listof3DPt.Add(aX@aY@minZ)
          elseif (theFL = "T") then
            maxZ = ListoffestW.Get(5)
            Listof3DPt.Add(aX@aY@maxZ)
          end
        end
      end
      ListofListof3DPt={}
      ListofListof3DPt.Add(Listof3DPt)
      the3DGL=polylineZ.make(ListofListof3DPt)
      Listof3DGL.Add(the3DGL)
      Listof3DFL.Add(theFL)
    end
  end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzGL*100)
  if (not more) then
    break
  end
end


'Ein neues Feature-Shape-File für Gitterlinien (PolyLineZ) wird hergestellt.
aWDStr=theProject.GetWorkDir.AsString
aDfnStr=PLThm.AsString.Left(6)
fnStr=FileName.Make(aWDStr).MakeTmp(aDfnStr,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PolyLineZ)")
if (fName=nil) then exit end
fName.SetExtension("shp")
NPLFTab=FTab.MakeNew(fName, PolyLineZ)
NShpFld=NPLFTab.FindField("shape")
NIDFld=Field.Make("ID", #Field_Short, 4, 0)
NFLFld=Field.Make("Flag", #Field_Char, 2, 0)
ListofFlds1={NIDFld, NFLFld}
NPLFTab.AddFields(ListofFlds1)
NPLFTab.SetEditable(false)
NPLFTab.SetEditable(true)
AnzNPLZ=Listof3DGL.Count
IdxNPLZ=AnzNPLZ-1
for each i in 0..IdxNPLZ
  theNPLZ=Listof3DGL.Get(i)
  theFL=Listof3DFL.Get(i)
  NPLFTab.AddRecord
  NPLFTab.SetValue(NShpFld, i, theNPLZ)
  NPLFTab.SetValue(NIDFld, i, i)
  NPLFTab.SetValue(NFLFld, i, theFL)
end
NPLFTab.SetEditable(false)
thmNew=FTheme.Make(NPLFTab)
theView.AddTheme(thmNew)
```

```
'gtzsn6w.ave
'3D-Gitterlinien (PolylineZ) werden durch Polygone ausgeschnitten
'und in einem Thema für Gitterlinien gespeichert.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("gtzsn6w2")
myScript.SetNumberFormat("d.dd")
ListofThms=theView.GetThemes
ListofPLThm={}
ListofPgThm = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(PolylineZ)) then
       ListofPLThm.Add(aT)
     elseif (aFTab.GetShapeClass.IsSubclassOf(PolygonZ)) then
       ListofPgThm.Add(aT)
     end
   end
end
av.ShowMsg("Eingabe des PolylineZ-Themas und"
     ++"Feststellung der Anzahl der Datensätze im Thema ...")
PLThm=MsgBox.ChoiceAsString(ListofPLThm,
       "das die PolyLine für Gitter enthält,",
       "Eingabe eines Themas im View"++theView.AsString)
PLFTab=PLThm.GetFTab
PLShpFld=PLFTab.FindField("Shape")
ListofPLFlds = PLFTab.GetFields
PLFlagFld=MsgBox.ListAsString(ListofPLFlds, "für eine Kennzeichnung (Flag)",
        "Auswahl eines Feldes im Thema"++PLThm.AsString)
'Feststellung der Anzahl der PolyLine in dem Thema

AnzPL=0
for each rec in PLFTab
   AnzPL=AnzPL+1
end
IdxPL=AnzPL-1
AnzPLStr = AnzPL.SetFormat("").SetFormat("d").AsString
MsgBox.Info(AnzPLStr, "Anzahl der PolyLine im Thema"++PLThm.AsString)

'Eingabe eines PolygonZ-Themas, um die Gitterlinien abzuschneiden
PgThm=MsgBox.ChoiceAsString(ListofPgThm, "das die 3D-Polygone enthält",
       "Eingabe eines Themas im View"++theView.AsString)
PgFTab=PgThm.GetFTab
PgShpFld=PgFTab.FindField("Shape")
ListofPgFlds = PgFTab.GetFields
PgNameFld=MsgBox.ListAsString(ListofPgFlds,
        "für eine Kennzeichnung des Polygons",
        "Auswahl eines Feldes im Thema"++PgThm.AsString)
AnzPgZ=0
for each rec in PgFTab
  AnzPgZ=AnzPgZ+1
end
IdxPgZ=AnzPgZ-1

'Feststellung der Datensätze im PolygonZ-Thema
AnzPgFld=ListofPgFlds.Count
IdxPgFld=AnzPgFld-1
ListofDtStr={}
```

```
for each j in 0..IdxPgZ
  DtStr=""
  for each i in 0..IdxPgFld
    aFld=ListofPgFlds.Get(i)
    aFldStr=aFld.GetName
    if (aFldStr <> "Shape") then
      aValue=PgFTab.ReturnValue(aFld, j)
      DtStr=DtStr+aValue.AsString+"; "
    end
  end
  ListofDtStr.Add(DtStr)
end
ListofPgFL = {"N", "S", "O", "W", "T", "B"}
Listof2DPg={}
Listof2DPgFL={}
minZ = 0.00
maxZ = 200.00
for each i in 0..IdxPgZ
   Nr = i
   the3dPg=PgFTab.ReturnValue(PgShpFld, i)
   aDtS = ListofDtStr.Get(i)
   AnzPgZStr = AnzPgZ.SetFormat("").SetFormat("d").AsString
   theFL = MsgBox.ListAsString(ListofPgFL, "für den"
        ++Nr.SetFormat("d").AsString+"."++"Datensatz"+NL+
        "("+aDts+")",
        "Auswahl eines Kennwortes von"++AnzPgZStr++"Pg")

  Listof2DPgPt={}
  ListofListofPgPt=the3DPg.AsList
  for each aListofPgPt in ListofListofPgPt
    for each aPgPt in aListofPgPt
      aX=aPgPt.Getx
      aY=aPgPt.Gety
      aZ=aPgPt.Getz
      if (theFL = "S") then
        Listof2DPgPt.Add(aX@aZ)
        minHW = aY
      elseif (theFL = "N") then
        Listof2DPgPt.Add(aX@aZ)
        maxHW = aY
      elseif (theFL = "W") then
        Listof2DPgPt.Add(aY@aZ)
        minRW = aX
      elseif (theFL = "O") then
        Listof2DPgPt.Add(aY@aZ)
        maxRW = aX
      elseif (theFL = "B") then
        Listof2DPgPt.Add(aX@aY)
        minZ = aZ
      elseif (theFL = "T") then
        Listof2DPgPt.Add(aX@aY)
        maxZ = aZ
      end
    end
  end
  ListofListof2DPgPt={}
  ListofListof2DPgPt.Add(Listof2DPgPt)
  the2DPg=Polygon.Make(ListofListof2DPgPt)
  Listof2DPg.Add(the2DPg)
  Listof2DPgFL.Add(theFL)
end
```

```
ListoffestW = {minHW,maxHW,minRW,maxRW,minZ,maxZ}

av.ShowMsg("Die Gitterlinien werden durch Polygone ausgeschnitten ...")
av.ShowStopButton
Listof3DGL={}
Listof3DFL={}

for each aPg in 0..IdxPgZ
  aPgZ = PgFTab.ReturnValue(PgShpFld, aPg)
  aPgZFL = Listof2DPgFL.Get(aPg)

  ListofagwPL = {}  'Auswahl der PolyLineZ
  for each a3PL in 0..IdxPL
    the3PL = PLFTab.ReturnValue(PLShpFld, a3PL)
    if (the3PL.Intersects(aPgZ)) then
      ListofagwPL.Add(the3PL)
    end
  end
  AnzagwPL = ListofagwPL.Count
  IdxagwPL = AnzagwPL - 1
  Listof2DGL = {}  'Bildung der 2D-PolyLine
  for each agw in 0..IdxagwPL
     agwPL = ListofagwPL.Get(agw)
     Listof2DGLPt={}
     ListofListofGLPt=agwPL.AsList
     for each aListofGLPt in ListofListofGLPt
       for each aGLPt in aListofGLPt
         aX=aGLPt.Getx
         aY=aGLPt.Gety
         aZ=aGLPt.Getz
         if (aPgZFL = "S") then
            Listof2DGLPt.Add(aX@aZ)
         elseif (aPgZFL = "N") then
            Listof2DGLPt.Add(aX@aZ)
         elseif (aPgZFL = "W") then
            Listof2DGLPt.Add(aY@aZ)
         elseif (aPgZFL = "O") then
            Listof2DGLPt.Add(aY@aZ)
         elseif (aPgZFL = "B") then
            Listof2DGLPt.Add(aX@aY)
         elseif (aPgZFL = "T") then
            Listof2DGLPt.Add(aX@aY)
         end
       end
     end
     ListofListof2DGLPt={}
     ListofListof2DGLPt.Add(Listof2DGLPt)
     the2DGL=Polyline.Make(ListofListof2DGLPt)
     Listof2DGL.Add(the2DGL)
  end

  the2DPg=Listof2DPg.Get(aPg)
  AnzGL=Listof2DGL.Count
  IdxGL=AnzGL-1

  for each i in 0..IdxGL 'Schneiden der PolyLine
    Ng=i+1
    thePLZ=Listof2DGL.Get(i)

    ListofNewPolyL={}
    if (thePLZ.Intersects(the2DPg)) then
      NewPolyL=thePLZ.LineIntersection(the2DPg)
```

```
        ListofCheck1=NewPolyL.AsList
        Check1=ListofCheck1.Count
        if (Check1 <> 0) then
          ListofTeile=NewPolyL.Explode
          AnzderTeile=ListofTeile.Count
          AnzdTIdx=AnzderTeile-1
          for each j in 0..AnzdTIdx
            NewPolyLT=ListofTeile.Get(j)
            ListofNewPolyL.Add(NewPolyLT)
          end
        end
      end
      AnzNewPL=ListofNewPolyL.Count
      if (AnzNewPL <> 0) then
        IdxNewPL=AnzNewPL-1
        for each j in 0..IdxNewPL
          aNPL=ListofNewPolyL.Get(j)
          aNListofListofPt=aNPL.AsList
          Listof3DPt={}
          for each aNListofPt in aNListofListofPt
            for each aNPt in aNListofPt
              aX=aNPt.Getx
              aY=aNPt.Gety
              if (aPgZFL = "S") then
                minHW = ListoffestW.Get(0)
                Listof3DPt.Add(aX@minHW@aY)
              elseif (aPgZFL = "N") then
                maxHW = ListoffestW.Get(1)
                Listof3DPt.Add(aX@maxHW@aY)
              elseif (aPgZFL = "W") then
                minRW = ListoffestW.Get(2)
                Listof3DPt.Add(minRW@aX@aY)
              elseif (aPgZFL = "O") then
                maxRW = ListoffestW.Get(3)
                Listof3DPt.Add(maxRW@aX@aY)
              elseif (aPgZFL = "B") then
                minZ = ListoffestW.Get(4)
                Listof3DPt.Add(aX@aY@minZ)
              elseif (aPgZFL = "T") then
                maxZ = ListoffestW.Get(5)
                Listof3DPt.Add(aX@aY@maxZ)
              end
            end
          end
          ListofListof3DPt={}
          ListofListof3DPt.Add(Listof3DPt)
          the3DGL=polylineZ.make(ListofListof3DPt)
          Listof3DGL.Add(the3DGL)
          Listof3DFL.Add(aPgZFL)
        end
      end
      'Show percentage complete with enabled stop button
      more=av.SetStatus(Ng/AnzGL*100)
      if (not more) then
        break
      end
    end
  end  'Ende von (for each aPg in 0..IdxPgZ)

  'Ein neues Feature-Shape-File für Gitterlinien (PolyLineZ) wird hergestellt.
  aWDStr=theProject.GetWorkDir.AsString
  aDfnStr=PLThm.AsString.Left(6)
```

```
fnStr=FileName.Make(aWDStr).MakeTmp(aDfnStr,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PolyLineZ)")
if (fName=nil) then exit end
fName.SetExtension("shp")
NPLFTab=FTab.MakeNew(fName, PolyLineZ)
NShpFld=NPLFTab.FindField("shape")
NIDFld=Field.Make("ID", #Field_Short, 4, 0)
NFLFld=Field.Make("Flag", #Field_Char, 2, 0)
ListofFlds1={NIDFld, NFLFld}
NPLFTab.AddFields(ListofFlds1)
NPLFTab.SetEditable(false)
NPLFTab.SetEditable(true)
AnzNPLZ=Listof3DGL.Count
IdxNPLZ=AnzNPLZ-1
for each i in 0..IdxNPLZ
  theNPLZ=Listof3DGL.Get(i)
  theFL=Listof3DFL.Get(i)
  NPLFTab.AddRecord
  NPLFTab.SetValue(NShpFld, i, theNPLZ)
  NPLFTab.SetValue(NIDFld, i, i)
  NPLFTab.SetValue(NFLFld, i, theFL)
end
NPLFTab.SetEditable(false)
thmNew=FTheme.Make(NPLFTab)
theView.AddTheme(thmNew)
```

```
'Legendfm.ave
'Alle Arten von Legenden eines Themas oder vieler Themen werden
'in einem aktiven View verändert.
'Dieses Skript wird als ein Menü oder als eine Schaltfläche
'in einem aktiven View zum Anklicken benutzt.

theProject = av.GetProject
theView = av.GetActiveDoc  'ein aktives View
myScript = theProject.FindScript("Legendfm")
myScript.SetNumberFormat("d")  'eine Format der Nummer für Script

ListofAnz = {"ein aktives Thema", "viele Themen"}
AnzStr=MsgBox.ListAsString(ListofAnz,"um Legende zu korrigieren",
    "Auswahl der Anzahl der Themen im View"++theView.AsString)
IdxAnz = ListofAnz.FindByValue(AnzStr)

LofArt={"Querprofilschnitt (PolyLine) in Qprf-View",
       "Querprofilschnitt (Polygon) in Qprf-View",
       "Querprofilschnitt (Bohrung) in Qprf-View",
       "beliebiger Profilschnitt",
       "geologische Karte oder Formen in Kt-View", "Punkt-Thema"}
ArtofThemen = MsgBox.ListAsString(LofArt,"zur Korrektur der Legende",
           "Feststellung der Art der Themen")
IdxArt = LofArt.FindByValue(ArtofThemen)

if (IdxAnz = 0) then
  theTheme=theView.GetActiveThemes.Get(0)
  ThStr = theTheme.AsString
  kt00=MsgBox.YesNo("Ist das aktive Thema"++ThStr
    +NL+"zur Korrektur der Legende richtig?",
    "Kontrolle des aktiven Themas", true)
  if (Not kt00) then
    MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
              "Das aktive Thema ist neu auszuwählen!", "")
```

```
      exit
    end
    AnzHW = 1
    AnzHWIdx = 0
  elseif (IdxAnz = 1) then
   if (IdxArt < 3) then
    Listof2B={"B5","C5","F5","G5","K5","P5","S5","U5","V5","X5"}
    V2B=MsgBox.ChoiceAsString(Listof2B,
      "Die ersten zwei Buchstaben der vorhandenen Querprofilschnitte",
      "Auswahl der Querprofilschnitte im aktiven View")
    B1=V2B.Left(1)

    'Bestimmung des minimalen und maximalen Hochwertes im View
    ListofThemes=theView.GetThemes
    AnzThms=ListofThemes.Count
    ThmsIdx=AnzThms-1
    minHW=5642000
    maxHW=5618000
    for each eThm in 0..ThmsIdx
       theagTh=ListofThemes.Get(eThm)
       agThStr=theagTh.AsString
       erst2B=agThStr.Left(2)
       if (erst2B = V2B) then
          ListofagThStr=agThStr.AsTokens("VXSKPFfGBCU.shp")
          aHWBP=ListofagThStr.Get(0)
          aHWBPNr=aHWBP.AsNumber
          if (aHWBPNr < minHW) then
            minHW=aHWBPNr
          end
          if (aHWBPNr > maxHW) then
            maxHW=aHWBPNr
          end
       end
    end
    minHWStr=minHW.AsString
    maxHWStr=maxHW.AsString
    AnfHWStr=MsgBox.Input(
      "um die Legenden der Querprofilschnitte im View"
      ++theView.AsString++"zu verändern",
      "Eingabe des kleinsten Hochwertes der Querprofilschnitte",
      minHWStr)
    EndHWStr=MsgBox.Input(
      "um die Legenden der Querprofilschnitte im View"
      ++theView.AsString++"zu verändern",
      "Eingabe des größten Hochwertes der Querprofilschnitte",
       maxHWStr)
    AnfHW=AnfHWStr.AsNumber
    EndHW=EndHWStr.AsNumber
    HWAbstStr00="50"
    HWAbstStr =MsgBox.Input("zwischen Querprofilschnitten",
         "Eingabe eines HW-Abstandes (m)", HWAbstStr00)
    HWAbst=HWAbstStr.AsNumber

    AnzHW=(EndHW-AnfHW)/ HWAbst +1
    AnzHWIdx=AnzHW-1
     'MsgBox.Info(AnzHW.AsString,
     "'Anzahl der gesuchten Themen im View:"++theView.AsString)

    'Auswahl eines Themas, um das Feld zur Klassifizierung
    'zu bestimmen
    aHW=AnfHW
    aHWInt=AnfHW.SetFormat("").SetFormat("d")
```

```
    afnStr=B1+(aHWInt).AsString+".shp"
    theTheme=theView.FindTheme(afnStr)
   elseif (IdxArt > 2) then
    MsgBox.Error("Das ausgewählte Thema kann nicht als ein"
      +NL+"Teil mehrerer Themen bearbeitet werden!"
      +NL+"Das Programm wird abgebrochen!", "")
    exit
   end
end

'Definition der Farbe und des Musters der Legende
'Bezeichnung der Datensätze

aListofdfm = {"---",20,41, "a",43,6,  "a/Mj",24,6,  "Aussen",5,0,
    "d",36,38, "f",8,13, "Gy",24,36,  "H",32,43, "hg/plRR",3,35,
    "Hj",32,43, "Hn",11,34, "Lf",17,8, "Lfh/N",14,6, "Lö",47,8,
    "Lö/Hj",30,27, "Lö/Mj",24,29, "Löy",6,32,  "Ma",29, 28, "mi-olK",35,4,
    "miIV",11, 4, "Mj",26, 28, "N",16, 46, "plRR",53, 7, "Rhein",20,44,
    "Sf",18,5, "Sfh/N",16,5,  "sSo",24,41, "tAb",53,38, "tTt",51,35,
    "Deck",53,38,  "D",53,38, "De",53,38, "NT",14,46, "MT",26,28,
    "HT",32,43, "Präq",3,5, "Präm",3,5,  "GH",53,38, "NA",5,28, "NQ",8,5,
    "TOK",14,23, "NTabF",5,28, "MTabF",8,5, "QB",8,5, "QmitD",20,10,
    "QohneD",26,23, "QohneT",8,21, "TrorAe",3,5,  "UMT",26,28,
    "UMT III",26,28, "UMT IV",26,33, "OMT",41,39, "rMTl",20,28,
    "lMTl",26,28, "rHTl",32,43, "lNT",14,46, "rNT",17,46, "lHTr",32,43,
    "rHTr",32,43, "HTr",32,43, "MTl",26,28, "MTr",20,28, "HTl",32,43,
    "Deck-Mu",53,38, "Deck-LS",47,23, "UMT2",26,28, "Holst-Sd",38,8,
    "Holst-Tf",35,34, "UMT1",41,35, "Holst-Ton",23,15, "Holst-U",44,7,
    "SdScht-HR",45,9, "MMT-R",20,33, "Geländeoberfläche",53,38,
    "Basisfläche der Deckschichten (TOK)",14,46, "NT abgedeckte Fläche",5,28,
    "MT abgedeckte Fläche",8,5, "Quartärbasis",8,5,
    "Oberfläche der als MT ältere Schichten",8,5,
    "Oberfläche der Präquartär-Schichten",8,5,
    "Deckschichten",53,38, "Niederterrassen",14,46, "Mittelterrassen",26,28,
    "Präquartäre Schichten",3,5,
    "Deckschichten (Mutter oder Waldboden)",53,38,
    "Deckschichten (Lehm oder Sand)",47,23, "Untere Mittelterrasse 2",26,28,
    "Untere Mittelterrasse 1",41,35, "Holstein (Torf, z.T. Tonhaltig)",35,34,
    "Holstein (Sandschichten mit Tonlagen)",38,8,
    "Holstein (Tonschichten)",23,15, "Holstein (Schluffschichten)",44,7,
    "Sandschichten (z.T. Holstein, z.T. Rinnenschotter)",45,9,
    "Die mittlere Mittelterrasse (Rinnenschotter)",20,33,
    "Untere Mittelterrasse",26,28, "Obere Mittelterrasse",41,39,
    "Hauptterrasse",32,43, "als MT ältere Schichten",3,5,
    "Präquartär-Schichten",3,5, "Untere Mittelterrasse III",26,33,
    "Untere Mittelterrasse IV",26,28, "Sonst",0,0}

Legendeinfo = 0

aFTab = theTheme.GetFTab
ListofFlds = aFTab.GetFields
aShpFldClassStr = aFTab.GetShapeClass.GetClassName

AnzFlds = ListofFlds.Count
IdxFlds = AnzFlds - 1

'Feststellung des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxFlds
   aFld=ListofFlds.Get(i)
   aFldStr=aFld.GetName
```

```
    if (aFldStr <> "Shape") then
        FldStr = FldStr + aFldStr+"; "
        aValue=aFTab.ReturnValue(aFld, 0)
        DtStr=DtStr+aValue.AsString+"; "
    end
end

MsgBox.Report("Die Namen der Felder"+NL+FldStr+NL+NL+
            "Der erste Datensatz:"+NL+DtStr,
            "Information")
aSchtFld = MsgBox.ListAsString(ListofFlds,
  "für Klassifizierung der Legende",
  "Auswahl eines Feldes des Themas"++theTheme.AsString)
aSchFldStr = aSchtFld.AsString

if ((aShpFldClassStr = "Polygon") or
   (aShpFldClassStr = "PolygonZ")) then
   LegendeArt = {"Farbe", "Muster"}
   aLA = MsgBox.ListAsString(LegendeArt,"zur Füllung des Polygons",
       "Auswahl einer Art")

   if (aLA = "Farbe") then
     av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
     Farb = "ein"
   elseif (aLA = "Muster") then
     av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_FILL)
     Farb = "aus"
   end
elseif (((aShpFldClassStr = "Point") or
      (aShpFldClassStr = "Polyline")) or
      (aShpFldClassStr = "PolylineZ")) then
   av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
   Farb = "ein"
end
thePalette = av.GetSymbolWin.GetPalette

av.ShowMsg("Veränderung der Legenden der Themen...")
av.ShowStopButton
For each Thm in 0..AnzHWIdx
  Ng=Thm+1

  if (IdxAnz = 1) then
    aHW=AnfHW+(HWAbst*Thm)
    aHWInt=aHW.SetFormat("").SetFormat("d")
    afnStr=B1+(aHWInt).AsString+".shp"
    theTheme=theView.FindTheme(afnStr)
  end

  theLegend=theTheme.GetLegend
  theLegend.SetLegendType(#Legend_Type_Unique)
  theLegend.Unique(theTheme, aSchFldStr)
  ListofKlasse=theLegend.GetClassifications
  AnzKlasse=ListofKlasse.Count
  IdxKlasse=AnzKlasse-2
  'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")

  aListofSColor = {}
  aListofMuster = {}

  for each i in 0..IdxKlasse
     theKlasseLb=ListofKlasse.Get(i).GetLabel
     'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
```

```
    aIdxLb = aListofdfm.FindByValue(theKlasseLb)
    if (aIdxLb <> -1) then
      if ((aShpFldClassStr = "Polygon") or
        (aShpFldClassStr = "PolygonZ")) then
        if (aLA = "Farbe") then
          aCNr = aListofdfm.Get((aIdxLb + 1))
        elseif (aLA = "Muster") then
          aMNr = aListofdfm.Get((aIdxLb + 2))
        end
      elseif (((aShpFldClassStr = "Point") or
           (aShpFldClassStr = "Polyline")) or
           (aShpFldClassStr = "PolylineZ")) then
        aCNr = aListofdfm.Get((aIdxLb + 1))
      end
    elseif (aIdxLb = -1) then
      if ((aShpFldClassStr = "Polygon") or
        (aShpFldClassStr = "PolygonZ")) then
        if (aLA = "Farbe") then
          aR = i Mod 60
          if (aR = 0) then
            aCNr = 1
          elseif (aR <> 0) then
            aCNr = aR
          end
        elseif (aLA = "Muster") then
          aR = i Mod 47
          if (aR = 0) then
            aMNr = 1
          elseif (aR <> 0) then
            aMNr = aR
          end
        end
      elseif (((aShpFldClassStr = "Point") or
           (aShpFldClassStr = "Polyline")) or
           (aShpFldClassStr = "PolylineZ")) then
        aR = i Mod 60
        if (aR = 0) then
          aCNr = 1
        elseif (aR <> 0) then
          aCNr = aR
        end
      end
      Legendeinfo = 1
    end

    if (Farb = "ein") then
      theColor=(thePalette.GetList(#PALETTE_LIST_COLOR).Get(aCNr))
      theRgbList=theColor.GetRgbList
      aColor=Color.Make
      aColor.SetRgbList(theRgbList)
      aListofSColor.Add(aColor)
    elseif (Farb = "aus") then
      theMuster=(thePalette.GetList(#PALETTE_LIST_FILL). Get(aMNr))
      aListofMuster.Add(theMuster)
    end
  end

  aListofSymbol=theLegend.GetSymbols
  AnzSymb=aListofSymbol.Count
  AnzSymbIdx=AnzSymb-2
  'MsgBox.Info(AnzSymb.AsString, "Anzahl der Symbole")
```

```
  if (Farb = "ein") then
    for each symb in 0..AnzSymbIdx
      aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
    end
  elseif (Farb = "aus") then
    for each symb in 0..AnzSymbIdx
      theSymbol=aListofSymbol.Get(symb)
      theFillM=aListofMuster.Get(symb)
      theSymbol.Copy(theFillM)
      theSymbol.SetColor(Color.GetBlack)
      theSymbol.SetOlColor(Color.GetBlack)
      theSymbol.SetOlWidth(0.1)
    end
  end
  theTheme.UpdateLegend
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzHW*100)
  if (not more) then
    break
  end
end
if (Legendeinfo = 1) then
   MsgBox.Report("Die Farbe der Legende ist zum Teil"
        +NL+"oder gar nicht definiert!"
        +NL+"Die Definition der Farbe"
        +NL+"in diesem Programm ist zu ändern.",
   "Information")
end
```

```
'lgdgrh11.ave
'Eine Legende eines Polygon-Themas für Schichten (rechtsrheinische
'Mittelterrassen) wird neben der Abbildung als Graphik hergestellt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc  'ein aktives Profilschnitt-View

'Zeichnung der Legende rechts neben der Abbildung als Graphik

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_FILL)
thePalette=av.GetSymbolWin.GetPalette

theDisplay=theView.GetDisplay
theGraphicList=theView.GetGraphics

theGString=("Legende")
theGPoint=Point.Make(5641300.00, 3875.00)
theGText=GraphicText.Make(theGString, theGPoint)
theGText.SetAlignment(#TEXTCOMPOSER_JUST_LEFT)

theTextSymbol=theGText.ReturnSymbols.Get(0)
theTextSymbol.SetSize(9)
newFont=Font.Make("Arial", "normal")
theTextSymbol.SetFont(newFont)
theTextSymbol.SetColor(Color.GetBlack)

theGraphicList.Add(theGText)
```

```
ListofLPg={}
xlL=5641300.00
xrL=5641550.00
yoL=3700.00
yuL=3525.00

ListofGP={}
xText=5641725.00
yText=3525.00

'10 Polygone für Legende werden erzeugt und in Liste gespeichert.
for each aL in 0..9
  thePolyg=Polygon.Make({{xlL@yoL, xrL@yoL, xrL@yuL, xlL@yuL}})
  ListofLPg.Add(thePolyg)
  yoL=yoL-350.00
  yuL=yoL-175.00
  ListofGP.Add(xText@yText)
  yText=yText-350.00
end

AnzLPg=ListofLPg.Count
IdxLPg=AnzLPg-1

'Definition des Musters der Legende
'Bezeichnung der Datensätze
aListofGStr={"Deckschichten"+NL+"Mutter- oder Waldboden",
          "Deckschichten"+NL+"Lehm oder Sand",
          "Untere Mittelterrasse 2"+NL+"Kies, Sand",
          "Holstein"+NL+"Sandschichten",
          "Holstein"+NL+"Torfschichten, z.T. tonhaltig",
          "Untere Mittelterrasse 1"+NL+"Kies, sandig",
          "Holstein"+NL+"Tonschichten",
          "Holstein"+NL+"Schluffschichten",
          "Sandschichten"+NL+"z.T. Holstein, z.T. Rinnenschotter",
          "Rinnenschotter"+NL+"Kies, Sand"}
'Die Nummer des Musters im Symbolwindow
aListofMNr={38,23,28,8,34,35,15,7,9,33}
Legendeinfo = 0


if (IdxLPg > 9) then
  MsgBox.Error("Es gibt mehr Schichten als Legende!"+NL+
    "Deshalb kann Legende hiermit nicht gezeichnet werden!", "")
elseif (IdxLPg < 10) then
  for each aRec in 0..IdxLPg
    thePolyg=ListofLPg.Get(aRec)
    theGraphicPolygon=GraphicShape.Make(thePolyg)
    theMNr = aListofMNr.Get(aRec)
    theGString = aListofGStr.Get(aRec)

    theSymbol=theGraphicPolygon.GetSymbol
    theSymbol=thePalette.GetList(#PALETTE_LIST_FILL).Get(theMNr)
    theSymbol.SetColor(Color.GetBlack)
    theSymbol.SetOlColor(Color.GetBlack)
    theSymbol.SetOlWidth(0.1)

    theStyle=theSymbol.GetStyle
    'MsgBox.Info(theStyle.AsString, "Symbol_Style")
    theGraphicPolygon.SetSymbol(theSymbol)
    theGraphicPolygon.Invalidate
    theGraphicList.Add(theGraphicPolygon)
```

```
      theGPoint=ListofGP.Get(aRec)
      theGText=GraphicText.Make(theGString, theGPoint)
      theGText.SetAlignment(#TEXTCOMPOSER_JUST_LEFT)

      theTextSymbol=theGText.ReturnSymbols.Get(0)
      theTextSymbol.SetSize(7)

      newFont=Font.Make("Arial", "normal")
      theTextSymbol.SetFont(newFont)
      theTextSymbol.SetColor(Color.GetBlack)
      theGText.Invalidate
      theGraphicList.Add(theGText)
  end
end
theDisplay.Invalidate(true)



'maustast.ave
'Ein Punkt-, Polyline-, Polygon- oder PolygonZ-Thema wird durch
'Maus-Klicken oder Tastatur-Eingaben in einem aktiven 2D-View hergestellt.
'Dieses Script wird als ein Werkzeug (Tool) im aktiven View benutzt.

theProject = av.GetProject
theView = av.GetActiveDoc   'Ein aktives View

'mehrmalige Klicken auf das Bildschirm im aktiven View zur Bestimmung der Stelle
aEingPolyL = theView.GetDisplay.ReturnUserPolyLine

ListofListofEingPt = aEingPolyL.AsList
ListofEingPt = ListofListofEingPt.Get(0)
AnzM = ListofEingPt.Count
AnzMIdx = AnzM - 1

ListofMPt = {}
ListofPt = {}

for each mPt in 0..AnzMIdx
  aMPt = ListofEingPt.Get(mPt)
  ListofMPt.Add(aMPt)
  ListofPt.Add(aMPt)
end

ListofFormen = {"Punkte", "Polyline", "Polygon", "PolygonZ"}
aForm = MsgBox.ListAsString(ListofFormen, "für neues Thema", "Auswahl einer Form")

if ((aForm = "Polygon") or (aForm = "PolygonZ")) then
  MsgBox.Info("Bei einem Polygon sind die Punkte im Uhrzeigersinn einzugeben.",
           "Information")
  aAnzPt = ListofMPt.Count
  if (aAnzPt < 3) then
    MsgBox.Error("Für Herstellung eines Polygons sind mindestens"
     ++"3 Punkte notwendig!"+NL+"Das Programm wird abgebrochen!", "")
    Exit
  end
end

HFaktorStr = MsgBox.Input("zur vertikalen Überhöhung der y-Koordinate"
+NL+"bei einem Profilschnitt."+NL+"Bei einer Kartendarstellung: Faktor = 1",
         "Eingabe eines Faktors", "50")
```

```
HFaktor = HFaktorStr.AsNumber

ListofPunkte = {"Maus-Klicken", "Tastatur-Eingabe"}
aPtW = MsgBox.ListAsString(ListofPunkte,
      "durch","Eingabe der Punkte")

if (aPtW = "Tastatur-Eingabe") then
  ListofPt = {}
  MNr = -1
  weiter = true

  while (weiter)
    MNr = MNr + 1
    Nr = MNr + 1
    if (MNr <= AnzMIdx) then
      aMPt = ListofMPt.Get(MNr)
      aMPtxStr = (aMPt.Getx.SetFormat("d.dd")).AsString
      aMPtyStr = (((aMPt.Gety)/HFaktor).SetFormat("d.dd")).AsString
    end
    aTxStr = MsgBox.Input("die"++Nr.AsString+"."++"x-Koordinate [m]",
          "Eingabe einer Koordinate", aMPtxStr)
    aTyStr = MsgBox.Input("die"++Nr.AsString+"."++"y-Koordinate [m]",
          "Eingabe einer Koordinate", aMPtyStr)
    aTx = aTxStr.AsNumber
    aTy = aTyStr.AsNumber
    aTPt = Point.Make(aTx, aTy)
    ListofPt.Add(aTPt)
    weiter = MsgBox.YesNo("Ist noch ein weiterer Punkt einzugeben?",
          "Tastatur-Eingabe", true)
  end
end

if (aForm = "PolygonZ") then
  aNPtzStr = MsgBox.Input("für alle Punkte vom PolygonZ",
        "Eingabe einer z-Koordinate [m]", "0.00")
  aNPtz = aNPtzStr.AsNumber
end

theKW = MsgBox.Input("für die neue Polyline", "Eingabe eines Kennwortes",
                "Grmtro")

'Speicherung der neuen Polyline
ListofSp11 = {"Herstellung als ein neues Thema",
          "Speicherung in einem vorhandenen Thema"}

AW11 = MsgBox.ListAsString(ListofSp11,
     "zur Speicherung der neuen Polyline",
     "Auswahl eines Themas")

if (AW11 = "Herstellung als ein neues Thema") then
  WDStr=av.GetProject.GetWorkDir.AsString

  fn00 = "Plrmto"
  fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
  'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File ("+aForm+")")
  if (fName =nil) then exit end
  fName.SetExtension("shp")
  if (aForm = "Punkte") then
    NFTab=FTab.MakeNew(fName, Point)
  elseif (aForm = "Polyline") then
    NFTab=FTab.MakeNew(fName, Polyline)
```

```
  elseif (aForm = "Polygon") then
    NFTab=FTab.MakeNew(fName, Polygon)
  elseif (aForm = "PolygonZ") then
    NFTab=FTab.MakeNew(fName, PolygonZ)
  end

  NShpFld = NFTab.FindField("shape")
  'aCNm = NFTab.GetShapeClass.GetClassName
  'MsgBox.Info(aCNm, "die neue Form")
  NIDFld=Field.Make("ID", #Field_Short, 3, 0)
  NKWFld=Field.Make("Kennwort", #Field_Char, 10, 0)
  ListofNFlds={NIDFld, NKWFld}
  NFTab.AddFields(ListofNFlds)
  NThm=FTheme.Make(NFTab)
  theView.AddTheme(NThm)
  recNr = -1

elseif (AW11 = "Speicherung in einem vorhandenen Thema") then

  ListofThms=theView.GetThemes
  ListofPtThms = {}
  ListofPLThms = {}
  ListofPqThms = {}
  ListofPgzThms = {}

  for each aT in ListofThms
    if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aCNm = aFTab.GetShapeClass.GetClassName
      if (aCNm = "Point") then
        ListofPtThms.Add(aT)
      elseif (aCNm = "PolyLine") then
        ListofPLThms.Add(aT)
      elseif (aCNm = "Polygon") then
        ListofPqThms.Add(aT)
      elseif (aCNm = "PolygonZ") then
        ListofPgzThms.Add(aT)
      end
    end
  end

  if (aForm = "Punkte") then
    NThm = MsgBox.ChoiceAsString(ListofPtThms,
      "zur Speicherung der neuen Punkte",
      "Auswahl eines Themas im View"++theView.AsString)
  elseif (aForm = "Polyline") then
    NThm = MsgBox.ChoiceAsString(ListofPLThms,
      "zur Speicherung der neuen Polyline",
      "Auswahl eines Themas im View"++theView.AsString)
  elseif (aForm = "Polygon") then
    NThm = MsgBox.ChoiceAsString(ListofPqThms,
      "zur Speicherung der neuen Polygon",
      "Auswahl eines Themas im View"++theView.AsString)
  elseif (aForm = "PolygonZ") then
    NThm = MsgBox.ChoiceAsString(ListofPgzThms,
      "zur Speicherung der neuen PolygonZ",
      "Auswahl eines Themas im View"++theView.AsString)
  end

  NFTab = NThm.GetFTab
  NShpFld = NFTab.FindField("Shape")
  NIDFld = NFTab.FindField("ID")
```

```
  'Anzahl der Datensätze im Thema
  AnzNDs=0
  for each rec in NFTab
     AnzNDs = AnzNDs + 1
  end
  IdxNDs = AnzNDs - 1
  recNr = IdxNDs

  'Anzahl der Felder im Thema
  ListofNFlds = NFTab.GetFields
  AnzNFlds = ListofNFlds.Count
  IdxNFlds = AnzNFlds - 1

 'Feststellung der Datensätze im Thema
  ListofPLDatens = {}
  FldStr=""
  for each j in 0..IdxNDs
     DtStr=""
     for each i in 0..IdxNFlds
        aFld = ListofNFlds.Get(i)
        aFldStr = aFld.GetName
        if (aFldStr <> "Shape") then
           if (j = 0) then
              FldStr = FldStr + aFldStr+"; "
           end
           aValue = NFTab.ReturnValue(aFld, j)
           DtStr=DtStr+aValue.AsString+"; "
        end
     end
     ListofPLDatens.Add(DtStr)
   end

   MsgBox.ListAsString(ListofPLDatens, FldStr,
         "Die Namen der Felder und Datensätze")
   NKWFld = MsgBox.ListAsString(ListofNFlds,
       "für Kennwort der Polyline",
       "Auswahl eines Feldes des Themas"++NThm.AsString)
end

av.ShowMsg("Speicherung der neuen Formen"++aForm++" ...")

AnzNPt = ListofPt.Count
IdxNPt = AnzNPt - 1
ListofNPt = {}
ListofNPtZ = {}

for each i in 0..IdxNPt
   aNPt = ListofPt.Get(i)
   aNPtx = aNPt.Getx
   aNPty = (aNPt.Gety) * HFaktor
   if (aForm = "PolygonZ") then
      ListofNPtZ.Add(aNPtx@aNPty@aNPtz)
   else
      ListofNPt.Add(aNPtx@aNPty)
   end
end

if (aForm = "PolygonZ") then
   ListofListofNPtZ = {}
   ListofListofNPtZ.Add(ListofNPtZ)
else
```

```
  ListofListofNPt = {}
  ListofListofNPt.Add(ListofNPt)
end

NFTab.SetEditable(false)
NFTab.SetEditable(true)

if (aForm = "Punkte") then
  for each i in 0..IdxNPt
    theNShp = ListofNPt.Get(i)
    recNr = recNr + 1
    NFTab.AddRecord
    NFTab.SetValue(NShpFld, recNr, theNShp)
    NFTab.SetValue(NIDFld, recNr, recNr)
    NFTab.SetValue(NKWFld, recNr, theKW)
  end
else
  if (aForm = "Polyline") then
    theNShp = Polyline.Make(ListofListofNPt)
  elseif (aForm = "Polygon") then
    theNShp = Polygon.Make(ListofListofNPt)
  elseif (aForm = "PolygonZ") then
    theNShp = PolygonZ.Make(ListofListofNPtZ)
  end
  recNr = recNr + 1
  NFTab.AddRecord
  NFTab.SetValue(NShpFld, recNr, theNShp)
  NFTab.SetValue(NIDFld, recNr, recNr)
  NFTab.SetValue(NKWFld, recNr, theKW)
end
NFTab.SetEditable(false)

theLegend = NThm.GetLegend
theLegend.SetLegendType(#LEGEND_TYPE_SIMPLE)
theLegend.SingleSymbol
if ((aForm = "Polygon") or (aForm = "PolygonZ")) then
  theLegend.GetSymbols.Get(0).SetColor(Color.GetBlue)
else
  theLegend.GetSymbols.Get(0).SetColor(Color.GetBlack)
end

NThm.UpdateLegend



'mttokran.ave
'Bestimmung des Randes der Terrassenoberkante der Mittelterrasse
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject = av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("mttokran")
myScript.SetNumberFormat("d.dd")

ListofThms=theView.GetThemes
ListofFThms = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
       ListofFThms.Add(aT)
```

```
      end
   end
end

'Eingabe eines Punkt-Themas für eine Grenze der Terrassenoberkante der NT
TOKNTThm=MsgBox.ChoiceAsString(ListofFThms,
      "Grenze der Terrassenoberkante der NT", "Auswahl eines Punkt-Themas")
NTFTab=TOKNTThm.GetFTab
ListofNTFlds=NTFTab.GetFields
NTShpFld = NTFTab.FindField("Shape")
NTHFld=MsgBox.ChoiceAsString(ListofNTFlds, "Eingabe des Feldes für Höhen",
        "Auswahl eines Feldes im Thema:"++TOKNTThm.AsString)
AnzNTPts=0
Listof3DNTPts={}
Nr=-1
for each rec in NTFTab
   Nr=Nr+1
   AnzNTPts=AnzNTPts+1
   thePt=NTFTab.ReturnValue(NTShpFld, Nr)
   theRW=thePt.Getx
   theHW=thePt.Gety
   theHh=NTFTab.ReturnValue(NTHFld, Nr)
   Listof3DNTPts.Add(theRW@theHW@theHh)
end
IdxNTPts=AnzNTPts-1

'Eingabe des Punkt-Themas für eine Grenze der Quartärbasis der NT
QbThm=MsgBox.ChoiceAsString(ListofFThms,
      "Grenze der Quartärbasis der NT", "Auswahl eines Punkt-Themas")
QbFTab=QbThm.GetFTab
ListofQFlds=QbFTab.GetFields
QbShpFld= QbFTab.FindField("Shape")
QbHFld=MsgBox.ChoiceAsString(ListofQFlds, "Eingabe des Feldes für Höhen",
       "Auswahl eines Feldes im Thema:"++QbThm.AsString)
AnzQbPts=0
Listof3DQbPts={}
Nr=-1
for each rec in QbFTab
   Nr=Nr+1
   AnzQbPts=AnzQbPts+1
   thePt=QbFTab.ReturnValue(QbShpFld, Nr)
   theRW=thePt.Getx
   theHW=thePt.Gety
   theHh=QbFTab.ReturnValue(QbHFld, Nr)
   Listof3DQbPts.Add(theRW@theHW@theHh)
end
IdxQbPts=AnzQbPts-1

'Eingabe des Punkt-Themas für Grenze der Terrassenoberkante der MT
TOKMTThm=MsgBox.ChoiceAsString(ListofFThms,
      "Grenze der Terrassenoberkante der MT", "Auswahl eines Punkt-Themas")
MTFTab=TOKMTThm.GetFTab
ListofMTFlds=MTFTab.GetFields
MTShpFld= MTFTab.FindField("Shape")
MTHFld=MsgBox.ChoiceAsString(ListofMTFlds, "Eingabe des Feldes für Höhen",
       "Auswahl eines Feldes in Thema:"++TOKMTThm.AsString)
AnzMTPts=0
Listof3DMTPts={}
Nr=-1
for each rec in MTFTab
   Nr=Nr+1
   AnzMTPts=AnzMTPts+1
```

```
    thePt=MTFTab.ReturnValue(MTShpFld, Nr)
    theRW=thePt.Getx
    theHW=thePt.Gety
    theHh=MTFTab.ReturnValue(MTHFld, Nr)
    Listof3DMTPts.Add(theRW@theHW@theHh)
end
IdxMTPts=AnzMTPts-1

'Die neuen Grenze-Punkte für den Rand der Terrassenoberkante der MT
av.ShowMsg("Bestimmung der neuen Koordinaten der TOK-Grenze-Punkte der MT")
av.ShowStopButton
ListofNMtPts={}
ListofListof2DPt={}
minEntf=10000
'Der Punkt mit der kürzesten Entfernung wird gesucht.
for each aPt in 0..IdxNTPts
    Ng=aPt+1
    the3DPt=Listof3DNTPts.Get(aPt)
    NTRW=the3DPt.Getx
    NTHW=the3DPt.Gety
    NTHh=the3DPt.Getz
    minEntf=1000000
    for each ePt in 0..IdxQbPts
        the3DPt=Listof3DQbPts.Get(ePt)
        QbRW=the3DPt.Getx
        QbHW=the3DPt.Gety
        QbHh=the3DPt.Getz
        theEntf=(((QbRW-NTRW)^2)+((QbHW-NTHW)^2)).Sqrt
        if (theEntf < minEntf) then
            minEntf=theEntf
            minQbRW=QbRW
            minQbHW=QbHW
            minQbHh=QbHh
        end
    end
    Listof2DPt={}
    Listof2DPt.Add(NTRW@NTHW)
    Listof2DPt.Add(minQbRW@minQbHW)
    ListofListof2DPt.Add(Listof2DPt)
    xq=minQbRW
    yq=minQbHW
    minMtEntf=1000000
    for each ePt in 0..IdxMTPts
        the3DPt=Listof3DMTPts.Get(ePt)
        MTRW=the3DPt.Getx
        MTHW=the3DPt.Gety
        MTHh=the3DPt.Getz
        theEntf=(((MTRW-NTRW)^2)+((MTHW-NTHW)^2)).Sqrt
        if (theEntf < minMtEntf) then
            minMtEntf=theEntf
            minMTRW=MTRW
            minMTHW=MTHW
            minMTHh=MTHh
        end
    end
    MTHh=minMTHh  'da die Höhen der MT in der Nähe fast gleich sind
    HMT=(MTHh-minQbHh)
    HNT=(NTHh-minQbHh)
    if (NTRW > MTRW) then
      EntfMTQb = minEntf + minMtEntf
    elseif (NTRW <= MTRW) then
      EntfMTQb = minEntf - minMtEntf
```

```
    end
    if (NTRW <> minQbRW) then
      Neig=(NTHW-minQbHW)/(NTRW-minQbRW)
      Schn=NTHW-(Neig*NTRW)
      P=((2*Neig*Schn)-(2*Neig*yq)-(2*xq))/(1+(Neig^2))
      q=((xq^2)+(Schn^2)-(2*Schn*yq)+(yq^2)-(EntfMTQb^2))/(1+(Neig^2))
      x1=((-1)*(p/2))+((((p/2)^2)-q).Sqrt)
      x2=((-1)*(p/2))-((((p/2)^2)-q).Sqrt)
      y1=Neig*x1+Schn
      y2=Neig*x2+Schn
      ListofNMtPts.Add(x1@y1@MTHh)
      ListofNMtPts.Add(x2@y2@MTHh)
    elseif (NTRW = minQbRW) then
      if (HNT <> 0) then
        nMTHW=(HMT/HNT)*(NTHW-yq)+yq
      elseif (HNT = 0) then
        nMTHW=(HMT)*(NTHW-yq)+yq
      end
      nMTRW=NTRW
      ListofNMtPts.Add(nMTRW@nMTHW@MTHh)
    end
      'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzNTPts*100)
    if (not more) then
      break
    end
end

'Ein Feature-Shape-File (Punkt) wird hergestellt.
WDStr=theProject.GetWorkDir.AsString
fnDefault=FileName.Make(WDStr).MakeTmp("Grtokmtn","shp")
'fnDefault=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("Grtokmtn","shp")
fnOutPut=FileDialog.Put(fnDefault, "*.shp", "Output shape File (Point)")
if (fnOutPut=nil) then exit end
fnOutPut.SetExtension("shp")
ftbOutPut=FTab.MakeNew(fnOutPut, point)
ShapeField1=ftbOutPut.FindField("shape")
IDField1=Field.Make("ID", #Field_SHORT, 4, 0)
RWField1=Field.Make("RW", #Field_Float, 10, 2)
HWField1=Field.Make("HW", #Field_Float, 10, 2)
TOKField1=Field.Make("MTTOKH", #Field_Float, 8, 4)
ListofftbFld={IDField1, RWField1, HWField1, TOKField1}
ftbOutPut.AddFields(ListofftbFld)
ftbOutPut.SetEditable(false)
ftbOutPut.SetEditable(true)
AnzNeuPt=ListofNMtPts.Count
IdxNeuPt=AnzNeuPt-1
av.ShowMsg("Speicherung der neuen Grenzdatei für Terrassenoberkante der MT")
av.ShowStopButton
for each i in 0..IdxNeuPt
    Ng=i+1
    thePt=ListofNMtPts.Get(i)
    theRW=thePt.Getx
    theHW=thePt.Gety
    theHh=thePt.Getz
    theShp=Point.Make(theRW, theHW)
    ftbOutPut.Addrecord
    ftbOutPut.SetValue(ShapeField1, i, theShp)
    ftbOutPut.SetValue(IDField1, i, i)
    ftbOutPut.SetValue(RWField1, i, theRW)
    ftbOutPut.SetValue(HWField1, i, theHW)
    ftbOutPut.SetValue(TOKField1, i, theHh)
```

```
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzNeuPt*100)
    if (not more) then
      break
    end
end
ftbOutPut.SetEditable(false)
thmNew=FTheme.Make(ftbOutPut)
theView.AddTheme(thmNew)

'Ein Feature-Shape-File (Polyline) wird hergestellt.
fnStr=FileName.Make(WDStr).MakeTmp("Plntqb","shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("Plntqb","shp")
fnGrMT=FileDialog.Put(fnStr, "*.shp", "Output shape File (Polyline)")
if (fnGrMT=nil) then exit end
fnGrMT.SetExtension("shp")
nPLFTab=FTab.MakeNew(fnGrMT, polyline)
nPLShpFld=nPLFTab.FindField("shape")
nPLIDFld=Field.Make("ID", #Field_SHORT, 4, 0)
nPLFTab.AddFields({nPLIDFld})

nPLFTab.SetEditable(false)
nPLFTab.SetEditable(true)
AnzPL=ListofListof2DPt.Count
IdxPL=AnzPL-1
av.ShowMsg("Speicherung der neuen Grenzdatei für Terrassenoberkante der MT")
av.ShowStopButton
for each i in 0..IdxPL
    Ng=i+1
    theList=ListofListof2DPt.Get(i)
    ListofListofnPL={}
    ListofListofnPL.Add(theList)
    thePL=Polyline.Make(ListofListofnPL)
    nPLFTab.Addrecord
    nPLFTab.SetValue(nPLShpFld, i, thePL)
    nPLFTab.SetValue(nPLIDFld, i, i)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzPL*100)
    if (not more) then
      break
    end
end
nPLFTab.SetEditable(false)
thmPLNew=FTheme.Make(nPLFTab)
theView.AddTheme(thmPLNew)
```

```
'pfabm2th.ave
'Höhen eines Querprofilschnittes werden in einem Abschnitt durch einen Mittelwert
'der Höhen der beiden anderen Querprofilschnitte im gleichen Abschnitt (im gleichen
'Bereich des Rechtswertes) gebildet. Die beiden Querprofilschnitte liegen an den
'beiden Seiten des Querprofilschnittes in den anderen Themen. Der Abschnitt wird
'durch Klicken der Maus auf den Querprofilschnitt bestimmt.
'Dieses Script wird als ein Werkzeug (Tool) im aktiven Querprofilschnitt-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("pfabm2th")
```

```
myScript.SetNumberFormat( "d.dd") ' script default

'zweimalige Maus-Klicken auf das Bildschirm
aEingPolyL=theView.GetDisplay.ReturnUserPolyLine

ListofThemen = {}
theTheme=theView.GetActiveThemes.Get(0)
ThStr=theTheme.AsString

kt00=MsgBox.YesNo("Ist das aktive Thema"++ThStr
     +NL+"zur Korrektur eines Abschnittes richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
           "Das aktive Thema ist neu auszuwählen!", "")
  exit
end
ListofThemen.Add(theTheme)

'Feststellung der Koordinaten des Maus-Klickens
ListofListofEingPt=aEingPolyL.AsList
ListofEingPt=ListofListofEingPt.Get(0)
AnzE=ListofEingPt.Count
AnzEIdx=AnzE-1
aMousePt=ListofEingPt.Get(0)
aMousePt2=ListofEingPt.Get(AnzEIdx)

aMPtAnfx=aMousePt.Getx
aMPtAnfy=aMousePt.Gety
aMPtEndx=aMousePt2.Getx
aMPtEndy=aMousePt2.Gety

'MsgBox.Report("Der Anfang des Abschnittes"+NL+
'            "Die X-Koordinate: "++ aMPtAnfx.AsString +NL+
'            "Die Y-Koordinate: "++ aMPtAnfy.AsString+NL+NL+
'            "Das Ende des Abschnittes"+NL+
'            "Die X-Koordinate: "++ aMPtEndx.AsString +NL+
'            "Die Y-Koordinate: "++ aMPtEndy.AsString,
'            "Koordinaten des Maus-Klickens")
ListofMPt = {}
ListofMPt.Add(aMPtAnfx@aMPtAnfy)
ListofMPt.Add(aMPtEndx@aMPtEndy)
ListofGrPt = {}
ListofListofFLPt = {}
ListofZiel = {"um einen Abschnitt zu korrigieren","um Daten zu übernehmen"}
ListofRecIdx = {}
ListofListofGrIdx = {}
ListofminDist = {}
ListofminDistIdx = {}

'Auswahl eines anderen Themas, um daten zu übernehmen (1)
ListofThemes=theView.GetThemes
ListofPLFThm = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(PolyLine)) then
       ListofPLFThm.Add(aT)
     end
   end
end
```

```
vPLTheme = MsgBox.ChoiceAsString(ListofPLFThm, ListofZiel.Get(1),
       "Auswahl eines Themas (1) im View"++ theView.AsString)
ListofThemen.Add(vPLTheme)

'Auswahl eines anderen Themas, um daten zu übernehmen (2)
nPLTheme = MsgBox.ChoiceAsString(ListofPLFThm, ListofZiel.Get(1),
       "Auswahl eines Themas (2) im View"++ theView.AsString)
ListofThemen.Add(nPLTheme)

For each aP in 0..2          'Anfang des großen Blockes
  theThm = ListofThemen.Get(aP)
  theFTab =  theThm.GetFTab
  theShpFld = theFTab.FindField("Shape")
  ListofFlds= theFTab.GetFields
  if (aP = 0) then
    ThNStr = ""
    NZ = 0
  elseif (aP > 0) then
    ThNStr = aP.SetFormat( "d").AsString
    NZ = 1
  end
  FLFld=MsgBox.ChoiceAsString(ListofFlds, "das den Namen der Fläche enthält,"
     +NL+ListofZiel.Get(NZ),
     "Auswahl eines Feldes im"++ThNStr+". Thema"++ theThm.AsString)

  'Feststellung der Anzahl der 2D-PolyLine (Datensätze) im Thema
  Anz2DPL=0
  for each rec in theFTab
    Anz2DPL=Anz2DPL+1
  end
  Anz2DPLIdx=Anz2DPL-1

  ListofFLNm={}      'eine Liste für Flächennamen in den Datensätzen des Feldes
  for each aL in 0..Anz2DPLIdx
     aFL= theFTab.ReturnValue(FLFld, aL)
     ListofFLNm.Add(aFL)
  end

  aFLNm=MsgBox.ChoiceAsString(ListofFLNm,
        ListofZiel.Get(NZ),
       "Auswahl eines Datensatzes in"++ theThm.AsString)
  'MsgBox.Info(aFLNm, "Name der Fläche im Thema"++ theThm.AsString)

  'Feststellung der Koordinaten der ausgewählten geologischen Fläche im Thema
  aFLIdx=ListofFLNm.Find(aFLNm)
  ListofRecIdx.Add(aFLIdx)
  theProfilList1={}
  theShp1= theFTab.ReturnValue(theShpFld, aFLIdx)
  theProfilList1.Add({theShp1})

  ListofFLx = {}     'Liste der Vertices der geologischen Fläche
  ListofFLy = {}
  ListofFLPt = {}
  minxkrd = 100000000
  minykrd = 100000000
  maxxkrd = 0
  maxykrd = 0

  if (theProfilList1 <> 0) then
    for each q in theProfilList1
         theLines=q.Get(0).AsList
         for each m in theLines
```

```
            for each ptx in m
               myx=ptx.Getx
               myy=ptx.Gety
               ListofFLx.Add(myx)
               ListofFLy.Add(myy)
               ListofFLPt.Add(myx@myy)
               if (myx < minxkrd) then
                  minxkrd = myx
               end
               if (myx > maxxkrd) then
                  maxxkrd = myx
               end
               if (myy < minykrd) then
                  minykrd = myy
               end
               if (myy > maxykrd) then
                  maxykrd = myy
               end
            end
         end
      end
   end

   PtAnz= ListofFLx.Count  'Anzahl der Vertices der geologischen Fläche
   PtAnzIdx= PtAnz-1
   ListofListofFLPt.Add(ListofFLPt)

'Bestimmung des Abschnittes der geologischen Fläche
'als Index der Vertices mit dem kleinsten Abstand von den Maus-Klicken

for each aM in 0..1   ' Anfang der Schleife für einen Abschnitt

minDist=1000
minIdx=-1

aMPt = ListofMPt.Get(aM)
aMPtx = aMPt.Getx
aMPty = aMPt.Gety

if (aMPtx < minxkrd) then
   aMPtx = minxkrd
elseif (aMPtx > maxxkrd) then
   aMPtx = maxxkrd
end
if (aMPty < minykrd) then
   aMPty = minykrd
elseif (aMPty > maxykrd) then
   aMPty = maxykrd
end

for each i in 0..PtAnzIdx
   xkrd= ListofFLx.Get(i)
   ykrd= ListofFLy.Get(i)
   xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
   yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
   xyAbst = ((xAbst  + yAbst).sqrt) .Abs
   if (xyAbst < minDist) then
      minDist = xyAbst
      minIdx = i  'Index des nähesten Punktes des Maus-Klickens
   end
end
ListofminDist .Add(minDist)
```

```
ListofminDistIdx.Add(minIdx)

'Bestimmung des Abschnittes der geologischen Fläche
'als x-, y-Koordinaten

if (minIdx = 0) then
   vIdx = 0     'ein Vertex auf der Linie vor dem Maus-Klicken
   nIdx = 1     'ein Vertex auf der Linie nach dem Maus-Klicken
elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
   xkrdv = ListofFLx.Get((minIdx -1))   'x-Koord. des letzten Punktes
   xkrd  = ListofFLx.Get(minIdx)        'x-Koord. des nähesten Punktes
   xkrdn = ListofFLx.Get((minIdx +1))  'x-Koord. des nächsten Punktes

   ykrdv = ListofFLy.Get((minIdx -1))   'y-Koord. des letzten Punktes
   ykrd = ListofFLy.Get(minIdx)         'y-Koord. des nähesten Punktes
   ykrdn = ListofFLy.Get((minIdx +1))   'y-Koord. des nächsten Punktes

   xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
   yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
   xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten Punkt vom nähesten Punkt

   xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
   yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
   xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten Punkt vom nähesten Punkte

   xML = (xkrd - aMPtx) * (xkrd - aMPtx)
   yML = (ykrd - aMPty) * (ykrd - aMPty)
   xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Maus-Klicken vom nähesten Punkt

   xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
   yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
   xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Maus-Klicken vom letzten Punkt

   xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
   yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
   xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Maus-Klicken vom nächsten Punkt

   vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken und dem
   vLpML = xyAv + xyML        'letzten Punkt

   nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken und dem
   nLpML = xyAn + xyML        'nächsten Punkt

   vM = (vLmML - xyMLv).Abs   'Differenz zwischen der theorischen und der
   vP = (vLpML - xyMLv).Abs   'tatsächlichen Länge im Bezug auf den letzten Punkt

   nM = (nLmML - xyMLn).Abs   'Differenz zwischen der theorischen und der
   nP = (nLpML - xyMLn).Abs   'tatsächlichen Länge im Bezug auf den nächsten Punkt

   ListofLg = {vM, vP, nM, nP}   'Suche nach der kleinsten Länge
   MinLg = 10000
   TheLgIdx = -1
   For each aLg in 0..3
        theLg = ListofLg.Get(aLg)
        if (theLg < MinLg) then
             MinLg = theLg
             TheLgIdx = aLg
        end
   end
   if (xyML < 10) then
     vIdx = minIdx - 1
     nIdx = minIdx + 1
```

```
    end
    if (xyML >= 10) then
        if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
            vIdx = minIdx - 1     'ein Vertex auf der Linie vor dem Maus-Klicken
            nIdx= minIdx          'ein Vertex auf der Linie nach dem Maus-Klicken
        elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
            vIdx = minIdx         'ein Vertex auf der Linie vor dem Maus-Klicken
            nIdx = minIdx + 1     'ein Vertex auf der Linie nach dem Maus-Klicken
        else
            vIdx = minIdx         'ein Vertex auf der Linie vor dem Maus-Klicken
            nIdx = minIdx + 1     'ein Vertex auf der Linie nach dem Maus-Klicken
        end
    end
elseif (minIdx = PtAnzIdx)   then
     vIdx = minIdx - 1
     nIdx = minIdx
end

'Bestimmung der Koordinaten des Maus-Klickens auf dem Profilschnitt
vPtx = ListofFLx.Get(vIdx)
vPty = ListofFLy.Get(vIdx)
nPtx = ListofFLx.Get(nIdx)
nPty = ListofFLy.Get(nIdx)

if ((minDist = 0) or (minDist < 10)) then 'Der näheste Punkt liegt innerhalb
   Ptx = ListofFLx.Get(minIdx)     '10 m vom dem Maus-Klicken auf der Linie
   Pty = ListofFLy.Get(minIdx)
   if ( minIdx = 0) then
     vGrIdx = minIdx
     nGrIdx = minIdx + 1
   elseif (( minIdx > 0) and ( minIdx < PtAnzIdx)) then
     vGrIdx = minIdx -1
     nGrIdx = minIdx + 1
   elseif ( minIdx = PtAnzIdx) then
     vGrIdx = minIdx -1
     nGrIdx = minIdx
   end
elseif (minDist >= 10) then  'Berechnung der Koordinaten des Punktes
      vGrIdx = vIdx      'auf der Linie als Projektion des Maus-Klickens
      nGrIdx = nIdx          'auf die Linie (Profilschnitt)
      if  (vPtx  = aMPtx) then
          Ptx = vPtx
          Pty = vPty
      elseif ((vPtx  <> aMPtx) and (aMPtx <> nPtx)) then
          if (vPty = nPty) then
                Ptx = aMPtx
                Pty = vPty
          elseif (vPty < nPty) then
                if (nPtx = vPtx) then
                     Ptx = nPtx
                     Pty = nPty
                elseif (nPtx <> vPtx) then
                     xnvDiff = nPtx - vPtx
                     ynvDiff = nPty - vPty
                     xMvDiff = aMPtx - vPtx
                     Ptx = aMPtx
                     Pty  = (ynvDiff / xnvDiff) * xMvDiff + vPty
                end
          elseif (vPty > nPty) then
                if (nPtx = vPtx) then
                     Ptx = nPtx
                     Pty = nPty
```

```
                elseif (nPtx <> vPtx) then
                    xnvDiff = nPtx - vPtx
                    yvnDiff = vPty - nPty
                    xnMDiff = nPtx - aMPtx
                    Ptx = aMPtx
                    Pty  = (yvnDiff / xnvDiff) * xnMDiff + nPty
                end
            end
        elseif (nPtx  = aMPtx)  then
            Ptx = nPtx
            Pty = nPty
        end
end             ' Ende von (if ((minDist = 0) or (minDist < 10)) then)
ListofGrPt.Add(Ptx@Pty)       'Ein Grenz-Punkt für den Abschnitt

ListofGrIdx = {}
ListofGrIdx.Add(vGrIdx)
ListofGrIdx.Add(nGrIdx)
ListofListofGrIdx.Add(ListofGrIdx)

end                         ' Ende der Schleife für Abschnitt
end                         ' Ende des großen Blockes (for-Schleife)

'Neuer Profilschnitt des Datensatzes wird durch Ersetzen des Abschnittes hergestellt.
ListofFLPt00 = ListofListofFLPt.Get(0)  ' Liste der Punkte im aktiven Profilschnitt
AnzFLPt00 = ListofFLPt00.Count          ' zur Korrektur des Abschnittes
IdxFLPt00 = AnzFLPt00 - 1
AnfFLPt00 = ListofGrPt.Get(0)
EndFLPt00 = ListofGrPt.Get(1)
aListofGrIdx = ListofListofGrIdx.Get(0)
vGrIdxAnf00 = aListofGrIdx.Get(0)
aListofGrIdx = ListofListofGrIdx.Get(1)
nGrIdxEnd00 = aListofGrIdx.Get(1)
minDistAnf00 = ListofminDist.Get(0)
minDistEnd00 = ListofminDist.Get(1)
minIdxAnf00 = ListofminDistIdx.Get(0)
minIdxEnd00 = ListofminDistIdx.Get(1)

ListofFLPt11 = ListofListofFLPt.Get(1) ' Liste der Punkte im 1.anderen Profilschnitt
AnzFLPt11 = ListofFLPt11.Count         ' zur Übernahme des Abschnittes
IdxFLPt11 = AnzFLPt11 - 1
AnfFLPt11 = ListofGrPt.Get(2)
EndFLPt11 = ListofGrPt.Get(3)
aListofGrIdx = ListofListofGrIdx.Get(2)
nGrIdxAnf11 = aListofGrIdx.Get(1)
aListofGrIdx = ListofListofGrIdx.Get(3)
vGrIdxEnd11 = aListofGrIdx.Get(0)

ListofFLPt22 = ListofListofFLPt.Get(2) ' Liste der Punkte im 2.anderen Profilschnitt
AnzFLPt22 = ListofFLPt22.Count         ' zur Übernahme des Abschnittes
IdxFLPt22 = AnzFLPt22 - 1
AnfFLPt22 = ListofGrPt.Get(4)
EndFLPt22 = ListofGrPt.Get(5)
aListofGrIdx = ListofListofGrIdx.Get(4)
nGrIdxAnf22 = aListofGrIdx.Get(1)
aListofGrIdx = ListofListofGrIdx.Get(5)
vGrIdxEnd22 = aListofGrIdx.Get(0)

'Berechnung der Mittelwerte von den beiden Listen
ListofVQHx = {}
ListofVQHy = {}
xAnfFLPt11 = AnfFLPt11.Getx
```

```
yAnfFLPt11 = AnfFLPt11.Gety
ListofVQHx.Add(xAnfFLPt11)
ListofVQHy.Add(yAnfFLPt11)

for each i in 0..IdxFLPt11
   if ((i >= nGrIdxAnf11) and (i <= vGrIdxEnd11)) then
      aPt = ListofFLPt11.Get(i)
      xkrd = aPt.Getx
      ykrd = aPt.Gety
      ListofVQHx.Add(xkrd)
      ListofVQHy.Add(ykrd)
   end
end
xEndFLPt11 = EndFLPt11.Getx
yEndFLPt11 = EndFLPt11.Gety
ListofVQHx.Add(xEndFLPt11)
ListofVQHy.Add(yEndFLPt11)

ListofNQHx = {}
ListofNQHy = {}
xAnfFLPt22 = AnfFLPt22.Getx
yAnfFLPt22 = AnfFLPt22.Gety

ListofNQHx.Add(xAnfFLPt22)
ListofNQHy.Add(yAnfFLPt22)

for each i in 0..IdxFLPt22
   if ((i >= nGrIdxAnf22) and (i <= vGrIdxEnd22)) then
      aPt = ListofFLPt22.Get(i)
      xkrd = aPt.Getx
      ykrd = aPt.Gety
      ListofNQHx.Add(xkrd)
      ListofNQHy.Add(ykrd)
   end
end
xEndFLPt22 = EndFLPt22.Getx
yEndFLPt22 = EndFLPt22.Gety

ListofNQHx.Add(xEndFLPt22)
ListofNQHy.Add(yEndFLPt22)

av.ShowMsg("erste Berechnung der Mittelwerte des Abschnittes ...")
av.ShowStopButton
QHVAnz=ListofVQHx.Count
QHVIndAnz=QHVAnz-1

QHNAnz=ListofNQHx.Count
QHNIndAnz=QHNAnz-1

ListofxMWV={}
ListofyMWV={}

for each pkte in 0..QHVIndAnz
   Nr = pkte + 1
   QHVxkrd=ListofVQHx.Get(pkte)
   QHVykrd=ListofVQHy.Get(pkte)
   for each ppkte in 0..QHNIndAnz
      Prfx1=ListofNQHx.Get(ppkte)
      Prfy1=ListofNQHy.Get(ppkte)
     if (ppkte < QHNIndAnz) then
       Prfx2=ListofNQHx.Get(ppkte+1)
       Prfy2=ListofNQHy.Get(ppkte+1)
```

```
        if (Prfx1 > Prfx2) then
          Prfgx=Prfx1
          Prfkx=Prfx2
        elseif (Prfx2 > Prfx1) then
          Prfgx=Prfx2
          Prfkx=Prfx1
        end
        if (Prfy1 > Prfy2) then
          Prfgy=Prfy1
          Prfky=Prfy2
        elseif (Prfy2 > Prfy1) then
          Prfgy=Prfy2
          Prfky=Prfy1
        end
        if ((QHVxkrd > Prfkx) and (QHVxkrd < Prfgx)) then
          QHNxkrd=QHVxkrd
          if (Prfy1 <> Prfy2) then
            if (Prfy2 > Prfy1) then
             QHNykrd=((((QHVxkrd-Prfkx).Abs)/((Prfgx-Prfkx).Abs))*((Prfgy-Prfky).Abs))+Prfky
            elseif (Prfy1 > Prfy2) then
             QHNykrd=((((Prfgx-QHVxkrd).Abs)/((Prfgx-Prfkx).Abs))*((Prfgy-Prfky).Abs))+Prfky
            end
          elseif (Prfy1 = Prfy2) then
            QHNykrd=Prfy1
          end
        elseif (Prfx1 = QHVxkrd) then
          QHNxkrd=Prfx1
          QHNykrd=Prfy1
        end
      elseif (ppkte = QHNIndAnz) then
        if (Prfx1 = QHVxkrd) then
          QHNxkrd=Prfx1
          QHNykrd=Prfy1
        end
      end
    end
    QHMxkrd=QHNxkrd
    QHMykrd=(QHVykrd+QHNykrd)/2
    ListofxMWV.Add(QHMxkrd)
    ListofyMWV.Add(QHMykrd)
    'Show percentage complete with enabled stop button
    more = av.SetStatus((Nr / (QHVAnz)) * 100)
    if (not more) then
      exit
    end
end
'MsgBox.ListAsString(ListofxMWV, "x-Koordinaten des Mittelwertes", "1. Mittelwert")
'MsgBox.ListAsString(ListofyMWV, "y-Koordinaten des Mittelwertes", "1. Mittelwert")

av.ShowMsg("zweite Berechnung der Mittelwerte des Abschnittes ...")
av.ShowStopButton

ListofxMWN={}
ListofyMWN={}

for each pkte in 0..QHNIndAnz
    Nr = pkte + 1
    QHNxkrd=ListofNQHx.Get(pkte)
    QHNykrd=ListofNQHy.Get(pkte)
    for each ppkte in 0..QHVIndAnz
       Prfx1=ListofVQHx.Get(ppkte)
       Prfy1=ListofVQHy.Get(ppkte)
```

```
      if (ppkte < QHVIndAnz) then
        Prfx2=ListofVQHx.Get(ppkte+1)
        Prfy2=ListofVQHy.Get(ppkte+1)
        if (Prfx1 > Prfx2) then
          Prfgx=Prfx1
          Prfkx=Prfx2
        elseif (Prfx2 > Prfx1) then
          Prfgx=Prfx2
          Prfkx=Prfx1
        end
        if (Prfy1 > Prfy2) then
          Prfgy=Prfy1
          Prfky=Prfy2
        elseif (Prfy2 > Prfy1) then
          Prfgy=Prfy2
          Prfky=Prfy1
        end
        if ((QHNxkrd > Prfkx) and (QHNxkrd < Prfgx)) then
          QHVxkrd=QHNxkrd
          if (Prfy1 <> Prfy2) then
            if (Prfy2 > Prfy1) then
             QHVykrd=((((QHNxkrd-Prfkx).Abs)/((Prfgx-Prfkx).Abs))*((Prfgy-Prfky).Abs))+Prfky
            elseif (Prfy1 > Prfy2) then
             QHVykrd=((((Prfgx-QHNxkrd).Abs)/((Prfgx-Prfkx).Abs))*((Prfgy-Prfky).Abs))+Prfky
            end
          elseif (Prfy1 = Prfy2) then
            QHVykrd=Prfy1
          end
        elseif (Prfx1 = QHNxkrd) then
          QHVxkrd=Prfx1
          QHVykrd=Prfy1
        end
      elseif (ppkte = QHVIndAnz) then
        if (Prfx1 = QHNxkrd) then
          QHVxkrd=Prfx1
          QHVykrd=Prfy1
        end
      end
    end
    QHMxkrd=QHNxkrd
    QHMykrd=(QHVykrd+QHNykrd)/2
    ListofxMWN.Add(QHMxkrd)
    ListofyMWN.Add(QHMykrd)
    'Show percentage complete with enabled stop button
    more = av.SetStatus((Nr / (QHNAnz)) * 100)
    if (not more) then
      exit
    end
  end
  'MsgBox.ListAsString(ListofxMWN, "x-Koordinaten des Mittelwertes", "2. Mittelwert")
  'MsgBox.ListAsString(ListofyMWN, "y-Koordinaten des Mittelwertes", "2. Mittelwert")

  'Aus beiden Listen wird eine List hergestellt.
  ListofNQHMWx={}
  ListofNQHMWy={}

  QHMWVAnz=ListofxMWV.Count
  QHMWNAnz=ListofxMWN.Count

  if ((QHMWVAnz > QHMWNAnz) or (QHMWVAnz = QHMWNAnz)) then
    QHMWVInd=QHMWVAnz-1
    for each aMTW in 0..QHMWVInd
```

```
      xMWVkrd=ListofxMWV.Get(aMTW)
      yMWVkrd=ListofyMWV.Get(aMTW)
      ListofNQHMWx.Add(xMWVkrd)
      ListofNQHMWy.Add(yMWVkrd)
   end
elseif (QHMWVAnz < QHMWNAnz) then
   QHMWNInd=QHMWNAnz-1
   for each aMTW in 0..QHMWNInd
      xMWNkrd=ListofxMWN.Get(aMTW)
      yMWNkrd=ListofyMWN.Get(aMTW)
      ListofNQHMWx.Add(xMWNkrd)
      ListofNQHMWy.Add(yMWNkrd)
   end
end

'MsgBox.ListAsString(ListofNQHMWx, "x-Koordinaten des Mittelwertes", "Liste zum Ersetzen")
'MsgBox.ListAsString(ListofNQHMWy, "y-Koordinaten des Mittelwertes", "Liste zum Ersetzen")

ListofNT = {}
for each j in 0..vGrIdxAnf00     'Punkte vor dem Abschnitt der Korrektur
     aFLPt = ListofFLPt00.Get(j)
     if (minDistAnf00 < 10) then
        if (minIdxAnf00 > 0) then
           ListofNT.Add(aFLPt)
        end
     elseif (minDistAnf00 >= 10) then
        ListofNT.Add(aFLPt)
     end
end

AnzNQHMWx = ListofNQHMWx.Count
IdxNQHMWx = AnzNQHMWx - 1

for each i in 0..IdxNQHMWx
   xkrd = ListofNQHMWx.Get(i)
   ykrd = ListofNQHMWy.Get(i)
   ListofNT.Add(xkrd@ykrd)          ' neue Punkte
end

for each j in 0..IdxFLPt00      'Punkte nach dem Abschnitt der Korrektur
     if (j >= nGrIdxEnd00) then
        aFLPt = ListofFLPt00.Get(j)
        if (minDistEnd00 < 10) then
           if (minIdxEnd00 < IdxFLPt00) then
             ListofNT.Add(aFLPt)
           end
        elseif (minDistEnd00 >= 10) then
           ListofNT.Add(aFLPt)
        end
     end
end

av.ShowMsg("Speicherung der neuen 2D-Profilschnitte in einem Thema ...")
ListofListofNPoint={}
ListofListofNPoint.Add(ListofNT)
thePolyLine=PolyLine.Make(ListofListofNPoint)
'MsgBox.report(thePolyLine.AsString, "the PolyLine")

theFTab = theTheme.GetFTab
theShpFld = theFTab.FindField("Shape")
RecNr = ListofRecIdx.Get(0)
theFTab.SetEditable(false)
```

```
theFTab.SetEditable(true)
theFTab.SetValue(theShpFld, recNr, thePolyLine)
theFTab.SetEditable(false)

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette
aListofColor={}
aListofNr={53, 5, 8, 20, 32, 26, 1}
for each Nmb in 0..5
  aNumb=aListofNr.Get(Nmb)
  theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
  theRgbList=theColor.GetRgbList
  aColor=Color.Make
  aColor.SetRgbList(theRgbList)
  aListofColor.Add(aColor)
end

theLegend=theTheme.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(theTheme, "Flaeche")
'theLegend.SetNullValue("Flaeche", "Null")
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofSColor={}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   if (theKlasseLb="GH") then
     theColorNr=aListofColor.Get(0)
   elseif (theKlasseLb="Na") then
     theColorNr=aListofColor.Get(1)
   elseif (theKlasseLb="NQ") then
     theColorNr=aListofColor.Get(2)
   else
     theColorNr=aListofColor.Get(3)
   end
   aListofSColor.Add(theColorNr)
end

aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
AnzSymbIdx=AnzSymb-2

for each symb in 0..AnzSymbIdx
   aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
end
theTheme.UpdateLegend




'pfbalk1s.ave
'Eine geologische Schicht wird als Balken in einem Profilschnitt-View
'gezeichnet. Die Daten bestehen aus Punkten in einem Punkt-Thema
'mit Höhen der Ober- und Unterkanten. Dieses kann auch ein Ereignis-Thema
'in dem Profilschnitt-View sein.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
```

```
myScript=theProject.FindScript("pfbalk1s")
myScript.SetNumberFormat( "d.dd") ' script default
theView = av.GetActiveDoc  'ein aktives Profilschnitt-View

thePtThm = theView.GetActiveThemes.Get(0)
ThStr = thePtThm.AsString

kt00=MsgBox.YesNo("Ist das aktive Thema"++ThStr
     +NL+"zur Herstellung der Schicht-Balken richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
            "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

PtFTab = thePtThm.GetFTab          'Tabelle für das Punkt-Thema
ListofPtFlds = PtFTab.GetFields    'Liste der Überschrift der Tabelle

PtShpFld = ListofPtFlds.Get(0)
PtOkFld=MsgBox.ChoiceAsString(ListofPtFlds,
      "Feld für Oberkante der Schicht [m]","Eingabe vom Feld in"++ThStr)
PtUkFld=MsgBox.ChoiceAsString(ListofPtFlds,
      "Feld für Unterkante der Schicht [m]","Eingabe vom Feld in"++ThStr)
PtSchtFld=MsgBox.ChoiceAsString(ListofPtFlds,
      "Feld für ein Kennwort der Schicht", "Eingabe vom Feld in"++ThStr)
theScht = PtFTab.ReturnValue(PtSchtFld, 0)

YFaktorStr = MsgBox.Input("zur Überhöhung der Höhen im Profil",
           "Eingabe eines Faktors", "50")
YFkt = YFaktorStr.AsNumber

AnzPt=0
for each rec in PtFTab
   AnzPt=AnzPt+1
end
AnzPtIdx=AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"++ThStr)

av.ShowMsg("Die Balken der geologischen Schicht werden als Polyline"
       ++"aus der Tabelle"++ThStr++"hergestellt"++"...")
av.ShowStopButton

'Ein Feature-Shape-File (PolyLine) für einen Profilschnitt wird hergestellt.
ListofPL={}
Ng=0
for each i in 0..AnzPtIdx
   Ng=Ng+1
   xkrd = (PtFTab.ReturnValue(PtShpFld, i)).Getx
   theOk = PtFTab.ReturnValue(PtOkFld, i)
   theUk = PtFTab.ReturnValue(PtUkFld, i)

   ykrdOk = theOk * YFkt
   ykrdUk = theUk * YFkt
   ListofPt = {}
   ListofPt.Add(xkrd@ykrdOk)
   ListofPt.Add(xkrd@ykrdUk)
   ListofListofPoint={}
   ListofListofPoint.Add(ListofPt)
   thePolyLine=PolyLine.Make(ListofListofPoint)
   ListofPL.Add(thePolyLine)
```

```
    'Show percentage complete with enabled stop button
    more=av.SetStatus((Ng/(AnzPt))*100)
    if (not more) then
      break
    end
  end

'Wenn ein Thema im aktiven View bei der Bearbeitung ist,
'wird die Bearbeitung vor der Herstellung eines neuen Themas beendet.

editThm = theView.GetEditableTheme
if (editThm <> nil) then
  doSave = MsgBox.YesNoCancel("Save edits to "+editThm.GetName+"?",
        "Stop Editing",true)
  if (doSave = nil) then
    return nil
  end
  if (editThm.StopEditing(doSave).Not) then
     MsgBox.Info("Unable to Save Edits to "
               + editThm.GetName +
               ", please use the Save Edits As option", "")
     return nil
  else
     theView.SetEditableTheme(NIL)
  end
end

ListofSp11 = {"Herstellung als ein neues Thema",
            "Speicherung im vorhandenen Thema"}

AW11 = MsgBox.ListAsString(ListofSp11, "zur Speicherung der Balken"
  +NL+"für die geologische Schicht"++theScht,"Auswahl eines Themas")

if (AW11 = "Herstellung als ein neues Thema") then
  WDStr=av.GetProject.GetWorkDir.AsString
  fn00 = ("Bk"+(theScht.LCase)).Left(6)
  fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
  'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
  if (fName =nil) then exit end
  fName.SetExtension("shp")

  BkFTab = FTab.MakeNew(fName, PolyLine)
  if (BkFTab.HasError) then
    if (BkFTab.HasLockError) then
      MsgBox.Error("Unable to acquire Write Lock for file "
              + fName.GetBaseName, "")
    else
      MsgBox.Error("Unable to create " + fName.GetBaseName, "")
    end
    exit
  end

  BkIDfld = Field.Make("ID", #FIELD_DECIMAL, 8, 0)
  BkSNfld = Field.Make("Kennwort", #FIELD_Char, 8, 0)
  BkIDfld.SetVisible( TRUE )
  BkSNfld.SetVisible( TRUE )
  BkFTab.AddFields({BkIDfld,BkSNfld})
  BkShpFld = BkFTab.FindField("Shape")
  BkThm = FTheme.Make(BkFTab)
  theView.AddTheme(BkThm)
  recNr = -1
```

```
elseif (AW11 = "Speicherung im vorhandenen Thema") then

  ListofThms = theView.GetThemes
  ListofPLThms = {}

  for each aT in ListofThms
    if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aCNm = aFTab.GetShapeClass.GetClassName
      if (aCNm = "Polyline") then
        ListofPLThms.Add(aT)
      end
    end
  end

  'ein Thema für die neuen Balken

  BkThm = MsgBox.ChoiceAsString(ListofPLThms,
        "um die Balken für"++theScht++"zu speichern"+NL+
        "Der Name eines Polyline-Themas in View"++theView.AsString,
        "Auswahl eines Themas")

  BkFTab = BkThm.GetFTab
  BkShpFld = BkFTab.FindField("Shape")
  BkIDfld = BkFTab.FindField("ID")
  BkSNfld = BkFTab.FindField("Kennwort")

  'Feststellung der Anzahl der in dem Thema schon vorhandenen Balken
  AnzBk = 0
  for each rec in BkFTab
    AnzBk = AnzBk + 1
  end
  IdxBk = AnzBk - 1
  recNr = IdxBk
  'MsgBox.Info(AnzBk.AsString, "die Anzahl der Datensätze im Thema"
  '       ++BkThm.AsString)
end

AnzNBk = ListofPL.Count    'Anzahl der neuen Balken
IdxNBk = AnzNBk - 1

BkFTab.SetEditable(False)
BkFTab.SetEditable(true)

for each i in 0..IdxNBk
   thePolyLine = ListofPL.Get(i)
   recNr = recNr + 1
   BkFTab.AddRecord
   BkFTab.SetValue(BkShpFld, recNr, thePolyLine)
   BkFTab.SetValue(BkIDfld, recNr, recNr)
   BkFTab.SetValue(BkSNfld, recNr, theScht)
end

BkFTab.SetEditable(false)

BkThm.SetActive(TRUE)
BkThm.SetVisible(TRUE)
theProject.SetModified(true)

theLegend = BkThm.GetLegend
```

```
theLegend.SetLegendType(#LEGEND_TYPE_SIMPLE)
theLegend.SingleSymbol
theLegend.GetSymbols.Get(0).SetColor(Color.GetBlue)
BkThm.UpdateLegend



'pfbalkhg.ave
'Bordaten werden aus einer dBase-Datenbank-Datei auf einer Festplatte
'eingelesen und als unterschiedliche geologische Balken in einem aktiven
'Längsprofilschnitt-View gezeichnet.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc

'Eingabe einer dBase-Tabelle auf der Festplatte
theFileName=FileDialog.Show("*.dbf", "dBase-File als eine Datenbank",
          "Eingabe einer Datei für Bohrungen")
if (theFileName = nil) then exit end

theVtab=Vtab.Make(theFileName,false,false)

'Eine Tabelle gleich wie die dBase-Tabelle auf der Festplatte wird im Project
'für spätere Aufgabe hergestellt.
'Bg_Tab=Table.Make(theVtab)

'Feststellung der Anzahl der Datensätze in der Tabelle
AnzRec=0
for each rec in theVtab
   AnzRec=AnzRec+1
end
idxAnzRec=AnzRec-1
'MsgBox.Info("Anzahl der Datensätze in der Tabelle: "
'         ++(AnzRec).AsString, "Kontrolle der Daten")

YFtStr=MsgBox.Input("zur Überhöhung der Höhe",  "Eingabe eines Faktors", "50")
YFt=YFtStr.AsNumber

FldList=theVtab.GetFields
AnzFld=FldList.Count
idxAnzFld=AnzFld-1
'MsgBox.Info("Anzahl der Felder in der Tabelle:"
'      ++AnzFld.AsString,"Kontrolle der Daten")
'MsgBox.ListAsString(FldList,  "Name der Felder", "Kontrolle der Daten")
theVtab.SetEditable(false)
ListofBNr={}
ListofPShapes={}
ListofRW={}
ListofHW={}
ListofPetro={}
ListofRecEndStr={}
NRFld=theVtab.FindField("Nr")
RWFld=theVtab.FindField("Rw")
HWFld=theVtab.FindField("Hw")
GHFld=MsgBox.ListAsString(FldList, "für Geländehöhe [m]",
     "Auswahl eines Feldes im"++theVtab.AsString)

for each aRec in theVTab
   aBNr=theVtab.ReturnValue(NRFld,aRec)
```

```
   aRW=theVtab.ReturnValue(RWFld,aRec)
   aHW=theVtab.ReturnValue(HWFld,aRec)
   aGh=theVtab.ReturnValue(GHFld,aRec)
   aPt=theVtab.ReturnValue((FldList.Get(4)),aRec)
   ListofBNr.Add(aBNr)
   aGh50=aGh* YFt
   ListofPShapes.Add(aHW@aGh50)
   ListofRW.Add(aRW)
   ListofHW.Add(aHW)
   ListofPetro.Add(aPt)
   ListofRecEndStr.Add("")
   SW="aus"
   for each aFld in 0..idxAnzFld
      if (aFld > 4) then
         if (FldList.Get(aFld).IsTypeNumber) then
            aTiefe=theVtab.ReturnValue((FldList.Get(aFld)),aRec)
            if (aTiefe <> 0) then
               aHoehe=aGh-aTiefe
               aHoe50=aHoehe* YFt
               ListofBNr.Add(aBNr)
               ListofPShapes.Add(aHW@aHoe50)
               ListofRW.Add(aRW)
               ListofHW.Add(aHW)
               SW="an"
            elseif (aTiefe = 0) then
               SW="aus"
            end
         elseif (FldList.Get(aFld).IsTypeString) then
            if (SW = "an") then
              aPetro=theVtab.ReturnValue((FldList.Get(aFld)),aRec)
              ListofPetro.Add(aPetro)
              ListofRecEndStr.Add("")
            end
         end
      end
   end
   idxStr=ListofRecEndStr.Count-1
   ListofRecEndStr.Remove(idxStr)
   ListofRecEndStr.Add("Ende")
end

av.ShowMsg("Herstellung der neuen Schichten-Balken ...")
av.ShowStopButton
AnzPoint=ListofBNr.Count
AnzPShps=ListofPShapes.Count
AnzHW=ListofHW.Count
AnzPtr=ListofPetro.Count
AnzREST=ListofRecEndStr.Count
'MsgBox.Report("Anzahl der Punkte:"+NL+NL+
'             "Bohr-Nr:"++"   "++AnzPoint.AsString+NL+
'             "Pt-Shapes:"++" "++AnzPShps.AsString+NL+
'             "Hochwert:"++"  "++AnzHW.AsString+NL+
'             "Petrogr:"++"   "++AnzPtr.AsString+NL+
'             "RecEnd:"++"    "++AnzREST.AsString,
'             "Kontrolle")

'Herstellung der Schichten-Balken
IdxPoint=AnzPoint-1
Ng=0
ListofBkShps={}
ListofPtKrdLb={}
ListofyKrdLb={}
```

```
for each aPt in 0..IdxPoint
  Ng=Ng+1
  idxNr=aPt
  thePt=ListofPShapes.Get(aPt)
  xKrd=thePt.Getx
  y1Krd=thePt.Gety
  ListofPolyL={}
  ListofPolyL.Add(thePt)
  RecEndStr=ListofRecEndStr.Get(aPt)
  if (RecEndStr <> "Ende") then
    aPt2=idxNr+1
    thePt2=ListofPShapes.Get(aPt2)
    y2Krd=thePt2.Gety
    if (y2Krd < 380) then
      y2Krd=380.00
    end
    yKrd=(y1Krd+y2Krd)/2
  elseif (RecEndStr = "Ende") then
    y2Krd=380.00
    yKrd=y1Krd/2
  end
  ListofPolyL.Add(xKrd@y2Krd)
  ListofListofPL={}
  ListofListofPL.Add(ListofPolyL)
  theBkShp=PolyLine.Make(ListofListofPL)
  ListofBkShps.Add(theBkShp)
  ListofPtKrdLb.Add(xKrd@yKrd)
  ListofyKrdLb.Add(yKrd)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzPoint*100)
  if (not more) then
    break
  end
end
AnzBNrBk=ListofBNr.Count
AnzBkShps=ListofBkShps.Count
AnzBkHW=ListofHW.Count
AnzPtrBk=ListofPetro.Count
AnzPtKrdLb=ListofPtKrdLb.Count
AnzyKrdLb=ListofyKrdLb.Count

'MsgBox.Report("Anzahl der Balken:"+NL+NL+
'             "Bohr-Nr:"++"     "++AnzBNrBk.AsString+NL+
'             "Bk-Shapes:"++"   "++AnzBkShps.AsString+NL+
'             "Hochwert:"++"    "++AnzBkHW.AsString+NL+
'             "Petrogr:"++"     "++AnzPtrBk.AsString+NL+
'             "x,y-Koord.:"++"    "++AnzPtKrdLb.AsString+NL+
'             "y-Koord.:"++"    "++AnzyKrdLb.AsString,
'             "Kontrolle")

'Ein Feature-Shape-File für Profile (Polyline) wird hergestellt
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("Bkmtrgz1","shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("Bkmtrgz1","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PolyLine)")
if (fName =nil) then exit end
fName.SetExtension("shp")
FLFTab=FTab.MakeNew(fName, Polyline)
ShapeField1= FLFTab.FindField("shape")
IDField1=Field.Make("ID", #Field_Short, 4, 0)
BNrField1=Field.Make("Bohr-Nr", #Field_Byte, 3, 0)
RWField1=Field.Make("RW", #Field_FLOAT, 10, 2)
```

```
HWField1=Field.Make("HW", #Field_FLOAT, 10, 2)
PtField1=Field.Make("Petro", #Field_Char, 5, 0)
yLbField1=Field.Make("Lb_Hoehe", #Field_FLOAT, 8, 2)
ListofFlds1={IDField1,BNrField1,RWField1,HWField1,PtField1,yLbField1}
FLFTab.AddFields(ListofFlds1)

av.ShowMsg("Herstellung des neuen Polyline-Themas"
      ++"für Schichten-Balken ...")
av.ShowStopButton
FLFTab.SetEditable(false)
FLFTab.SetEditable(true)
IdxBk=AnzBkShps-1
Ng=0
recNr=-1

for each aBk in 0..IdxBk
  Ng=Ng+1
  theBk=ListofBkShps.Get(aBk)
  theId=aBk
  theBNr=ListofBNr.Get(aBk)
  theRW=ListofRW.Get(aBk)
  theHW=ListofHW.Get(aBk)
  thePtr=ListofPetro.Get(aBk)
  theyLb=ListofyKrdLb.Get(aBk)
  recNr=recNr+1
  FLFTab.AddRecord
  FLFTab.SetValue(ShapeField1, recNr, theBk)
  FLFTab.SetValue(IDField1, recNr, theId)
  FLFTab.SetValue(BNrField1, recNr, theBNr)
  FLFTab.SetValue(RWField1, recNr, theRW)
  FLFTab.SetValue(HWField1, recNr, theHW)
  FLFTab.SetValue(PtField1, recNr, thePtr)
  FLFTab.SetValue(yLbField1, recNr, theyLb)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzBkShps*100)
  if (not more) then
    break
  end
end
FLFTab.SetEditable(false)
thmNew=FTheme.Make(FLFTab)
theView.AddTheme(thmNew)

'Veränderung der Legende des Themas im Profil
'Definition der Farbe für die Balken
ListofFarbD = {"Mu", 53, "Wb", 53, "aufB", 53, "Lehm", 47, "Ls", 47,
   "Ubr", 47, "Uff", 47, "Uf", 47, "Ut", 47, "Ufs", 47, "Ufst", 47,
   "USt", 47, "Ust", 47, "US", 47, "Bk", 11, "BkB", 11, "Bkt", 11,
   "BkT", 11, "Bkhz", 11, "TBks", 11, "TBK", 11, "Ubks", 11,
   "Tdbr", 11, "Ton", 20, "Tgr", 20, "Tbr", 20, "Tbk", 20, "Tons", 20,
   "Ts", 20, "Tsgr", 20, "TonS", 20, "sT", 20, "TU", 20, "Tufs", 20,
   "Tus", 20, "Tsu", 20, "TSG", 20, "Tsst", 20, "USmS", 17, "SMu", 17,
   "Sd", 17, "Sand", 17, "Sbr", 17, "Sgr", 17, "ST", 17, "St", 17,
   "STla", 17, "St'", 17, "St!", 17, "Su", 17, "Sut", 17, "Sut'", 17,
   "Sl", 17, "Sg'", 17, "SmTv", 17, "Smgs", 17, "LUS", 17, "fS", 17,
   "tfS", 17, "fSgr", 17, "fSt", 17, "fSt!", 17, "fST", 17, "fSl", 17,
   "fSu'", 17, "fSu", 17, "fSU", 17, "fSut", 17, "FSut", 17, "fSmS", 17,
   "fSms", 17, "fmS", 17, "fmSt", 17, "mS", 17, "mSbr", 17, "mSgr", 17,
   "mSra", 17, "mSs", 17, "mSl", 17, "mfS", 17, "mfSU", 17, "mfSu", 17,
   "mfST", 17, "mST", 17, "mSt", 17, "mSTe", 17, "mSu'", 17, "mSL", 17,
   "mSl", 17, "mSfs", 17, "mSgs", 17, "mSgl", 17, "fmgS", 17, "mgS", 17,
   "mgST", 17, "mgSU", 17, "tgS", 17, "gS", 17, "gSbr", 17, "gSfs", 17,
```

```
  "gST", 17, "gST'", 17, "gSU'", 17, "gmS", 17, "Tgr", 4, "TTr", 4,
  "Tgel", 4, "Tr", 4, "Tr?", 4, "fSTr", 4, "mSTr", 4, "KA", 1,
  "K.A.", 1, "", 1}

theTheme=thmNew
theLegend=theTheme.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(theTheme, "Petro")
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofNr={}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   aLbIdx = ListofFarbD.FindByValue(theKlasseLb)
   if (aLbIdx <> -1) then
     aCNrIdx = ListofFarbD.Get(aLbIdx + 1)
   elseif (aLbIdx = -1) then
     aRNr = i Mod 60
     if (aRNr = 0) then
       aCNrIdx = 1
     elseif (aRNr <> 0) then
       aCNrIdx = aRNr
     end
   end
   aListofNr.Add(aCNrIdx)
end

aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
'MsgBox.Info(AnzSymb.AsString, "Anzahl der Symbole")
AnzSymbIdx=AnzSymb-2
aListofColor={}

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

for each Nmb in 0..IdxKlasse
  aNumb=aListofNr.Get(Nmb)
  theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
  theRgbList=theColor.GetRgbList
  aColor=Color.Make
  aColor.SetRgbList(theRgbList)
  aListofColor.Add(aColor)
end

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_Pen)
thePPalette=av.GetSymbolWin.GetPalette
theBasicPen=(thePPalette.GetList(#PALETTE_LIST_Pen).Get(0))

for each symb in 0..AnzSymbIdx
   theSymbol=aListofSymbol.Get(symb)
   theSymbol.Copy(theBasicPen)
   theSymbol.SetColor(aListofColor.Get(symb))
   theSymbol.SetSize(0.5)
end

theTheme.UpdateLegend
theRecNr=ListofPtKrdLb.Count
IdxRecNr=theRecNr-1
```

```
theGraphicList=theView.GetGraphics

for each aRecNr in 0..IdxRecNr
  theGString=ListofPetro.Get(aRecNr)
  if ((theGString = "") or (theGString = "KA")) then
    Continue
  elseif (theGString = "K.A.") then
    Continue
  elseif (theGString <> "") then
    theGPoint=ListofPtKrdLb.Get(aRecNr)
    theGText=GraphicText.Make(theGString, theGPoint)
    theGText.SetAlignment(#TEXTCOMPOSER_JUST_CENTER)

    theTextSymbol=theGText.ReturnSymbols.Get(0)
    theTextSymbol.SetSize(10)

    newFont=Font.Make("Arial", "normal")
    theTextSymbol.SetFont(newFont)
    theTextSymbol.SetColor(Color.GetBlack)
    theGText.Invalidate
    theGraphicList.Add(theGText)
  end
end



'pfbboent.ave
'Bohrungen, die innerhalb eines Abstandes von einem beliebigen Profilschnitt
'ausgewählt worden sind, werden auf den Profilschnitt senkrecht projiziert,
'um sie in einem Profilschnitt-View mit dem Profilschnitt zusammen zu zeichnen.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc         'Aktives Karten-View

'Eingabe der Datei, die ausgewählte Bohrdaten enthält,
'zur Bestimmung der Lage auf einer beliebigen Profilschnittlinie

ListofThms=theView.GetThemes
ListofFThm = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
       ListofFThm.Add(aT)
     end
   end
end
BohrThm=MsgBox.ChoiceAsString(ListofFThm,
      "das die Punkte der Bohrdaten enthält",
      "Auswahl eines Themas im View"++theView.AsString)

BohrFTab=BohrThm.GetFTab
ListofBoFTabFlds=BohrFTab.GetFields

BoShpFld=ListofBoFTabFlds.Get(0)
BoEntfFld=MsgBox.ChoiceAsString(ListofBoFTabFlds,
       "für Entfernung der Bohrungen"+NL+
       "vom Anfang der Profilschnittlinie",
       "Auswahl eines Feldes in Thema:"++BohrThm.AsString)
AbstFld=MsgBox.ChoiceAsString(ListofBoFTabFlds,
```

```
  "für Abstand zwischen den Bohrungen"+NL+"und der Profilschnittlinie",
  "Auswahl eines Feldes im Thema:"++BohrThm.AsString)

'Feststellung der Anzahl der Bohrungen in dem Thema BohrThm
AnzBoPt=0
for each rec in BohrFTab
   AnzBoPt=AnzBoPt+1
end
IdxBoPt=AnzBoPt-1
MsgBox.Info(AnzBoPt.AsString, "Anzahl der Bohrungen in dem Thema"
      ++BohrThm.AsString)

ListofLg={"auf einer Profilschnittlinie", "außerhalb einer Profilschnittlinie",
        "noch nicht festgestellt"}
Qt1=MsgBox.ChoiceAsString(ListofLg, "die Lagen der Bohrungen",
  "Feststellung der Lage der Bohrungen")
if (Qt1 = "noch nicht festgestellt") then
  MsgBox.Report("Die Lage der Bohrungen muss"+NL+
        "noch festgestellt werden."+NL+
        "Das Programm wird jetzt unterbrochen!", "Information")
  Exit
end

'Eingabe der Datei, die alle Punkte
'auf einer beliebigen Profilschnittlinie enthält.

PTTheme=MsgBox.ChoiceAsString(ListofFThm,
       "das alle Punkte auf einer"+NL+
       "Profilschnittlinie enthält", "Eingabe eines Themas")
PTFTab=PTTheme.GetFTab
ListofFlds=PTFTab.GetFields

PTShpFld=MsgBox.ChoiceAsString(ListofFlds, "Eingabe des Feldes für Shape",
       "Auswahl eines Feldes in Thema:"++PTTheme.AsString)
PTEfFld=MsgBox.ChoiceAsString(ListofFlds,
       "Eingabe des Feldes für Entfernungen vom Anfang",
       "Auswahl eines Feldes im Thema:"++PTTheme.AsString)
ListofPtFld={PTShpFld, PTEfFld}

'Feststellung der Anzahl der Punkte in dem Thema PTTheme
AnzPT=0
for each rec in PTFTab
   AnzPT=AnzPT+1
end
IdxPT=AnzPT-1
MsgBox.Info(AnzPT.AsString, "Anzahl der Punkte in dem Thema"++PTTheme.AsString)

av.ShowMsg("Bestimmung der Lage der ausgewählten Bohrdaten"
       ++"auf einer Profilschnittlinie ...")
av.ShowStopButton

ListofNEntf={}
ListofAbstPrf={}

for each gzd in 0..IdxBoPt
  Ng=gzd+1
  theBo=BohrFTab.ReturnValue(BoShpFld, gzd) 'ein Punkt für eine Bohrung

  AnfX=theBo.Getx
  AnfY=theBo.Gety
  minAbstA=100000
```

```
'Suche nach dem Punkt auf der Profilschnittlinie im Punkt-Thema
'mit dem kleinsten Abstand von einer Bohrung
for each i in 0..IdxPT
    thePT=PTFTab.ReturnValue(PTShpFld, i) 'ein Punkt auf dem Profilschnittlinie
    theX=thePT.Getx
    theY=thePT.Gety
    DifXA2=((theX-AnfX)*(theX-AnfX))
    DifYA2=((theY-AnfY)*(theY-AnfY))
    AbstA=(DifXA2+DifYA2).Sqrt.Abs
    if (AbstA < minAbstA) then
      minAbstA=AbstA    'der kleinste Abstand zwischen einer Bohrung und
      IdxPTA=i          'einem Punkt auf der Profilschnittlinie
    end
  end

  'Berechnung der Entfernung der Punkte der auf der Profilschnittlinie projizierten
  'Bohrdaten vom Anfang der Profilschnittlinie
  if (minAbstA <> 0) then
    VIdxPTA=IdxPTA-1
    NIdxPTA=IdxPTA+1

    thePTVA=PTFTab.ReturnValue(PTShpFld, VIdxPTA)
    thePTNA=PTFTab.ReturnValue(PTShpFld, NIdxPTA)
    thePTIA=PTFTab.ReturnValue(PTShpFld, IdxPTA) 'der Punkt mit dem kleinsten Abstand

    thePTVAX=thePTVA.Getx
    thePTVAY=thePTVA.Gety

    thePTNAX=thePTNA.Getx
    thePTNAY=thePTNA.Gety

    thePTIAX=thePTIA.Getx
    thePTIAY=thePTIA.Gety

    DifXVA2=((thePTVAX-AnfX)*(thePTVAX-AnfX))
    DifYVA2=((thePTVAY-AnfY)*(thePTVAY-AnfY))
    AbstVA=(DifXVA2+DifYVA2).Sqrt.Abs

    DifXNA2=((thePTNAX-AnfX)*(thePTNAX-AnfX))
    DifYNA2=((thePTNAY-AnfY)*(thePTNAY-AnfY))
    AbstNA=(DifXNA2+DifYNA2).Sqrt.Abs

    DifXVIA=((thePTVAX-thePTIAX)*(thePTVAX-thePTIAX))
    DifYVIA=((thePTVAY-thePTIAY)*(thePTVAY-thePTIAY))
    AbstVIA=(DifXVIA+DifYVIA).Sqrt.Abs

    DifXNIA=((thePTNAX-thePTIAX)*(thePTNAX-thePTIAX))
    DifYNIA=((thePTNAY-thePTIAY)*(thePTNAY-thePTIAY))
    AbstNIA=(DifXNIA+DifYNIA).Sqrt.Abs

    if ((AbstVA < AbstVIA) and (AbstNA > AbstNIA)) then
      VIdxPTA=IdxPTA-1
      NIdxPTA=IdxPTA
    elseif ((AbstVA > AbstVIA) and (AbstNA < AbstNIA)) then
      VIdxPTA=IdxPTA
      NIdxPTA=IdxPTA+1
    end

    thePTVA=PTFTab.ReturnValue(PTShpFld, VIdxPTA)
    thePTNA=PTFTab.ReturnValue(PTShpFld, NIdxPTA)

    thePTVAX=thePTVA.Getx
```

```
    thePTVAY=thePTVA.Gety

    thePTNAX=thePTNA.Getx
    thePTNAY=thePTNA.Gety

    if(Qt1 = "auf einer Profilschnittlinie") then
      DifXVA2=((thePTVAX-AnfX)*(thePTVAX-AnfX))
      DifYVA2=((thePTVAY-AnfY)*(thePTVAY-AnfY))
      AbstPrf=0
    elseif(Qt1 = "außerhalb einer Profilschnittlinie") then
      if (thePTVAX <> thePTNAX) then
        if (thePTVAY <> thePTNAY) then
          Neiga=(thePTNAY-thePTVAY)/(thePTNAX-thePTVAX)
          Achsb=thePTVAY-(Neiga*thePTVAX)
          Neigag=(-1)/Neiga
          Achsbg=AnfY-(Neigag*AnfX)
          BPtprjx=(Achsbg-Achsb)/(Neiga-Neigag)
          BPtprjy=(Neiga*BPtprjx)+Achsb
        elseif (thePTVAY = thePTNAY) then
          BPtprjx=AnfX
          BPtprjy=thePTVAY
        end
      elseif (thePTVAX = thePTNAX) then
        BPtprjx=thePTVAX
        BPtprjy=AnfY
      end
      DifXVA2=((thePTVAX-BPtprjx)*(thePTVAX-BPtprjx))
      DifYVA2=((thePTVAY-BPtprjy)*(thePTVAY-BPtprjy))
      Difprjx2=((AnfX-BPtprjx)*(AnfX-BPtprjx))
      Difprjy2=((Anfy-BPtprjy)*(Anfy-BPtprjy))
      AbstPrf=(Difprjx2+Difprjy2).Sqrt.Abs
    end
    AbstVA=(DifXVA2+DifYVA2).Sqrt.Abs
    EntfVA=PTFTab.ReturnValue(PTEfFld, VIdxPTA)
    EntfPTA=EntfVA+AbstVA

  elseif (minAbstA = 0) then
    EntfPTA=PTFTab.ReturnValue(PTEfFld, IdxPTA)
    AbstPrf=0
  end

  ListofNEntf.Add(EntfPTA) 'Entfernungen der Bohrdaten auf der Profilschnittlinie
  ListofAbstPrf.Add(AbstPrf) 'zur Zeichnung der Bohrungen in einem Profilschnitt-View

  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzBoPt*100)
  if (not more) then
     break
  end
end

av.ShowMsg("Speicherung der Entfernung der auf der Profilschnittlinie projizierten"
        ++"Bohrdaten vom Anfang der Profilschnittlinie ...")
av.ShowStopButton

BohrFTab.SetEditable(false)
BohrFTab.SetEditable(true)

for each aPt in 0..IdxBoPt
  Ng=aPt+1
  theNEnf=ListofNEntf.Get(aPt)
  AbstPrf=ListofAbstPrf.Get(aPt)
```

```
  BohrFTab.SetValue(BoEntfFld, aPt, theNEnf)
  BohrFTab.SetValue(AbstFld, aPt, AbstPrf)

  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzBoPt*100)
  if (not more) then
    break
  end
end

BohrFTab.SetEditable(false)
BohrThm.UpdateLegend




'pfbbopg1.ave
'Die Polygone für Bohrungen, die auf einem beliebigen Profilschnittlinie
'projiziert worden sind, werden bei einem beliebigen Profilschnitt gezeichnet.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
'Profilschnitt-View, wo Polygone für ausgewälte Bohrungen gezeichnet werden.
PfView=av.GetActiveDoc

ListofKtViews={"Kt1_gg", "Kt1_ggd", "Kt1_hg", "Kt1_hgd", "Kt1_tga",
               "Kt1_tgd","Kt2_gg", "Kt2_ggd", "Kt2_hg", "Kt2_hgd",
               "Kt2_tga", "Kt2_tgd"}

KtViewStr = MsgBox.ChoiceAsString(ListofKtViews,
     "wo sich ein Thema für die ausgewählen Bohrungen"
     ++"mit den Entfernungen vom Anfang befindet",
     "Auswahl eines Views")
KtView = theProject.FindDoc(KtViewStr)

ListofThms = KtView.GetThemes
ListofFThm = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
       ListofFThm.Add(aT)
     end
   end
end

PTTheme = MsgBox.ChoiceAsString(ListofFThm, "Datei mit Bohrungsdaten",
     "Auswahl eines Themas im View"++KtView.AsString)

PTFTab=PTTheme.GetFTab
PTFields=PTFTab.GetFields
PTShpFld=MsgBox.ChoiceAsString(PTFields, "Feld für Punkt-Shape",
                        "Eingabe vom Feld in"++PTTheme.AsString)
PTEfFld=MsgBox.ChoiceAsString(PTFields, "Feld für Entfernung vom Anfang"+NL+
                              "der Profilschnittlinie",
                        "Eingabe vom Feld in"++PTTheme.AsString)
PTGHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Geländehöhe",
                        "Eingabe vom Feld in"++PTTheme.AsString)
PTTHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Höhe der Terrassenoberkante",
                        "Eingabe vom Feld in"++PTTheme.AsString)
PTQHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Quartärbasishöhe",
```

```
                    "Eingabe vom Feld in"++PTTheme.AsString)
PTEHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Bohrungsendhöhe",
                    "Eingabe vom Feld in"++PTTheme.AsString)
PTAbstFld=MsgBox.ChoiceAsString(PTFields, "Feld für Abstand zwischen"
                      +NL+"Bohrungen und Profilschnittlinie",
                      "Eingabe vom Feld in"++PTTheme.AsString)

ListofPTFld={PTShpFld, PTEfFld, PTGHFld, PTTHFld, PTQHFld, PTEHFld, PTAbstFld}

'Feststellung der Anzahl der ausgewählten Bohrungen
AnzPT=0
for each rec in PTFTab
  AnzPT=AnzPT+1
end
IdxPT=AnzPT-1


ListofPTFld={PTShpFld, PTEfFld, PTGHFld, PTTHFld, PTQHFld, PTEHFld, PTAbstFld}
WDStr=theProject.GetWorkDir.AsString

fnStr=FileName.Make(WDStr).MakeTmp("Bpfpgbo1","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Polygon)")
if (fName=nil) then exit end
fName.SetExtension("shp")

FTabBPg=FTab.MakeNew(fName, Polygon)
ShapeFldpg=FTabBPg.FindField("shape")
IDFldpg=Field.Make("ID", #Field_Short, 5, 0)
PtIDFldPg=Field.Make("Bohr_ID", #Field_Short, 4, 0)
RWFldpg=Field.Make("RW", #Field_Float, 10, 2)
HWFldpg=Field.Make("HW", #Field_Float, 10, 2)
EntfFldPg=Field.Make("Entfernung", #Field_Float, 10, 2)
AbstFldpg=Field.Make("AbstB_Prf", #Field_Float, 8, 2)
ALbFldPg=Field.Make("AB_Prf-Lb", #Field_Char, 4, 0)
SchichtFldpg=Field.Make("Schicht", #Field_Char, 10, 0)

ListofB1={IDFldpg, PtIDFldPg, RWFldpg, HWFldpg, EntfFldPg,
       AbstFldpg, ALbFldPg, SchichtFldpg}
FTabBPg.AddFields(ListofB1)
ListofB2={ShapeFldpg, IDFldPg, PtIDFldPg, RWFldpg, HWFldpg,
       EntfFldPg, AbstFldpg, ALbFldPg, SchichtFldpg}

'Feature-Shape-File für Bohrungen (Polygon) wird hergestellt.

av.ShowMsg("Herstellung von Polygonen für Bohrungen ...")
av.ShowStopButton

HFStr=MsgBox.Input("zur Überhöhung des Profils",
    "Eingabe eines Faktors", "50")
HFaktor=HFStr.AsNumber

FTabBPg.SetEditable(false)
FTabBPg.SetEditable(true)
recNr=-1

for each i in 0..IdxPT
  Ng=i+1
  theShape=PTFTab.ReturnValue(PTShpFld, i)
  theRW=theShape.Getx
  theHW=theShape.Gety
  theEntf=PTFTab.ReturnValue(PTEfFld, i)
  theRWl=theEntf-75
```

```
theRWr=theEntf+75
theGH=(PTFTab.ReturnValue(PTGHFld, i))*HFaktor
theTH=(PTFTab.ReturnValue(PTTHFld, i))*HFaktor
theQH=(PTFTab.ReturnValue(PTQHFld, i))*HFaktor
theEH=(PTFTab.ReturnValue(PTEHFld, i))*HFaktor
theAbst=PTFTab.ReturnValue(PTAbstFld, i)
theALB=theAbst.Round.AsString
SW="N"

if (theTH = 0) then
  thePg=Polygon.Make({{theRWl@theGH, theRWr@theGH,
                theRWr@theQH, theRWl@theQH}})
  recNr=recNr+1
  FTabBPg.AddRecord
  FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
  FTabBPg.SetValue(IDFldpg, recNr, recNr)
  FTabBPg.SetValue(PtIDFldPg, recNr, i)
  FTabBPg.SetValue(RWFldpg, recNr, theRW)
  FTabBPg.SetValue(HWFldpg, recNr, theHW)
  FTabBPg.SetValue(EntfFldPg, recNr, theEntf)
  FTabBPg.SetValue(AbstFldpg, recNr, theAbst)
  FTabBPg.SetValue(ALbFldPg, recNr, theALB)
  SW = "V"
  FTabBPg.SetValue(SchichtFldpg, recNr, "QohneD")

  if (theEH < theQH) then
    if (theEH < 0) then
      theEH=(10*HFaktor)
    end
    thePg=Polygon.Make({{theRWl@theQH, theRWr@theQH,
                  theRWr@theEH, theRWl@theEH}})
    recNr=recNr+1
    FTabBPg.AddRecord
    FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
    FTabBPg.SetValue(IDFldpg, recNr, recNr)
    FTabBPg.SetValue(PtIDFldPg, recNr, i)
    FTabBPg.SetValue(RWFldpg, recNr, theRW)
    FTabBPg.SetValue(HWFldpg, recNr, theHW)
    FTabBPg.SetValue(EntfFldPg, recNr, theEntf)
    FTabBPg.SetValue(AbstFldpg, recNr, theAbst)
    if (SW = "N") then
      myALB=theALB
      SW = "V"
    elseif (SW = "V") then
      myALB=""
    end
    FTabBPg.SetValue(ALbFldPg, recNr, myALB)
    FTabBPg.SetValue(SchichtFldpg, recNr, "TrorAe")
  end
elseif (theTH <> 0) then
  thePg=Polygon.Make({{theRWl@theGH, theRWr@theGH,
                theRWr@theTH, theRWl@theTH}})
  recNr=recNr+1
  FTabBPg.AddRecord
  FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
  FTabBPg.SetValue(IDFldpg, recNr, recNr)
  FTabBPg.SetValue(PtIDFldPg, recNr, i)
  FTabBPg.SetValue(RWFldpg, recNr, theRW)
  FTabBPg.SetValue(HWFldpg, recNr, theHW)
  FTabBPg.SetValue(EntfFldPg, recNr, theEntf)
  FTabBPg.SetValue(AbstFldpg, recNr, theAbst)
  FTabBPg.SetValue(ALbFldPg, recNr, theALB)
```

```
SW = "V"
FTabBPg.SetValue(SchichtFldpg, recNr, "Deck")

if (theQH = 0) then
   if (theEH < theTH) then
      if (theEH < 0) then
        theEH=(10*HFaktor)
      end
      thePg=Polygon.Make({{theRWl@theTH, theRWr@theTH,
                    theRWr@theEH, theRWl@theEH}})
      recNr=recNr+1
      FTabBPg.AddRecord
      FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
      FTabBPg.SetValue(IDFldpg, recNr, recNr)
      FTabBPg.SetValue(PtIDFldPg, recNr, i)
      FTabBPg.SetValue(RWFldpg, recNr, theRW)
      FTabBPg.SetValue(HWFldpg, recNr, theHW)
      FTabBPg.SetValue(EntfFldPg, recNr, theEntf)
      FTabBPg.SetValue(AbstFldpg, recNr, theAbst)
      if (SW = "N") then
        myALB=theALB
        SW = "V"
      elseif (SW = "V") then
        myALB=""
      end
      FTabBPg.SetValue(ALbFldPg, recNr, myALB)
      FTabBPg.SetValue(SchichtFldpg, recNr, "QohneT")
   end
elseif (theQH <> 0) then
   thePg=Polygon.Make({{theRWl@theTH, theRWr@theTH,
                 theRWr@theQH, theRWl@theQH}})
   recNr=recNr+1
   FTabBPg.AddRecord
   FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
   FTabBPg.SetValue(IDFldpg, recNr, recNr)
   FTabBPg.SetValue(PtIDFldPg, recNr, i)
   FTabBPg.SetValue(RWFldpg, recNr, theRW)
   FTabBPg.SetValue(HWFldpg, recNr, theHW)
   FTabBPg.SetValue(EntfFldPg, recNr, theEntf)
   FTabBPg.SetValue(AbstFldpg, recNr, theAbst)
   if (SW = "N") then
     myALB=theALB
     SW = "V"
   elseif (SW = "V") then
     myALB=""
   end
   FTabBPg.SetValue(ALbFldPg, recNr, myALB)
   FTabBPg.SetValue(SchichtFldpg, recNr, "QmitD")

   if (theEH < theQH) then
     if (theEH < 0) then
       theEH=(10*HFaktor)
     end
     thePg=Polygon.Make({{theRWl@theQH, theRWr@theQH,
                   theRWr@theEH, theRWl@theEH}})
     recNr=recNr+1
     FTabBPg.AddRecord
     FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
     FTabBPg.SetValue(IDFldpg, recNr, recNr)
     FTabBPg.SetValue(PtIDFldPg, recNr, i)
     FTabBPg.SetValue(RWFldpg, recNr, theRW)
     FTabBPg.SetValue(HWFldpg, recNr, theHW)
```

```
          FTabBPg.SetValue(EntfFldPg, recNr, theEntf)
          FTabBPg.SetValue(AbstFldpg, recNr, theAbst)
          if (SW = "N") then
            myALB=theALB
            SW = "V"
          elseif (SW = "V") then
            myALB=""
          end
          FTabBPg.SetValue(ALbFldPg, recNr, myALB)
          FTabBPg.SetValue(SchichtFldpg, recNr, "TrorAe")
        end
      end
    end

    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzPt*100)
    if (not more) then
      break
    end
  end

  FTabBPg.SetEditable(false)
  thmNew=FTheme.Make(FTabBPg)
  PfView.AddTheme(thmNew)

  'Veränderung der Legende der Bohrungen als Stapel-Arbeit
  av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
  thePalette=av.GetSymbolWin.GetPalette

  av.ShowMsg("Veränderung der Legende der Bohrungen...")

  theTheme=thmNew
  FTabP=theTheme.GetFTab
  AnzRecP=0
  for each aRecP in FTabP
     AnzRecP=AnzRecP+1
  end


  'Veränderung der Legende der Bohrungen als Stapel-Arbeit
  av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
  thePalette=av.GetSymbolWin.GetPalette

  ListofFarbD = {"Deck", 53, "QohneD", 26, "QmitD", 20,
                 "QohneT", 8, "TrorAe", 3}

  av.ShowMsg("Veränderung der Legende der Bohrungen...")

  FTabP=theTheme.GetFTab
  AnzRecP=0
  for each aRecP in FTabP
     AnzRecP=AnzRecP+1
  end

  if (AnzRecP <> 0) then
    theLegend=theTheme.GetLegend
    theLegend.SetLegendType(#Legend_Type_Unique)
    theLegend.Unique(theTheme, "Schicht")
    ListofKlasse=theLegend.GetClassifications
    AnzKlasse=ListofKlasse.Count
    IdxKlasse=AnzKlasse-2
    'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
```

```
  aListofNr={}
  for each i in 0..IdxKlasse
     theKlasseLb=ListofKlasse.Get(i).GetLabel
     'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
     aLbIdx = ListofFarbD.FindByValue(theKlasseLb)
     if (aLbIdx <> -1) then
       aCNr = ListofFarbD.Get(aLbIdx + 1)
     elseif (aLbIdx = -1) then
       aCNr = 3
     end
     aListofNr.Add(aCNr)
  end
  'theLegend.SetNullValue("Schicht", "Deck")
  aListofSymbol=theLegend.GetSymbols
  AnzSymb=aListofSymbol.Count
  'MsgBox.Info(AnzSymb.AsString, "Anzahl der Symbole")
  AnzSymbIdx=AnzSymb-2
  aListofColor={}

  for each Nmb in 0..IdxKlasse
    aNumb=aListofNr.Get(Nmb)
    theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
    theRgbList=theColor.GetRgbList
    aColor=Color.Make
    aColor.SetRgbList(theRgbList)
    aListofColor.Add(aColor)
  end
  for each symb in 0..AnzSymbIdx
     aListofSymbol.Get(symb).SetColor(aListofColor.Get(symb))
  end
  theTheme.UpdateLegend
end




'pfbl2geo.ave
'Eine Profillinie in einem Thema wird durch eine digitale geologische Karte
'in Teillinien mit den geologischen Bezeichnungen unterteilt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc     'Aktives Karten-View

ListofThemes=theView.GetThemes
ListofPLFThm = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     aClassNm= aFTab.GetShapeClass
     if (aClassNm.IsSubclassOf(PolyLine)) then
       ListofPLFThm.Add(aT)
     end
   end
end

PLTheme=MsgBox.ChoiceAsString(ListofPLFThm,
 "das die 2D-Profillinien enthält,"+NL+"um geologische Linien herzustellen",
       "Eingabe eines Themas im View"++theView.AsString)
PLFTab=PLTheme.GetFTab
```

```
PLShpFld=PLFTab.FindField("Shape")
ListofPLFTabFlds=PLFTab.GetFields
PLIDFld=MsgBox.ChoiceAsString(ListofPLFTabFlds, "das ID der PolyLine enthält",
      "Auswahl eines Feldes im Thema:"++PLTheme.AsString)

'Feststellung der Anzahl der 2D-PolyLine in dem Thema
Anz2DPL=0
for each rec in PLFTab
   Anz2DPL=Anz2DPL+1
end
Idx2DPL=Anz2DPL-1
MsgBox.Info(Anz2DPL.AsString, "Anzahl der 2D-PolyLine im Thema"++PLTheme.AsString)

AnzPLFlds=ListofPLFTabFlds.Count
IdxPLFlds=AnzPLFlds-1
ListofDtStr={}

for each i in 0..Idx2DPL
   DtStr=""
   for each j in 0..IdxPLFlds
      aPLFld=ListofPLFTabFlds.Get(j)
      aPLFldStr=aPLFld.GetName
      if (aPLFldStr <> "Shape") then
         aDt=PLFTab.ReturnValue(aPLFld, i)
         DtStr=DtStr+aDt.AsString+"; "
      end
   end
   ListofDtStr.Add(DtStr)
end

'Eingabe des Themas von der digitalen geologischen Karte
GeoTheme=MsgBox.ChoiceAsString(ListofThemes, "das die Polygone für Geologie enthält",
       "Eingabe eines Themas im View"++theView.AsString) 'z.B. Clip1.shp

GeoFTab=GeoTheme.GetFTab
GeoShpFld=GeoFTab.FindField("Shape")
ListofFlds=GeoFTab.GetFields
GeoFld=MsgBox.ChoiceAsString(ListofFlds, "Das Feld für Geolgie",
      "Auswahl des Feldes im Thema"++GeoTheme.AsString)

'Feststellung der Anzahl der Shapes in dem geologischen Schichten-Polygon-Thema
AnzGR=0
for each rec in GeoFTab
   AnzGR=AnzGR+1
end
AnzGRIdx=AnzGR-1
MsgBox.Info(AnzGR.AsString, "Anzahl der Shapes in der geologischen
Karte"++GeoTheme.AsString)

av.ShowMsg("Herstellung der gelogisch geteilten Profillinien ...")

'Auswahl einer PolyLine im PolyLine-Thema
aDtSatz=MsgBox.ChoiceAsString(ListofDtStr, "mit einem Datensatz",
      "Auswahl einer PolyLine im Thema:"++PLTheme.AsString)
aIdx=ListofDtStr.FindByValue(aDtSatz)
PLShp=PLFTab.ReturnValue(PLShpFld, aIdx)

'Eingabe eines Polygon-Themas für das Modellierungsgebiet
theMRThm=MsgBox.ChoiceAsString(ListofThemes,
     "das das Polygon für Modellierungsrahmen enthält",
     "Auswahl eines Themas im View"++theView.AsString)
MRFTab=theMRThm.GetFTab
```

```
MRShpFld=MRFTab.FindField("Shape")
aMRPolygon=MRFTab.ReturnValue(MRShpFld, 0)

'Ausschneiden der Profillinie im Modellierungsgebiet
ListofPLPt={}
minPLx=100000000
maxPLx=0

ListofPolyLine={}
Qt1=aMRPolygon.Intersects (PLShp)
if (Qt1) then
  thePrfl=PLShp.LineIntersection(aMRPolygon)
  ListofPolyLine.Add({thePrfl})
elseif (Not Qt1) then
  Qt2=aMRPolygon.Contains (PLShp)
  if (Qt2) then
    ListofPolyLine.Add({PLShp})
  end
end

if (ListofPolyLine <> 0) then
  for each aShpL in ListofPolyLine
     aListofLtofPt=aShpL.Get(0).AsList
     for each aLtofPt in aListofLtofPt
        for each ePt in aLtofPt
           ListofPLPt.Add(ePt)
           myx=ePt.Getx
           myy=ePt.Gety
           if (myx < minPLx) then
             minPLx=myx
             minPLy=myy
           elseif (myx > maxPLx) then
             maxPLx=myx
             maxPLy=myy
           end
        end
     end
  end
end
ListofListofPLPt={}
ListofListofPLPt.Add(ListofPLPt)
thePLM=PolyLine.Make(ListofListofPLPt)

ListofGeolog={}
ListofPg={}

for each aShape in 0..AnzGRIdx
   theShape=GeoFTab.ReturnValue(GeoShpFld, aShape)
   if (thePLM.Intersects(theShape)) then
     theGeologie=GeoFTab.ReturnValue(GeoFld, aShape)
     ListofGeolog.Add(theGeologie)
     ListofPg.Add(theShape)
   end
end
AnzPg=ListofPg.Count
AnzPgIdx=AnzPg-1

if (AnzPg <> 0) then
 ListofGeo={}
 ListofNewPolyL={}
 for each aShape in 0..AnzPgIdx
    theGeologie=ListofGeolog.Get(aShape)
```

```
    theShape=ListofPg.Get(aShape)
    NewPolyL=thePLM.LineIntersection(theShape)
    ListofCheck1=NewPolyL.AsList
    Check1=ListofCheck1.Count
    if (Check1 <> 0) then
      ListofTeile=NewPolyL.Explode
      AnzderTeile=ListofTeile.Count
      AnzdTIdx=AnzderTeile-1
      for each j in 0..AnzdTIdx
        ListofGeo.Add(theGeologie)
        NewPolyLT=ListofTeile.Get(j)
        ListofNewPolyL.Add(NewPolyLT)
      end
    end
  end

  AnzNewPL=ListofNewPolyL.Count
  AnzNPLIdx=AnzNewPL-1

  MinX=100000000
  MaxX=0
  for each i in 0..AnzNPLIdx
    N=i+1
    Fg=ListofNewPolyL.Get(i)
    ListofFg=Fg.AsList
    AnfPt=ListofFg.Get(0).Get(0)
    RWAnf=AnfPt.Getx
    HWAnf=AnfPt.Gety

    EndPt=ListofFg.Get(0).Get(1)
    RWEnd=EndPt.Getx
    HWEnd=EndPt.Gety
    if (RWAnf < MinX) then
      MinX=RWAnf
      MinY=HWAnf
    end
    if (RWEnd > MaxX) then
      MaxX=RWEnd
      MaxY=HWEnd
    end
  end

  if (MinX > minPLx) then
    theExtraPL=PolyLine.Make({{minPLx@minPLy, MinX@MinY}})
    ListofNewPolyL.Add(theExtraPL)
    ListofGeo.Add("Aussen")
    AnzNPLIdx=AnzNPLIdx+1
  end
  if (MaxX < maxPLx) then
    theExtraPL=PolyLine.Make({{MaxX@MaxY, maxPLx@maxPLy}})
    ListofNewPolyL.Add(theExtraPL)
    ListofGeo.Add("Aussen")
    AnzNPLIdx=AnzNPLIdx+1
  end
end

av.ShowMsg("Speicherung der gelogisch geteilten Profillinien ...")
'Ein Feature-Shape-File für Geolgie (PolyLine) wird hergestellt.
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("pl2dbmg1","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PolyLine)")
if (fName=nil) then exit end
```

```
fName.SetExtension("shp")
PLgFTab=FTab.MakeNew(fName, PolyLine)
ShapeField1=PLgFTab.FindField("shape")
IDField1=Field.Make("ID", #Field_Short, 5, 0)
GeoField1=Field.Make("Geologie", #Field_Char, 30, 0)
ListofFlds1={IDField1, GeoField1}
PLgFTab.AddFields(ListofFlds1)
ListofFlds2={ShapeField1, IDField1, GeoField1}

PLgFTab.SetEditable(false)
PLgFTab.SetEditable(true)

for each i in 0..AnzNPLIdx
   ListofValue={}
   theShapeV=ListofNewPolyL.Get(i)
   theGeologie=ListofGeo.Get(i)
   ListofValue.Add(theShapeV)
   ListofValue.Add(i)
   ListofValue.Add(theGeologie)
   PLgFTab.AddRecord
   for each j in 0..2
      PLgFTab.SetValue(ListofFlds2.Get(j), i,
                  ListofValue.Get(j))
   end
end

PLgFTab.SetEditable(false)
thmNew=FTheme.Make(PLgFTab)
theView.AddTheme(thmNew)
```

```
'pfbl3dmg.ave
'Aus 2D-Profilschnittlinien in einem Thema im aktiven Karten-View
'wird ein neues Thema mit 3D-Profilschnittlinien, ein neues Thema
'mit Punkten auf den 2D-Profilschnittlinien im aktiven Karten-View
'und ein Thema mit Polygonen in einem 2D-Profilschnitt-View hergestellt.
'Die Höhenwerte werden aus einem Punkt-Thema auf einer langen
'Profilschnittlinie mit Höhenwerten im aktiven Karten-View übernommen.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
KtView=av.GetActiveDoc     'Aktives Karten-View

'Eingabe der Datei, die 2D-Profilschnittlinie zur Umwandlung in 3D enthält
ListofThemes =KtView.GetThemes
ListofPLThms = {}
ListofPtThms = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aShpClass=aFTab.GetShapeClass
      if (aShpClass.IsSubclassOf(Point)) then
         ListofPtThms.Add(aT)
      elseif (aShpClass.IsSubclassOf(Polyline)) then
         ListofPLThms.Add(aT)
      end
   end
end
```

```
PLTheme=MsgBox.ChoiceAsString(ListofPLThms,
     "das die geologischen 2D-Profillinien enthält",
     "Eingabe eines Themas im View"++KtView.AsString)

PLFTab=PLTheme.GetFTab
ListofPLFTabFlds=PLFTab.GetFields

PLShpFld=PLFTab.FindField("Shape")
PLIDFld=MsgBox.ChoiceAsString(ListofPLFTabFlds, "für ID",
       "Auswahl des Feldes im Thema:"++PLTheme.AsString)
PLGeoFld=MsgBox.ChoiceAsString(ListofPLFTabFlds, "für Geologie",
       "Auswahl des Feldes im Thema:"++PLTheme.AsString)
Listof2DFld={PLShpFld, PLIDFld, PLGeoFld}

'Feststellung der Anzahl der 2D-PolyLine in dem Thema PLTheme
Anz2DPL=0
for each rec in PLFTab
   Anz2DPL=Anz2DPL+1
end
Idx2DPL=Anz2DPL-1
MsgBox.Info(Anz2DPL.AsString, "Anzahl der 2D-PolyLine in dem Thema"
      ++PLTheme.AsString)

'Eingabe der Datei, die Punkte auf einer Profilschnittlinie mit
'Entfernungen vom Anfang des Profilschnittes und Hoehenwerten enthält,
'zur Bestmmung der 3D-Punkte der Profilschnittlinie

PTTheme=MsgBox.ChoiceAsString(ListofPtThms,
       "das Punkte auf einer Profilschnittlinie mit Hoehenwerten enthält",
       "Eingabe einesThemas")
PTFTab=PTTheme.GetFTab
ListofFlds=PTFTab.GetFields
PTShpFld=MsgBox.ChoiceAsString(ListofFlds,
       "Eingabe des Feldes für Shape",
       "Auswahl eines Feldes in Thema:"++PTTheme.AsString)
PTGhFld=MsgBox.ChoiceAsString(ListofFlds,
       "Eingabe des Feldes für Geländehöhen [m]",
       "Auswahl eines Feldes in Thema:"++PTTheme.AsString)
PTThFld=MsgBox.ChoiceAsString(ListofFlds,
       "Eingabe des Feldes für Höhen der Deckschichtenbasis [m]",
       "Auswahl eines Feldes in Thema:"++PTTheme.AsString)
PTEfFld=MsgBox.ChoiceAsString(ListofFlds,
       "Eingabe des Feldes für Entfernungen vom Anfang [m]",
       "Auswahl eines Feldes in Thema:"++PTTheme.AsString)
ListofPtFld={PTShpFld, PTGhFld, PTThFld}

'Feststellung der Anzahl der Punkte in dem Thema PTTheme
AnzPT=0
for each rec in PTFTab
   AnzPT=AnzPT+1
end
IdxPT=AnzPT-1
MsgBox.Info(AnzPT.AsString,
     "Anzahl der Punkte in dem Thema"++PTTheme.AsString)

av.ShowMsg("Bestimmung der Punkte auf den geologisch"
        ++"unterteilten Profilschnitten ...")
av.ShowStopButton

ListofListofNPT={}
ListofListofNEntf={}
```

```
ListofListofNGH={}
ListofListofNTH={}
ListofNGeol={}
for each gzd in 0..Idx2DPL
  Ng=gzd+1
  the2DPolyL=PLFTab.ReturnValue(PLShpFld, gzd)
  aGeolog=PLFTab.ReturnValue(PLGeoFld, gzd)
  ListofNGeol.Add(aGeolog)
  ListofLPT={}
  'Die Stützpunkte einer PolyLine werden bestimmt.
  theProfilList={}
  theProfilList.Add({the2DPolyL})
  if (theProfilList <> 0) then
    for each theShpL in theProfilList
      ListofListofPt=theShpL.Get(0).AsList
      for each ListofPt in ListofListofPt
         for each aPt in ListofPt
            ListofLPT.Add(aPt)
         end
      end
    end
  end
  AnzLPT=ListofLPT.Count
  IdxLPT=AnzLPT-1

  AnfPT=ListofLPT.Get(0)
  EndPT=ListofLPT.Get(IdxLPT)
  AnfX=AnfPT.Getx
  AnfY=AnfPT.Gety
  EndX=EndPT.Getx
  EndY=EndPT.Gety
  minAbstA=100000
  minAbstE=100000

  'Suche nach dem Punkt im Punkt-Thema mit dem kleinsten Abstand
  for each i in 0..IdxPT
     thePT=PTFTab.ReturnValue(PTShpFld, i)
     theX=thePT.Getx
     theY=thePT.Gety
     DifXA2=((theX-AnfX)*(theX-AnfX))
     DifYA2=((theY-AnfY)*(theY-AnfY))
     AbstA=(DifXA2+DifYA2).Sqrt.Abs
     if (AbstA < minAbstA) then
       minAbstA=AbstA
       IdxPTA=i
     end
     DifXE2=((theX-EndX)*(theX-EndX))
     DifYE2=((theY-EndY)*(theY-EndY))
     AbstE=(DifXE2+DifYE2).Sqrt.Abs
     if (AbstE < minAbstE) then
       minAbstE=AbstE
       IdxPTE=i
     end
  end

  'Berechnung der Entfernung und der Höhen
  'des ersten Punktes einer Profilschnittlinie
  if (minAbstA <> 0) then
    VIdxPTA=IdxPTA-1
    NIdxPTA=IdxPTA+1

    thePTVA=PTFTab.ReturnValue(PTShpFld, VIdxPTA)
```

```
thePTNA=PTFTab.ReturnValue(PTShpFld, NIdxPTA)
thePTIA=PTFTab.ReturnValue(PTShpFld, IdxPTA)

thePTVAX=thePTVA.Getx
thePTVAY=thePTVA.Gety

thePTNAX=thePTNA.Getx
thePTNAY=thePTNA.Gety

thePTIAX=thePTIA.Getx
thePTIAY=thePTIA.Gety

DifXVA2=((thePTVAX-AnfX)*(thePTVAX-AnfX))
DifYVA2=((thePTVAY-AnfY)*(thePTVAY-AnfY))
AbstVA=(DifXVA2+DifYVA2).Sqrt.Abs

DifXNA2=((thePTNAX-AnfX)*(thePTNAX-AnfX))
DifYNA2=((thePTNAY-AnfY)*(thePTNAY-AnfY))
AbstNA=(DifXNA2+DifYNA2).Sqrt.Abs

DifXVIA=((thePTVAX-thePTIAX)*(thePTVAX-thePTIAX))
DifYVIA=((thePTVAY-thePTIAY)*(thePTVAY-thePTIAY))
AbstVIA=(DifXVIA+DifYVIA).Sqrt.Abs

DifXNIA=((thePTNAX-thePTIAX)*(thePTNAX-thePTIAX))
DifYNIA=((thePTNAY-thePTIAY)*(thePTNAY-thePTIAY))
AbstNIA=(DifXNIA+DifYNIA).Sqrt.Abs

if ((AbstVA < AbstVIA) and (AbstNA > AbstNIA)) then
  VIdxPTA=IdxPTA-1
  NIdxPTA=IdxPTA
  P1IdxPTA=IdxPTA+1
  SWA="V"
elseif ((AbstVA > AbstVIA) and (AbstNA < AbstNIA)) then
  VIdxPTA=IdxPTA
  NIdxPTA=IdxPTA+1
  P1IdxPTA=IdxPTA+1
  SWA="N"
end
thePTVA=PTFTab.ReturnValue(PTShpFld, VIdxPTA)
thePTNA=PTFTab.ReturnValue(PTShpFld, NIdxPTA)

thePTVAX=thePTVA.Getx
thePTVAY=thePTVA.Gety

thePTNAX=thePTNA.Getx
thePTNAY=thePTNA.Gety

DifXVA2=((thePTVAX-AnfX)*(thePTVAX-AnfX))
DifYVA2=((thePTVAY-AnfY)*(thePTVAY-AnfY))
AbstVA=(DifXVA2+DifYVA2).Sqrt.Abs
EntfVA=PTFTab.ReturnValue(PTEfFld, VIdxPTA)
EntfPTA=EntfVA+AbstVA

DifXNA2=((thePTNAX-AnfX)*(thePTNAX-AnfX))
DifYNA2=((thePTNAY-AnfY)*(thePTNAY-AnfY))
AbstNA=(DifXNA2+DifYNA2).Sqrt.Abs
EntfNA=PTFTab.ReturnValue(PTEfFld, NIdxPTA)

theGHVA=PTFTab.ReturnValue(PTGhFld, VIdxPTA)
theGHNA=PTFTab.ReturnValue(PTGhFld, NIdxPTA)
```

```
  AbstVNA=(EntfNA-EntfVA).Abs
  GHuntVN=(theGHNA-theGHVA).Abs

  if (theGHVA > theGHNA) then
    GHPtA=(AbstNA/AbstVNA)*GHuntVN+theGHNA
  elseif (theGHVA < theGHNA) then
    GHPtA=(AbstVA/AbstVNA)*GHuntVN+theGHVA
  elseif (theGHVA = theGHNA) then
    GHPtA=theGHVA
  end

  theTHVA=PTFTab.ReturnValue(PTThFld, VIdxPTA)
  theTHNA=PTFTab.ReturnValue(PTThFld, NIdxPTA)
  THuntVN=(theTHNA-theTHVA).Abs

  if (theTHVA > theTHNA) then
    THPtA=(AbstNA/AbstVNA)*THuntVN+theTHNA
  elseif (theTHVA < theTHNA) then
    THPtA=(AbstVA/AbstVNA)*THuntVN+theTHVA
  elseif (theTHVA = theTHNA) then
    THPtA=theTHVA
  end

elseif (minAbstA = 0) then
  EntfPTA=PTFTab.ReturnValue(PTEfFld, IdxPTA)
  GHPtA=PTFTab.ReturnValue(PTGhFld, IdxPTA)
  THPtA=PTFTab.ReturnValue(PTThFld, IdxPTA)
  P1IdxPTA=IdxPTA+1
  SWA="N"
end

'Berechnung der Entfernung und der Höhen
'des letzten Punktes einer Profilschnittlinie

if (minAbstE <> 0) then
  VIdxPTE=IdxPTE-1
  NIdxPTE=IdxPTE+1

  thePTE=PTFTab.ReturnValue(PTShpFld, IdxPTE)
  thePTVE=PTFTab.ReturnValue(PTShpFld, VIdxPTE)
  thePTNE=PTFTab.ReturnValue(PTShpFld, NIdxPTE)

  thePTEX=thePTE.Getx
  thePTEY=thePTE.Gety

  thePTVEX=thePTVE.Getx
  thePTVEY=thePTVE.Gety

  thePTNEX=thePTNE.Getx
  thePTNEY=thePTNE.Gety

  DifXVE2=((thePTVEX-EndX)*(thePTVEX-EndX))
  DifYVE2=((thePTVEY-EndY)*(thePTVEY-EndY))
  AbstVE=(DifXVE2+DifYVE2).Sqrt.Abs

  DifXNE2=((thePTNEX-EndX)*(thePTNEX-EndX))
  DifYNE2=((thePTNEY-EndY)*(thePTNEY-EndY))
  AbstNE=(DifXNE2+DifYNE2).Sqrt.Abs

  DifXVIE=((thePTVEX-thePTEX)*(thePTVEX-thePTEX))
  DifYVIE=((thePTVEY-thePTEY)*(thePTVEY-thePTEY))
  AbstVIE=(DifXVIE+DifYVIE).Sqrt.Abs
```

```
DifXNIE=((thePTNEX-thePTEX)*(thePTNEX-thePTEX))
DifYNIE=((thePTNEY-thePTEY)*(thePTNEY-thePTEY))
AbstNIE=(DifXNIE+DifYNIE).Sqrt.Abs

if ((AbstVE < AbstVIE) and (AbstNE > AbstNIE)) then
  VIdxPTE=IdxPTE-1
  NIdxPTE=IdxPTE
  M1IdxPTE=IdxPTE-1
  SWE="V"
elseif ((AbstVE > AbstVIE) and (AbstNE < AbstNIE)) then
  VIdxPTE=IdxPTE
  NIdxPTE=IdxPTE+1
  M1IdxPTE=IdxPTE-1
  SWE="N"
end

thePTVE=PTFTab.ReturnValue(PTShpFld, VIdxPTE)
thePTNE=PTFTab.ReturnValue(PTShpFld, NIdxPTE)

thePTVEX=thePTVE.Getx
thePTVEY=thePTVE.Gety

thePTNEX=thePTNE.Getx
thePTNEY=thePTNE.Gety

DifXVE2=((thePTVEX-EndX)*(thePTVEX-EndX))
DifYVE2=((thePTVEY-EndY)*(thePTVEY-EndY))
AbstVE=(DifXVE2+DifYVE2).Sqrt.Abs
EntfVE=PTFTab.ReturnValue(PTEfFld, VIdxPTE)
EntfPTE=EntfVE+AbstVE

DifXNE2=((thePTNEX-EndX)*(thePTNEX-EndX))
DifYNE2=((thePTNEY-EndY)*(thePTNEY-EndY))
AbstNE=(DifXNE2+DifYNE2).Sqrt.Abs
EntfNE=PTFTab.ReturnValue(PTEfFld, NIdxPTE)

theGHVE=PTFTab.ReturnValue(PTGhFld, VIdxPTE)
theGHNE=PTFTab.ReturnValue(PTGhFld, NIdxPTE)

AbstVNE=(EntfNE-EntfVE).Abs
GHuntVN=(theGHNE-theGHVE).Abs

if (theGHVE > theGHNE) then
  GHPtE=(AbstNE/AbstVNE)*GHuntVN+theGHNE
elseif (theGHVE < theGHNE) then
  GHPtE=(AbstVE/AbstVNE)*GHuntVN+theGHVE
elseif (theGHVE = theGHNE) then
  GHPtE=theGHVE
end

theTHVE=PTFTab.ReturnValue(PTThFld, VIdxPTE)
theTHNE=PTFTab.ReturnValue(PTThFld, NIdxPTE)
THuntVN=(theTHNE-theTHVE).Abs

if (theTHVE > theTHNE) then
  THPtE=(AbstNE/AbstVNE)*THuntVN+theTHNE
elseif (theTHVE < theTHNE) then
  THPtE=(AbstVE/AbstVNE)*THuntVN+theTHVE
elseif (theTHVE = theTHNE) then
  THPtE=theTHVE
end
```

```
elseif (minAbstE = 0) then
  EntfPTE=PTFTab.ReturnValue(PTEfFld, IdxPTE)
  GHPtE=PTFTab.ReturnValue(PTGhFld, IdxPTE)
  THPtE=PTFTab.ReturnValue(PTThFld, IdxPTE)
  M1IdxPTE=IdxPTE-1
  SWE="V"
end

ListofNPT={}
ListofNEntf={}
ListofNGH={}
ListofNTH={}

ListofNPT.Add(AnfPT)
ListofNEntf.Add(EntfPTA)
ListofNGH.Add(GHPtA)
ListofNTH.Add(THPtA)

if (SWA = "V") then
   theNNPT=PTFTab.ReturnValue(PTShpFld, NIdxPTA)
   ListofNPT.Add(theNNPT)

   theNNEntf=PTFTab.ReturnValue(PTEfFld, NIdxPTA)
   ListofNEntf.Add(theNNEntf)

   theNNGH=PTFTab.ReturnValue(PTGhFld, NIdxPTA)
   ListofNGH.Add(theNNGH)

   theNNTH=PTFTab.ReturnValue(PTThFld, NIdxPTA)
   ListofNTH.Add(theNNTH)
end

for each i in 0..IdxPT
    if (((i = P1IdxPTA) or (i > P1IdxPTA)) and
       ((i = M1IdxPTE) or (i < M1IdxPTE))) then
       theNPT=PTFTab.ReturnValue(PTShpFld, i)
       ListofNPT.Add(theNPT)

       theNNEntf=PTFTab.ReturnValue(PTEfFld, i)
       ListofNEntf.Add(theNNEntf)

       theNNGH=PTFTab.ReturnValue(PTGhFld, i)
       ListofNGH.Add(theNNGH)

       theNNTH=PTFTab.ReturnValue(PTThFld, i)
       ListofNTH.Add(theNNTH)
    end
end

if (SWE = "N") then
   theNVPT=PTFTab.ReturnValue(PTShpFld, IdxPTE)
   ListofNPT.Add(theNVPT)

   theNNEntf=PTFTab.ReturnValue(PTEfFld, IdxPTE)
   ListofNEntf.Add(theNNEntf)

   theNNGH=PTFTab.ReturnValue(PTGhFld, IdxPTE)
   ListofNGH.Add(theNNGH)

   theNNTH=PTFTab.ReturnValue(PTThFld, IdxPTE)
   ListofNTH.Add(theNNTH)
```

```
  end

  ListofNPT.Add(EndPT)
  ListofNEntf.Add(EntfPTE)
  ListofNGH.Add(GHPtE)
  ListofNTH.Add(THPtE)

  ListofListofNPT.Add(ListofNPT)
  ListofListofNEntf.Add(ListofNEntf)
  ListofListofNGH.Add(ListofNGH)
  ListofListofNTH.Add(ListofNTH)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/Anz2DPL*100)
  if (not more) then
    break
  end
end

'Ein Feature-Shape-File für Geologie (PolyLineZ) wird hergestellt.
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("Pfplktg1","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output 3D shape File (PolyLineZ)")
if (fName=nil) then exit end
fName.SetExtension("shp")
GPLFTab=FTab.MakeNew(fName, PolyLineZ)
GPLShpFld=GPLFTab.FindField("shape")
GPLIDFld=Field.Make("ID", #Field_Short, 5, 0)
GPLFIdFld=Field.Make("ID_FL", #Field_Short, 5, 0)
GPLGeoFld=Field.Make("Geologie", #Field_Char, 14, 0)
ListofGPLFlds={GPLIDFld, GPLFIdFld, GPLGeoFld}
GPLFTab.AddFields(ListofGPLFlds)
ListofGPLFld2={GPLShpFld, GPLIDFld, GPLFIdFld, GPLGeoFld}

av.ShowMsg("Speicherung der 3D-PolyLine der geologisch"
        ++"unterteilten Profilschnitte ...")
av.ShowStopButton

GPLFTab.SetEditable(false)
GPLFTab.SetEditable(true)

RNr = -1
for each j in 0..1
  for each aPL in 0..Idx2DPL
    Ng=aPL+1
    RNr = RNr +1

    ListofNPT=ListofListofNPT.Get(aPL)
    ListofNEntf=ListofListofNEntf.Get(aPL)
    if (j = 0) then
      ListofNH=ListofListofNGH.Get(aPL)
      aNGeol=ListofNGeol.Get(aPL)++"/GH"
    elseif (j = 1) then
      ListofNH=ListofListofNTH.Get(aPL)
      aNGeol=ListofNGeol.Get(aPL)++"/DB"  'Basis der Deckschichten
    end

    AnzNPT=ListofNPT.Count
    IdxNPT=AnzNPT-1

    Listof3DNPT={}
    for each aPt in 0..IdxNPT
      theNPT=ListofNPT.Get(aPt)
```

```
      theNH=ListofNH.Get(aPt)
      theNPTx=theNPT.Getx
      theNPTy=theNPT.Gety
      Listof3DNPT.Add(theNPTx@theNPTy@theNH)
    end

    'Speicherung der Punkte
    ListofListofHNPt={}
    ListofListofHNPt.Add(Listof3DNPT)
    New3DPolyL=PolyLineZ.Make(ListofListofHNPt)

    GPLFTab.AddRecord
    GPLFTab.SetValue(GPLShpFld, RNr, New3DPolyL)
    GPLFTab.SetValue(GPLIDFld, RNr, RNr)
    GPLFTab.SetValue(GPLFIdFld, RNr, aPL)
    GPLFTab.SetValue(GPLGeoFld, RNr, aNGeol)

    'Show percentage complete with enabled stop button
    more=av.SetStatus((Ng/(Anz2DPL*2))*100)
    if (not more) then
      exit
    end
  end
end

GPLFTab.SetEditable(false)
GthmNew=FTheme.Make(GPLFTab)
KtView.AddTheme(GthmNew)    'Zeichnung des neuen Themas
                            'im aktiven 2D-Karten-View

'Ein Feature-Shape-File für Geologie (Point) wird hergestellt.
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("Pfptktg1","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output 2D shape File (Point)")
if (fName=nil) then exit end
fName.SetExtension("shp")
NPTFTab=FTab.MakeNew(fName, Point)
NPTShpFld=NPTFTab.FindField("shape")
NPTIDFld=Field.Make("ID", #Field_Short, 6, 0)
NPTPrfIDFld=Field.Make("PrfID", #Field_Short, 5, 0)
NPTEntfFld=Field.Make("Entfernung", #Field_Float, 9, 2)
NPTGHFld=Field.Make("GH (m)", #Field_Float, 7, 2)
NPTTHFld=Field.Make("DabH (m)", #Field_Float, 7, 2)
NPTGeoFld=Field.Make("Geologie", #Field_Char, 10, 0)
ListofNPTFlds={NPTIDFld, NPTPrfIDFld, NPTEntfFld, NPTGHFld,
            NPTTHFld, NPTGeoFld}
NPTFTab.AddFields(ListofNPTFlds)
ListofNPTFld2={NPTShpFld, NPTIDFld, NPTPrfIDFld, NPTEntfFld,
            NPTGHFld, NPTTHFld, NPTGeoFld}

av.ShowMsg("Speicherung der Punkte auf den geologisch"
        ++"unterteilten Profilschnitten ...")
av.ShowStopButton

NPTFTab.SetEditable(false)
NPTFTab.SetEditable(true)

recNr=-1
for each aPL in 0..Idx2DPL
  Ng=aPL+1
  theNPrfID=aPL
  ListofNPT=ListofListofNPT.Get(aPL)
```

```
  ListofNEntf=ListofListofNEntf.Get(aPL)
  ListofNGH=ListofListofNGH.Get(aPL)
  ListofNTH=ListofListofNTH.Get(aPL)
  aNGeol=ListofNGeol.Get(aPL)

  AnzNPT=ListofNPT.Count
  IdxNPT=AnzNPT-1

  for each aPt in 0..IdxNPT
     theNPT=ListofNPT.Get(aPt)
     theNEnf=ListofNEntf.Get(aPt)
     theNGH=ListofNGH.Get(aPt)
     theNTH=ListofNTH.Get(aPt)
     theNGeol=aNGeol
     recNr=recNr+1
     theNID=recNr
     NPTFTab.AddRecord
     NPTFTab.SetValue(NPTShpFld, recNr, theNPT)
     NPTFTab.SetValue(NPTIDFld, recNr, theNID)
     NPTFTab.SetValue(NPTPrfIDFld, recNr, theNPrfID)
     NPTFTab.SetValue(NPTEntfFld, recNr, theNEnf)
     NPTFTab.SetValue(NPTGHFld, recNr, theNGH)
     NPTFTab.SetValue(NPTTHFld, recNr, theNTH)
     NPTFTab.SetValue(NPTGeoFld, recNr, aNGeol)
  end

  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/Anz2DPL*100)
  if (not more) then
    break
  end
end

NPTFTab.SetEditable(false)
NthmNew=FTheme.Make(NPTFTab)
KtView.AddTheme(NthmNew)  'Zeichnung des neuen Themas
                          'im aktiven Karten-View
ListofView={"Qprf_gg", "Bprf_gg", "Eprf_gg", "Lprf_gg",
           "Qprf_ggd", "Bprf_ggd", "Eprf_ggd", "Lprf_ggd"}
theBPrfViewStr=MsgBox.ChoiceAsString(ListofView,
     "das die beliebigen Profilschnitte enthält", "Auswahl eines Views")
theBPrfView=theProject.FindDoc(theBPrfViewStr)

'Ein Feature-Shape-File für Geologie (Polygon) wird hergestellt.
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("Bpfpgdg1","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output 2D shape File (Polygon)")
if (fName=nil) then exit end
fName.SetExtension("shp")
DPgFTab=FTab.MakeNew(fName, Polygon)
DPgShpFld=DPgFTab.FindField("shape")
DPgIDFld=Field.Make("ID", #Field_Short, 5, 0)
DPgGeoFld=Field.Make("Geologie", #Field_Char, 10, 0)
ListofDPgFlds={DPgIDFld, DPgGeoFld}
DPgFTab.AddFields(ListofDPgFlds)
ListofDPgFld2={DPgShpFld, DPgIDFld, DPgGeoFld}

FaktStr=MsgBox.Input("zur Überhöhung des Profils",
                  "Eingabe des Faktors", "50")
Fakt=FaktStr.AsNumber

av.ShowMsg("Speicherung der Polygone der geologisch"
```

```
        ++"unterteilten Profilschnitten ...")
av.ShowStopButton

DPgFTab.SetEditable(false)
DPgFTab.SetEditable(true)

for each aPL in 0..Idx2DPL
  Ng=aPL+1
  ListofNEntf=ListofListofNEntf.Get(aPL)
  ListofNGH=ListofListofNGH.Get(aPL)
  ListofNTH=ListofListofNTH.Get(aPL)
  aNGeol=ListofNGeol.Get(aPL)

  AnzNPT=ListofNEntf.Count
  IdxNPT=AnzNPT-1

  Listof2DPgPT={}
  for each aPt in 0..IdxNPT
     theNEnf=ListofNEntf.Get(aPt)
     theNGH=(ListofNGH.Get(aPt))*Fakt
     Listof2DPgPT.Add(theNEnf@theNGH)
  end

  for each aPt in 0..IdxNPT
     UmkIdx=IdxNPT-aPt
     theNEnf=ListofNEntf.Get(UmkIdx)
     theNTH=(ListofNTH.Get(UmkIdx))*Fakt
     Listof2DPgPT.Add(theNEnf@theNTH)
  end

  ListofListof2DPgPT={}
  ListofListof2DPgPT.Add(Listof2DPgPT)
  theNPg=Polygon.Make(ListofListof2DPgPT)

  'Speicherung der Polygone
  DPgFTab.AddRecord
  DPgFTab.SetValue(DPgShpFld, aPL, theNPg)
  DPgFTab.SetValue(DPgIDFld, aPL, aPL)
  DPgFTab.SetValue(DPgGeoFld, aPL, aNGeol)

  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/Anz2DPL*100)
  if (not more) then
    break
  end
end

DPgFTab.SetEditable(false)
DPgthmNew=FTheme.Make(DPgFTab)
theBPrfView.AddTheme(DPgthmNew)  'Zeichnung des neuen Themas
                          'im 2D-Profilschnitt-View




'pfbohr3d.ave
'Bohrungen in einem Karten-View werden als 3D-PolygonZ in 3D-Szene gezeichnet.
'Ein Menü oder eine Schaltfläche in einer aktiven 3D-Szene zum Anklicken.

theProject=av.GetProject
```

```
myScript=theProject.FindScript("pfbohr3d")
myScript.SetNumberFormat( "d.dd") ' script default

theSzene=av.GetActiveDoc     '3D-Szene
ListofView={"Kt1_gg", "Kt1_ggd", "3D Szene1"}
KtViewStr=MsgBox.ListAsString(ListofView,
     "wo sich das Thema für Bohrungen befindet.",
     "Auswahl eines Dokumentes")
KtView=theProject.FindDoc(KtViewStr)

ListofKtThms=KtView.GetThemes
ListofPtThm = {}

for each aT in ListofKtThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
       ListofPtThm.Add(aT)
     end
   end
end

PTTheme=MsgBox.ChoiceAsString(ListofPtThm, "Datei mit Bohrdaten",
     "Auswahl eines Themas im View"++KtView.AsString)

PTFTab=PTTheme.GetFTab
PTFields=PTFTab.GetFields
PTShpFld=PTFTab.FindField("Shape")
PTGHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Geländehöhe",
                        "Eingabe eines Feldes in"++PTTheme.AsString)
PTTHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Höhe der Terrassenoberkante",
                        "Eingabe eines Feldes in"++PTTheme.AsString)
PTQHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Quartärbasishöhe",
                       "Eingabe eines Feldes in"++PTTheme.AsString)
PTEHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Bohrungsendhöhe",
                       "Eingabe eines Feldes in"++PTTheme.AsString)

ListofPTFld={PTShpFld, PTGHFld, PTTHFld, PTQHFld, PTEHFld}

'Feststellung der Anzahl der ausgewählten Bohrungen
AnzPT=0
for each rec in PTFTab
  AnzPT=AnzPT+1
end
IdxPT=AnzPT-1

WDStr=theProject.GetWorkDir.AsString
DfnStr=PTTheme.AsString.Left(6)

fnStr=FileName.Make(WDStr).MakeTmp(DfnStr,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PolygonZ)")
if (fName=nil) then exit end
fName.SetExtension("shp")

PgzFTab=FTab.MakeNew(fName, PolygonZ)
ShpFld=PgzFTab.FindField("shape")
IDFld=Field.Make("ID", #Field_Short, 5, 0)
BoIDFld=Field.Make("Bohr_ID", #Field_Short, 4, 0)
RWFld=Field.Make("RW", #Field_Float, 10, 2)
HWFld=Field.Make("HW", #Field_Float, 10, 2)
HobFld=Field.Make("Hoehe_Ok", #Field_Float, 7, 2)
HuntFld=Field.Make("Hoehe_Uk", #Field_Float, 7, 2)
```

```
SchichtFld=Field.Make("Schicht", #Field_Char, 10, 0)

ListofB1={IDFld,BoIDFld,RWFld,HWFld,HobFld,HuntFld,SchichtFld}
PgzFTab.AddFields(ListofB1)
ListofB2={ShpFld,IDFld,BoIDFld,RWFld,HWFld,HobFld,HuntFld,SchichtFld}

'Ein Feature-Shape-File für Bohrungen (PolygonZ) wird hergestellt.

av.ShowMsg("Herstellung von PolygonZ für Bohrungen ...")
av.ShowStopButton

ListofListofBohr={}
for each i in 0..IdxPT
  Ng=i+1
  theShape=PTFTab.ReturnValue(PTShpFld, i)
  theGH=PTFTab.ReturnValue(PTGHFld, i)
  theTH=PTFTab.ReturnValue(PTTHFld, i)
  theQH=PTFTab.ReturnValue(PTQHFld, i)
  theEH=PTFTab.ReturnValue(PTEHFld, i)

  if (theTH = 0) then
    ListofBohr={theShape, i, theGH, theQH, "QohneD"}
    ListofListofBohr.Add(ListofBohr)
    if (theEH < theQH) then
      if (theEH < 0) then
        theEH=0
      end
      ListofBohr={theShape, i, theQH, theEH, "HTorAe"}
      ListofListofBohr.Add(ListofBohr)
    end
  elseif (theTH <> 0) then
    ListofBohr={theShape, i, theGH, theTH, "Deck"}
    ListofListofBohr.Add(ListofBohr)
    if (theQH = 0) then
       if (theEH < theTH) then
          if (theEH < 0) then
            theEH=0
          end
          ListofBohr={theShape, i, theTH, theEH, "QohneAe"}
          ListofListofBohr.Add(ListofBohr)
       end
    elseif (theQH <> 0) then
       ListofBohr={theShape, i, theTH, theQH, "QmitD"}
       ListofListofBohr.Add(ListofBohr)
       if (theEH < theQH) then
         if (theEH < 0) then
           theEH=0
         end
         ListofBohr={theShape, i, theQH, theEH, "HTorAe"}
         ListofListofBohr.Add(ListofBohr)
       end
    end
  end

  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzPt*100)
  if (not more) then
    break
  end
end

AnzBohr=ListofListofBohr.Count
```

```
IdxBohr=AnzBohr-1

PgzFTab.SetEditable(false)
PgzFTab.SetEditable(true)
recNr=-1
for each i in 0..IdxBohr
  aListofBohr=ListofListofBohr.Get(i)
  aShp=aListofBohr.Get(0)
  aRW=aShp.Getx
  aHW=aShp.Gety
  aRWw=aRW-100
  aRWe=aRW+100
  aHWn=aHW+100
  aHWs=aHW-100
  BoId=aListofBohr.Get(1)
  Hob=aListofBohr.Get(2)
  Hunt=aListofBohr.Get(3)
  aText=aListofBohr.Get(4)

'   a6FlPgz=PolygonZ.Make({
'   {aRWw@aHWn@Hob, aRWe@aHWn@Hob, aRWe@aHWs@Hob, aRWw@aHWs@Hob},
'   {aRWw@aHWn@Hunt, aRWe@aHWn@Hunt, aRWe@aHWs@Hunt,
aRWw@aHWs@Hunt},
'   {aRWw@aHWs@Hob, aRWe@aHWs@Hob, aRWe@aHWs@Hunt, aRWw@aHWs@Hunt},
'   {aRWe@aHWn@Hob, aRWw@aHWn@Hob, aRWw@aHWn@Hunt, aRWe@aHWn@Hunt},
'   {aRWe@aHWs@Hob, aRWe@aHWn@Hob, aRWe@aHWn@Hunt, aRWe@aHWs@Hunt},
'   {aRWw@aHWn@Hob, aRWw@aHWs@Hob, aRWw@aHWs@Hunt,
aRWw@aHWn@Hunt}})

  aPgzob=PolygonZ.Make({{aRWw@aHWn@Hob,  aRWe@aHWn@Hob,
                  aRWe@aHWs@Hob,  aRWw@aHWs@Hob}})

  aPgzun=PolygonZ.Make({{aRWw@aHWn@Hunt, aRWe@aHWn@Hunt,
                  aRWe@aHWs@Hunt, aRWw@aHWs@Hunt}})

  aPgzs=PolygonZ.Make({{ aRWw@aHWs@Hob,  aRWe@aHWs@Hob,
                  aRWe@aHWs@Hunt, aRWw@aHWs@Hunt}})

  aPgzn=PolygonZ.Make({{ aRWe@aHWn@Hob,  aRWw@aHWn@Hob,
                  aRWw@aHWn@Hunt, aRWe@aHWn@Hunt}})

  aPgze=PolygonZ.Make({{ aRWe@aHWs@Hob,  aRWe@aHWn@Hob,
                  aRWe@aHWn@Hunt, aRWe@aHWs@Hunt}})

  aPgzw=PolygonZ.Make({{ aRWw@aHWn@Hob,  aRWw@aHWs@Hob,
                  aRWw@aHWs@Hunt, aRWw@aHWn@Hunt}})

  ListofPgz={aPgzob,aPgzun,aPgzs,aPgzn,aPgze,aPgzw}

  for each j in 0..5
    aShp=ListofPgz.Get(j)
    recNr=recNr+1
    PgzFTab.AddRecord
    PgzFTab.SetValue(ShpFld, recNr, aShp)
    PgzFTab.SetValue(IDFld, recNr, recNr)
    PgzFTab.SetValue(BoIDFld, recNr, BoId)
    PgzFTab.SetValue(RWFld, recNr, aRW)
    PgzFTab.SetValue(HWFld, recNr, aHW)
    PgzFTab.SetValue(HobFld, recNr, Hob)
    PgzFTab.SetValue(HuntFld, recNr, Hunt)
    PgzFTab.SetValue(SchichtFld, recNr, aText)
  end
```

```
end
PgzFTab.SetEditable(false)
theTheme=FTheme.Make(PgzFTab)
theSzene.AddTheme(theTheme)

'Veränderung der Legende der Bohrungen als Stapel-Arbeit
av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette
ListofFarbD = {"Deck", 53, "QohneD", 5, "QmitD", 20, "QohneAe", 32}

av.ShowMsg("Veränderung der Legende der Bohrungen...")
FTabP=theTheme.GetFTab
AnzRecP=0
for each aRecP in FTabP
   AnzRecP=AnzRecP+1
end

if (AnzRecP <> 0) then
  theLegend=theTheme.GetLegend
  theLegend.SetLegendType(#Legend_Type_Unique)
  theLegend.Unique(theTheme, "Schicht")
  ListofKlasse=theLegend.GetClassifications
  AnzKlasse=ListofKlasse.Count
  IdxKlasse=AnzKlasse-2
  'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
  aListofNr={}
  for each i in 0..IdxKlasse
     theKlasseLb=ListofKlasse.Get(i).GetLabel
     'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
     aLbIdx = ListofFarbD.FindByValue(theKlasseLb)
     if (aLbIdx <> -1) then
       aCNr = ListofFarbD.Get(aLbIdx + 1)
     elseif (aLbIdx = -1) then
       aCNr = 47
     end
     aListofNr.Add(aCNr)
  end
  'theLegend.SetNullValue("Schicht", "Deck")
  aListofSymbol=theLegend.GetSymbols
  AnzSymb=aListofSymbol.Count
  'MsgBox.Info(AnzSymb.AsString, "Anzahl der Symbole")
  AnzSymbIdx=AnzSymb-2
  aListofColor={}

  for each Nmb in 0..IdxKlasse
    aNumb=aListofNr.Get(Nmb)
    theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
    theRgbList=theColor.GetRgbList
    aColor=Color.Make
    aColor.SetRgbList(theRgbList)
    aListofColor.Add(aColor)
  end
  for each symb in 0..AnzSymbIdx
     aListofSymbol.Get(symb).SetColor(aListofColor.Get(symb))
  end
  theTheme.UpdateLegend
end
```

```
'pfbopthg.ave
'Bordaten werden aus einer dBase-Datenbank-Datei auf einer Festplatte
'eingelesen und als unterschiedliche geologische Punkte in einem aktiven
'Längsprofilschnitt-View gezeichnet.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc  'ein aktives Längsprofilschnitt-View

'Eingabe der dBase-Tabelle auf der Festplatte
theFileName=FileDialog.Show("*.dbf", "dBase-File",
        "Eingabe einer Datei für Bohrungen")
if (theFileName=nil) then exit end

theVtab=Vtab.Make(theFileName,false,false)

'Eine Tabelle gleich wie die Tabelle auf der Festplatte wird im Project
'für spätere Aufgabe hergestellt.
'theTable=Table.Make(theVtab)

'Feststellung der Anzahl der Datensätze in der Tabelle
AnzRec=0
for each rec in theVtab
   AnzRec=AnzRec+1
end
idxAnzRec=AnzRec-1
'MsgBox.Info("Anzahl der Datensätze in der Tabelle: "++(AnzRec).AsString,
'          "Kontrolle der Daten")

YFtStr=MsgBox.Input("zur Überhöhung der Höhe",
                "Eingabe eines Faktors", "50")
YFt=YFtStr.AsNumber

FldList=theVtab.GetFields
AnzFld=FldList.Count
idxAnzFld=AnzFld-1
'MsgBox.Info("Anzahl der Felder in der Tabelle:"++AnzFld.AsString,
'          "Kontrolle der Daten")
'MsgBox.ListAsString(FldList,  "Name der Felder", "Kontrolle der Daten")

theVtab.SetEditable(false)
ListofBNr={}
Listofshapes={}
ListofPetro={}
ListofBnrLb={}
ListofKoordLb={}
NRFld=theVtab.FindField("Nr")
HWFld=theVtab.FindField("Hw")
GHFld=MsgBox.ListAsString(FldList, "für Geländehöhe [m]",
    "Auswahl eines Feldes im"++theVtab.AsString)

for each aRec in theVTab
   aBNr=theVtab.ReturnValue(NRFld,aRec)
   aHW=theVtab.ReturnValue(HWFld,aRec)
   aGh=theVtab.ReturnValue(GHFld,aRec)
   aPt=theVtab.ReturnValue((FldList.Get(4)),aRec)
   ListofBNr.Add(aBNr)
   aGh50=aGh* YFt
   Listofshapes.Add(aHW@aGh50)
   ListofPetro.Add(aPt)
   SW="aus"
   ListofBnrLb.Add(aBNr)
```

```
    aLbh=(aGh+10)* YFt
    ListofKoordLb.Add(aHW@aLbh)
    for each aFld in 0..idxAnzFld
      if (aFld > 4) then
        if (FldList.Get(aFld).IsTypeNumber) then
          aTiefe=theVtab.ReturnValue((FldList.Get(aFld)),aRec)
          if (aTiefe <> 0) then
            aHoehe=aGh-aTiefe
            aHoe50=aHoehe* YFt
            ListofBNr.Add(aBNr)
            Listofshapes.Add(aHW@aHoe50)
            SW="an"
          elseif (aTiefe = 0) then
            SW="aus"
          end
        elseif (FldList.Get(aFld).IsTypeString) then
          if (SW = "an") then
            aPetro=theVtab.ReturnValue((FldList.Get(aFld)),aRec)
            ListofPetro.Add(aPetro)
          end
        end
      end
    end
end

'Ein Feature-Shape-File für Profilschnitte (Point) wird hergestellt.
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("Grmtrgz1","shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("Grmtrgz1","shp")

fName =FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName =nil) then exit end
fName.SetExtension("shp")
PtFTab=FTab.MakeNew(fName, Point)
ShapeField1= PtFTab.FindField("shape")
IDField1=Field.Make("ID1", #Field_Short, 4, 0)
BNrField1=Field.Make("Bohr-Nr", #Field_Byte, 3, 0)
PtField1=Field.Make("Petro", #Field_Char, 5, 0)
ListofFlds1={IDField1, BNrField1, PtField1}
PtFTab.AddFields(ListofFlds1)

av.ShowMsg("Herstellung des neuen Punkt-Themas ...")
av.ShowStopButton

AnzPoint=ListofBNr.Count
AnzShps=Listofshapes.Count
AnzPtr=ListofPetro.Count
'MsgBox.Report("Anzahl der Punkte:"+NL+NL+
'             "Bohr-Nr:"++"   "++AnzPoint.AsString+NL+
'             "Shapes:"++" "++AnzPShps.AsString+NL+
'             "Petrogr:"++"   "++AnzPtr.AsString,
'             "Kontrolle")

'Aus den Listen werden die Punkte hergestellt
IdxPoint=AnzPoint-1
recNr=-1
Ng=0
PtFTab.SetEditable(false)
PtFTab.SetEditable(true)

for each aPt in 0..IdxPoint
  Ng=Ng+1
```

```
  thePt=Listofshapes.Get(aPt)
  theId=aPt
  theBNr=ListofBNr.Get(aPt)
  thePtr=ListofPetro.Get(aPt)

  recNr=recNr+1
  PtFTab.AddRecord
  PtFTab.SetValue(ShapeField1, recNr, thePt)
  PtFTab.SetValue(IDField1, recNr, theId)
  PtFTab.SetValue(BNrField1, recNr, theBNr)
  PtFTab.SetValue(PtField1, recNr, thePtr)

  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzPoint*100)
  if (not more) then
    break
  end
end
PtFTab.SetEditable(false)
thmNew=FTheme.Make(PtFTab)
theView.AddTheme(thmNew)

'Veränderung der Legende des Themas im Profil
'Definition der Farbe für die Balken
ListofFarbD = {"Mu", 53, "Wb", 53, "aufB", 53, "Lehm", 47, "Ls", 47,
  "Ubr", 47, "Uff", 47, "Uf", 47, "Ut", 47, "Ufs", 47, "Ufst", 47,
  "USt", 47, "Ust", 47, "US", 47, "Bk", 11, "BkB", 11, "Bkt", 11,
  "BkT", 11, "Bkhz", 11, "TBks", 11, "TBK", 11, "Ubks", 11,
  "Tdbr", 11, "Ton", 20, "Tgr", 20, "Tbr", 20, "Tbk", 20, "Tons", 20,
  "Ts", 20, "Tsgr", 20, "TonS", 20, "sT", 20, "TU", 20, "Tufs", 20,
  "Tus", 20, "Tsu", 20, "TSG", 20, "Tsst", 20, "USmS", 17, "SMu", 17,
  "Sd", 17, "Sand", 17, "Sbr", 17, "Sgr", 17, "ST", 17, "St", 17,
  "STla", 17, "St'", 17, "St!", 17, "Su", 17, "Sut", 17, "Sut'", 17,
  "Sl", 17, "Sg'", 17, "SmTv", 17, "Smgs", 17, "LUS", 17, "fS", 17,
  "tfS", 17, "fSgr", 17, "fSt", 17, "fSt!", 17, "fST", 17, "fSl", 17,
  "fSu'", 17, "fSu", 17, "fSU", 17, "fSut", 17, "FSut", 17, "fSmS", 17,
  "fSms", 17, "fmS", 17, "fmSt", 17, "mS", 17, "mSbr", 17, "mSgr", 17,
  "mSra", 17, "mSs", 17, "mSl", 17, "mfS", 17, "mfSU", 17, "mfSu", 17,
  "mfST", 17, "mST", 17, "mSt", 17, "mSTe", 17, "mSu'", 17, "mSL", 17,
  "mSl", 17, "mSfs", 17, "mSgs", 17, "mSgl", 17, "fmgS", 17, "mgS", 17,
  "mgST", 17, "mgSU", 17, "tgS", 17, "gS", 17, "gSbr", 17, "gSfs", 17,
  "gST", 17, "gST'", 17, "gSU'", 17, "gmS", 17, "Tgr", 4, "TTr", 4,
  "Tgel", 4, "Tr", 4, "Tr?", 4, "fSTr", 4, "mSTr", 4, "KA", 1,
  "K.A.", 1, "", 1}

theTheme=thmNew

theLegend=theTheme.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(theTheme, "Petro")
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofNr={}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   aLbIdx = ListofFarbD.FindByValue(theKlasseLb)
   if (aLbIdx <> -1) then
     aCNrIdx = ListofFarbD.Get(aLbIdx + 1)
   elseif (aLbIdx = -1) then
```

```
      aRNr = i Mod 60
      if (aRNr = 0) then
        aCNrIdx = 1
      elseif (aRNr <> 0) then
        aCNrIdx = aRNr
      end
    end
    aListofNr.Add(aCNrIdx)
end

'theLegend.SetNullValue("Schicht", "Deck")
aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
'MsgBox.Info(AnzSymb.AsString, "Anzahl der Symbole")
AnzSymbIdx=AnzSymb-2
aListofColor={}
av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

for each Nmb in 0..IdxKlasse
  aNumb=aListofNr.Get(Nmb)
  theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
  theRgbList=theColor.GetRgbList
  aColor=Color.Make
  aColor.SetRgbList(theRgbList)
  aListofColor.Add(aColor)
end

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_Marker)
theMPalette=av.GetSymbolWin.GetPalette
theBasicMarker=(theMPalette.GetList(#PALETTE_LIST_Marker).Get(37))

for each symb in 0..AnzSymbIdx
   theSymbol=aListofSymbol.Get(symb)
   theSymbol.Copy(theBasicMarker)
   theSymbol.SetColor(aListofColor.Get(symb))
   theSymbol.SetSize(4)
end

theTheme.UpdateLegend
thrRecNr=ListofBnrLb.Count
IdxRecNr=thrRecNr-1
theGraphicList=theView.GetGraphics

for each aRecNr in 0..IdxRecNr
  theGString=ListofBnrLb.Get(aRecNr).AsString
  theGPoint=ListofKoordLb.Get(aRecNr)
  theGText=GraphicText.Make(theGString, theGPoint)
  theGText.SetAlignment(#TEXTCOMPOSER_JUST_CENTER)
  theTextSymbol=theGText.ReturnSymbols.Get(0)
  theTextSymbol.SetSize(6)
  newFont=Font.Make("Arial", "normal")
  theTextSymbol.SetFont(newFont)
  theTextSymbol.SetColor(Color.GetBlack)
  theGText.Invalidate
  theGraphicList.Add(theGText)
end
```

```
'pfbptapt.ave
'Profilschnitte mit mehreren Flächen werden als Punkte
'in einem aktiven Profilschnitt-View (evtl. einem Ereignis-Thema)
'in dem View hergestellt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc  'ein aktives Profilschnitt-View

ListofThms = theView.GetThemes
ListofPtThms = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofPtThms.Add(aT)
      end
   end
end

av.ShowMsg("Einagebe eines Punkt-Themas und"
     ++"Feststellung der Anzahl der Punkte in dem Thema ...")

PtTheme = MsgBox.ChoiceAsString(ListofPtThms,
      "um aus den Punkten einen Profilschnitt herzustellen.",
      "Auswahl eines Punkt-Themas")

PtFTab = PtTheme.GetFTab
ListofFlds = PtFTab.GetFields
MsgBox.ListAsString(ListofFlds, "die im Punkt-Thema"
      ++PtTheme.GetName++"enthalten sind", "Anzeige der Felder")

PtShpFld = ListofFlds.Get(0)

AnzHFldStr=MsgBox.Input("die die Höhen der Punkte enthalten",
        "Anzahl der Felder", "4")

if (AnzHFldStr = nil) then
  MsgBox.Info("Die Anzahl der Felder für Höhen"+NL+"im Thema"
      ++PtTheme.GetName+"ist noch festzustellen!", "Information")
  exit
end

AnzHFld=AnzHFldStr.AsNumber.SetFormat("d")
IdxHFld=AnzHFld-1
ListofHFlds={}
HStr=""
for each i in 0..IdxHFld
  Nr=i+1
  PtHFld = MsgBox.ListAsString(ListofFlds,
       "für"++Nr.AsString+". Höhe [m]"+NL+
       "bis jetzt eingegeben:"++HStr,
       "Auswahl eines Feldes im Thema"++PtTheme.GetName)
  ListofHFlds.Add(PtHFld)
  NHStr=PtHFld.GetAlias
  HStr=HStr++NHStr+";"
end
HFaktorStr=MsgBox.Input("zur vertikalen Überhöhung der Profilschnitte",
        "Eingabe eines Faktors", "50")
HFaktor=HFaktorStr.AsNumber

AnzPt=0
```

```
for each Pt in PtFTab
   AnzPt=AnzPt+1
end
AnzPtIdx=AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"
        ++PtTheme.AsString)

ListofListof2DPt={}
for each j in 0..IdxHFld
  aHFld=ListofHFlds.Get(j)
  Listof2DPt={}
  for each i in 0..AnzPtIdx
    aEntf = (PtFTab.ReturnValue(PtShpFld, i)).Getx
    aH = (PtFTab.ReturnValue(aHFld, i))*HFaktor
    Listof2DPt.Add(aEntf@aH)
  end
  ListofListof2DPt.Add(Listof2DPt)
end

av.ShowMsg("Speicherung der hergestellten 2D-PolyLine ...")
'Ein Feature-Shape-File für einen Profilschnitt (PolyLine) wird hergestellt.
theWDStr = theProject.GetWorkDir.AsString
DName = ("Pt"+(PtTheme.AsString.LCase)).Left(6)
fnStr=FileName.Make(theWDStr).MakeTmp(DName,"shp")
fnGrMt=FileDialog.Put(fnStr, "*.shp", "Output 2D shape File (PolyLine)")
if (fnGrMt=nil) then exit end
fnGrMt.SetExtension("shp")
GrMtFTab=FTab.MakeNew(fnGrMt, Point)
ShapeFld=GrMtFTab.FindField("shape")
IDFld=Field.Make("ID", #Field_LONG, 6, 0)
IDFLFld=Field.Make("IDFL", #Field_LONG, 4, 0)
NamenFld=Field.Make("Namen", #Field_Char, 10, 0)
ListofFlds1={IDFld, IDFLFld, NamenFld}
GrMtFTab.AddFields(ListofFlds1)

GrMtFTab.SetEditable(false)
GrMtFTab.SetEditable(true)
recNr = -1
for each j in 0..IdxHFld
  Listof2DPt = ListofListof2DPt.Get(j)
  AnzPt = Listof2DPt.Count
  IdxPt = AnzPt - 1
  aHFldStr = ListofHFlds.Get(j).AsString
  for each i in 0..IdxPt
    recNr = recNr + 1
    aPt = Listof2DPt.Get(i)
    GrMtFTab.AddRecord
    GrMtFTab.SetValue(ShapeFld, recNr, aPt)
    GrMtFTab.SetValue(IDFld, recNr, recNr)
    GrMtFTab.SetValue(IDFLFld, recNr, i)
    GrMtFTab.SetValue(NamenFld, recNr, aHFldStr)
  end
end
GrMtFTab.SetEditable(false)
thmNew=FTheme.Make(GrMtFTab)
theView.AddTheme(thmNew)
```

```
'pfq2bohr.ave
'Bohrungen werden in einem Profilschnitt-View als
'Polygone gezeichnet. Die Polygone für Bohrungen mit dem ersten
'Buchstaben von B liegen innerhalb 50 m entfernt von einem Querprofilschnitt
'und mit dem ersten Buchstaben von C innerhalb 100 m entfernt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
PfView=av.GetActiveDoc    'aktives Profilschnitt-View,
                    'wo die neuen Bohrungsdateien entstehen.

myScript=theProject.FindScript("pfq2bohr")
myScript.SetNumberFormat("d.dd")

ListofViews={"Kt1_gg", "Kt1_ggd", "Kt1_hg", "Kt1_hgd", "Kt1_tga", "Kt1_tgd",
          "Karten-View1", "Karten-View2", "Karten-View3"}

GP=MsgBox.ChoiceAsString(ListofViews,
  "Das View, wo sich die Bohrdaten befinden.","Auswahl eines Views")
theView=theProject.FindDoc(GP)

ListofThemes=theView.GetThemes
ListofPtFThm = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
       ListofPtFThm.Add(aT)
     end
   end
end
PTTheme=MsgBox.ChoiceAsString(ListofPtFThm,
 "das die Punkte für Bohrdaten enthält,"+NL+"um sie als Polygon zu zeichnen",
      "Eingabe eines Themas im View"++theView.AsString)

PTFTab=PTTheme.GetFTab
PTFields=PTFTab.GetFields
PTShpFld=MsgBox.ChoiceAsString(PTFields, "Feld für Punkt-Shape",
                    "Eingabe eines Feldes in"++PTTheme.AsString)
PTGHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Geländehöhe",
                    "Eingabe eines Feldes in"++PTTheme.AsString)
PTTHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Höhe der Terrassenoberkante",
                    "Eingabe eines Feldes in"++PTTheme.AsString)
PTQHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Quartärbasishöhe",
                   "Eingabe eines Feldes in"++PTTheme.AsString)
PTEHFld=MsgBox.ChoiceAsString(PTFields, "Feld für Bohrungsendhöhe",
                   "Eingabe eines Feldes in"++PTTheme.AsString)

ListofPTFld={PTShpFld, PTGHFld, PTTHFld, PTQHFld, PTEHFld}

av.ShowMsg("Herstellung der Polygone zur Auswahl der Bohrungen ...")
av.ShowStopButton

ListofPgA={}
ListofPgB={}
ListofIDHW={}
ListofHWNr={}

ListofRW = {"2558600.00", "2582450.00", "2563000.00", "2568000.00"}
ListofHW = {"5618450.00", "5641050.00", "5637000.00", "5641000.00"}
ListofAbst = {"50.00", "200.00"}
ListofFakt = {"50", "40"}
```

```
RWAnfStr00 = MsgBox.ListAsString(ListofRW,
        "Der kleinste Rechtswert des Profilschnittes",
        "Auswahl der Koordinaten")

RWEndStr00 = MsgBox.ListAsString(ListofRW,
        "Der größte Rechtswert des Profilschnittes",
        "Auswahl der Koordinaten")

VHWA00 = MsgBox.ListAsString(ListofHW,
        "Anfangs-HW in View"++PfView.AsString,
        "Auswahl der Koordinaten")

VHWE00 = MsgBox.ListAsString(ListofHW,
        "End-HW in View"++PfView.AsString,
        "Auswahl der Koordinaten")

PrfAbstStr00 = MsgBox.ListAsString(ListofAbst,
        "Der Abstand zwischen der Profilschnitte",
        "Eingabe der Profilschnitt-Daten")

HFaktorStr00 = MsgBox.ListAsString(ListofFakt,
        "Faktor zur Überhöhung der Höhen im Profilschnitt",
        "Eingabe der Profilschnitt-Daten")


RWAnfStr=MsgBox.Input("Der kleinste Rechtswert des Profilschnittes",
      "Änderungsmöglichkeit der Koordinaten", RWAnfStr00)
RWAnf=RWAnfStr.AsNumber
RWEndStr=MsgBox.Input("Der größte Rechtswert des Profilschnittes",
      "Änderungsmöglichkeit der Koordinaten", RWEndStr00)
RWEnd=RWEndStr.AsNumber
VHWA=MsgBox.Input("Anfangs-HW in View"++PfView.AsString,
      "Änderungsmöglichkeit der Koordinaten", VHWA00)
VHWANr=VHWA.AsNumber
VHWE=MsgBox.Input("End-HW in View"++PfView.AsString,
      "Änderungsmöglichkeit der Koordinaten", VHWE00)
VHWENr=VHWE.AsNumber

PrfAbstStr=MsgBox.Input("Der Abstand zwischen der Profilschnitte",
      "Änderungsmöglichkeit der Profilschnitt-Daten", PrfAbstStr00)
PrfAbst=PrfAbstStr.AsNumber
HFaktorStr=MsgBox.Input("Faktor zur Höhen-Überhöhung im Profilschnitt",
      "Änderungsmöglichkeit der Profilschnitt-Daten", HFaktorStr00)
HFaktor=HFaktorStr.AsNumber

AnzPg=((VHWENr-VHWANr)/PrfAbst+1).SetFormat("d")
AnzPgIdx=(AnzPg-1).SetFormat("d")
for each i in 0..AnzPgIdx
   N=i+1
   HWNr=VHWANr+(i*PrfAbst)

   HWobA=HWNr+50.00
   HWunA=HWNr-50.00
   HWobB=HWNr+100.00
   HWunB=HWNr-100.00

   IDHWNr=i
   thePolyGA=Polygon.Make({{RWAnf@HWobA, RWEnd@HWobA,
                    RWEnd@HWunA, RWAnf@HWunA}})
   ListofPgA.Add(thePolyGA)
   thePolyGB=Polygon.Make({{RWAnf@HWobB, RWEnd@HWobB,
```

```
                    RWEnd@HWunB, RWAnf@HWunB}})
    ListofPgB.Add(thePolyGB)
    ListofIDHW.Add(IDHWNr)
    ListofHWNr.Add(HWNr)

    'Show percentage complete with enabled stop button
    more=av.SetStatus(N/AnzPg*100)
    if (not more) then
      break
    end
    if (HWNr > VHWENr) then
      break
    end
end

av.ShowMsg("Auswahl der Bohrungen ...")
av.ShowStopButton

AnzPg=ListofPgA.Count
AnzPgIdx=(AnzPg-1).SetFormat("d")
ListofPTFld={PTShpFld, PTGHFld, PTTHFld, PTQHFld, PTEHFld}
WDStr=theProject.GetWorkDir.AsString

for each i in 0..AnzPgIdx
  N=i+1
  HWInt=ListofHWNr.Get(i).Truncate.SetFormat("").SetFormat("d")

  'Ein Feature-Shape-File für Bohrungen (Polygon) wird hergestellt.
  fnBStr="B"+(HWInt).AsString+".shp"
  fnBPg=FileName.Make(WDStr+fnBStr)
  'fnBPg=FileName.Make("C:\Verz1\Verz2\Verz3\"+fnBStr)

  FTabBPg=FTab.MakeNew(fnBPg, Polygon)
  ShapeFldpg=FTabBPg.FindField("shape")
  RWFldpg=Field.Make("RW", #Field_Float, 10, 2)
  HWFldpg=Field.Make("HW", #Field_Float, 10, 2)
  IDHWFldpg=Field.Make("IDHW", #Field_Short, 4, 0)
  HWPfFldpg=Field.Make("HW", #Field_Float, 10, 2)
  LgFldPg=Field.Make("Lage", #Field_Float, 10, 2)
  AWFldpg=Field.Make("Abstand", #Field_Short, 4, 0)
  SchichtFldpg=Field.Make("Schicht", #Field_Char, 7, 0)

  ListofB1={RWFldpg, HWFldpg, IDHWFldpg, HWPfFldpg,
          LgFldPg, AWFldpg, SchichtFldpg}
  FTabBPg.AddFields(ListofB1)
  ListofB2={ShapeFldpg, RWFldpg, HWFldpg, IDHWFldpg, HWPfFldpg,
          LgFldPg, AWFldpg, SchichtFldpg}

  FTabBPg.SetEditable(false)
  FTabBPg.SetEditable(true)

  recNr=-1
  thePgA=ListofPgA.Get(i)
  theHWPg=ListofHWNr.Get(i)
  theIDHW=ListofIDHW.Get(i)

  PTFTab.SelectByPolygon(thePgA, #VTAB_SELTYPE_NEW)
  if (PTFTab.GetSelection <> nil) then
    for each rec in PTFTab.GetSelection
      theShape=PTFTab.ReturnValue(PTShpFld, rec)
      theRW=theShape.Getx
      theHW=theShape.Gety
```

```
theRWl=theRW-50
theRWr=theRW+50
theGH=(PTFTab.ReturnValue(PTGHFld, rec))*HFaktor
theTH=(PTFTab.ReturnValue(PTTHFld, rec))*HFaktor
theQH=(PTFTab.ReturnValue(PTQHFld, rec))*HFaktor
theEH=(PTFTab.ReturnValue(PTEHFld, rec))*HFaktor
theLage=theHW-HWInt
theAbst=theLage.Abs.SetFormat("").SetFormat("d")

if (theTH = 0) then
  thePg=Polygon.Make({{theRWl@theGH, theRWr@theGH,
                theRWr@theQH, theRWl@theQH}})
  recNr=recNr+1
  FTabBPg.AddRecord
  FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
  FTabBPg.SetValue(RWFldpg, recNr, theRW)
  FTabBPg.SetValue(HWFldpg, recNr, theHW)
  FTabBPg.SetValue(IDHWFldpg, recNr, theIDHW)
  FTabBPg.SetValue(HWPfFldpg, recNr, theHWPg)
  FTabBPg.SetValue(LgFldpg, recNr, theLage)
  FTabBPg.SetValue(AWFldpg, recNr, theAbst)
  FTabBPg.SetValue(SchichtFldpg, recNr, "QohneD")

  if (theEH < theQH) then
    if (theEH < 0) then
      theEH=(10*HFaktor)
    end
    thePg=Polygon.Make({{theRWl@theQH, theRWr@theQH,
                  theRWr@theEH, theRWl@theEH}})
    recNr=recNr+1
    FTabBPg.AddRecord
    FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
    FTabBPg.SetValue(RWFldpg, recNr, theRW)
    FTabBPg.SetValue(HWFldpg, recNr, theHW)
    FTabBPg.SetValue(IDHWFldpg, recNr, theIDHW)
    FTabBPg.SetValue(HWPfFldpg, recNr, theHWPg)
    FTabBPg.SetValue(LgFldpg, recNr, theLage)
    FTabBPg.SetValue(AWFldpg, recNr, theAbst)
    FTabBPg.SetValue(SchichtFldpg, recNr, "TrorAe")
  end
elseif (theTH <> 0) then
  thePg=Polygon.Make({{theRWl@theGH, theRWr@theGH,
                theRWr@theTH, theRWl@theTH}})
  recNr=recNr+1
  FTabBPg.AddRecord
  FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
  FTabBPg.SetValue(RWFldpg, recNr, theRW)
  FTabBPg.SetValue(HWFldpg, recNr, theHW)
  FTabBPg.SetValue(IDHWFldpg, recNr, theIDHW)
  FTabBPg.SetValue(HWPfFldpg, recNr, theHWPg)
  FTabBPg.SetValue(LgFldpg, recNr, theLage)
  FTabBPg.SetValue(AWFldpg, recNr, theAbst)
  FTabBPg.SetValue(SchichtFldpg, recNr, "Deck")

  if (theQH = 0) then
    if (theEH < theTH) then
      if (theEH < 0) then
        theEH=(10*HFaktor)
      end
      thePg=Polygon.Make({{theRWl@theTH, theRWr@theTH,
                    theRWr@theEH, theRWl@theEH}})
      recNr=recNr+1
```

```
            FTabBPg.AddRecord
            FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
            FTabBPg.SetValue(RWFldpg, recNr, theRW)
            FTabBPg.SetValue(HWFldpg, recNr, theHW)
            FTabBPg.SetValue(IDHWFldpg, recNr, theIDHW)
            FTabBPg.SetValue(HWPfFldpg, recNr, theHWPg)
            FTabBPg.SetValue(LgFldpg, recNr, theLage)
            FTabBPg.SetValue(AWFldpg, recNr, theAbst)
            FTabBPg.SetValue(SchichtFldpg, recNr, "QohneT")
          end
        elseif (theQH <> 0) then
          thePg=Polygon.Make({{theRWl@theTH, theRWr@theTH,
                              theRWr@theQH, theRWl@theQH}})
          recNr=recNr+1
          FTabBPg.AddRecord
          FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
          FTabBPg.SetValue(RWFldpg, recNr, theRW)
          FTabBPg.SetValue(HWFldpg, recNr, theHW)
          FTabBPg.SetValue(IDHWFldpg, recNr, theIDHW)
          FTabBPg.SetValue(HWPfFldpg, recNr, theHWPg)
          FTabBPg.SetValue(LgFldpg, recNr, theLage)
          FTabBPg.SetValue(AWFldpg, recNr, theAbst)
          FTabBPg.SetValue(SchichtFldpg, recNr, "QmitD")

          if (theEH < theQH) then
            if (theEH < 0) then
              theEH=(10*HFaktor)
            end
            thePg=Polygon.Make({{theRWl@theQH, theRWr@theQH,
                                 theRWr@theEH, theRWl@theEH}})
            recNr=recNr+1
            FTabBPg.AddRecord
            FTabBPg.SetValue(ShapeFldpg, recNr, thePg)
            FTabBPg.SetValue(RWFldpg, recNr, theRW)
            FTabBPg.SetValue(HWFldpg, recNr, theHW)
            FTabBPg.SetValue(IDHWFldpg, recNr, theIDHW)
            FTabBPg.SetValue(HWPfFldpg, recNr, theHWPg)
            FTabBPg.SetValue(LgFldpg, recNr, theLage)
            FTabBPg.SetValue(AWFldpg, recNr, theAbst)
            FTabBPg.SetValue(SchichtFldpg, recNr, "TrorAe")
          end
        end
      end
    end
  end

  FTabBPg.SetEditable(false)
  thmNew=FTheme.Make(FTabBPg)
  PfView.AddTheme(thmNew)

'Ein Feature-Shape-File für Profilschnitte (Polygon) wird hergestellt.

fnCStr="C"+(HWInt).AsString+".shp"
fnCPg=FileName.Make(WDStr+fnCStr)
'fnCPg=FileName.Make("C:\Verz1\Verz2\Verz1\"+fnCStr)

FTabCPg=FTab.MakeNew(fnCPg, Polygon)
ShapeFldpg=FTabCPg.FindField("shape")
RWFldpg=Field.Make("RW", #Field_Float, 10, 2)
HWFldpg=Field.Make("HW", #Field_Float, 10, 2)
IDHWFldpg=Field.Make("IDHW", #Field_Short, 4, 0)
HWPfFldpg=Field.Make("HW", #Field_Float, 10, 2)
```

```
LgFldPg=Field.Make("Lage", #Field_Float, 10, 2)
AWFldpg=Field.Make("Abstand", #Field_Short, 4, 0)
SchichtFldpg=Field.Make("Schicht", #Field_Char, 7, 0)

ListofB1={RWFldpg, HWFldpg, IDHWFldpg, HWPfFldpg,
        LgFldPg, AWFldpg, SchichtFldpg}
FTabCPg.AddFields(ListofB1)
ListofB2={ShapeFldpg, RWFldpg, HWFldpg, IDHWFldpg, HWPfFldpg,
        LgFldPg, AWFldpg, SchichtFldpg}

FTabCPg.SetEditable(false)
FTabCPg.SetEditable(true)

recNr=-1

thePgB=ListofPgB.Get(i)

PTFTab.SelectByPolygon(thePgB, #VTAB_SELTYPE_NEW)
if (PTFTab.GetSelection <> nil) then
  for each rec in PTFTab.GetSelection
     theShape=PTFTab.ReturnValue(PTShpFld, rec)
     theRW=theShape.Getx
     theHW=theShape.Gety
     theRWl=theRW-50
     theRWr=theRW+50
     theGH=(PTFTab.ReturnValue(PTGHFld, rec))*HFaktor
     theTH=(PTFTab.ReturnValue(PTTHFld, rec))*HFaktor
     theQH=(PTFTab.ReturnValue(PTQHFld, rec))*HFaktor
     theEH=(PTFTab.ReturnValue(PTEHFld, rec))*HFaktor
     theLage=theHW-HWInt
     theAbst=theLage.Abs.SetFormat("").SetFormat("d")

     if (theTH = 0) then
       thePg=Polygon.Make({{theRWl@theGH, theRWr@theGH,
                    theRWr@theQH, theRWl@theQH}})
       recNr=recNr+1
       FTabCPg.AddRecord
       FTabCPg.SetValue(ShapeFldpg, recNr, thePg)
       FTabCPg.SetValue(RWFldpg, recNr, theRW)
       FTabCPg.SetValue(HWFldpg, recNr, theHW)
       FTabCPg.SetValue(IDHWFldpg, recNr, theIDHW)
       FTabCPg.SetValue(HWPfFldpg, recNr, theHWPg)
       FTabCPg.SetValue(LgFldpg, recNr, theLage)
       FTabCPg.SetValue(AWFldpg, recNr, theAbst)
       FTabCPg.SetValue(SchichtFldpg, recNr, "QohneD")

       if (theEH < theQH) then
         if (theEH < 0) then
           theEH=(10*HFaktor)
         end
         thePg=Polygon.Make({{theRWl@theQH, theRWr@theQH,
                      theRWr@theEH, theRWl@theEH}})
         recNr=recNr+1
         FTabCPg.AddRecord
         FTabCPg.SetValue(ShapeFldpg, recNr, thePg)
         FTabCPg.SetValue(RWFldpg, recNr, theRW)
         FTabCPg.SetValue(HWFldpg, recNr, theHW)
         FTabCPg.SetValue(IDHWFldpg, recNr, theIDHW)
         FTabCPg.SetValue(HWPfFldpg, recNr, theHWPg)
         FTabCPg.SetValue(LgFldpg, recNr, theLage)
         FTabCPg.SetValue(AWFldpg, recNr, theAbst)
         FTabCPg.SetValue(SchichtFldpg, recNr, "TrorAe")
```

```
  end
elseif (theTH <> 0) then
  thePg=Polygon.Make({{theRWl@theGH, theRWr@theGH,
                theRWr@theTH, theRWl@theTH}})
  recNr=recNr+1
  FTabCPg.AddRecord
  FTabCPg.SetValue(ShapeFldpg, recNr, thePg)
  FTabCPg.SetValue(RWFldpg, recNr, theRW)
  FTabCPg.SetValue(HWFldpg, recNr, theHW)
  FTabCPg.SetValue(IDHWFldpg, recNr, theIDHW)
  FTabCPg.SetValue(HWPfFldpg, recNr, theHWPg)
  FTabCPg.SetValue(LgFldpg, recNr, theLage)
  FTabCPg.SetValue(AWFldpg, recNr, theAbst)
  FTabCPg.SetValue(SchichtFldpg, recNr, "Deck")

  if (theQH = 0) then
    if (theEH < theTH) then
      if (theEH < 0) then
        theEH=(10*HFaktor)
      end
      thePg=Polygon.Make({{theRWl@theTH, theRWr@theTH,
                    theRWr@theEH, theRWl@theEH}})
      recNr=recNr+1
      FTabCPg.AddRecord
      FTabCPg.SetValue(ShapeFldpg, recNr, thePg)
      FTabCPg.SetValue(RWFldpg, recNr, theRW)
      FTabCPg.SetValue(HWFldpg, recNr, theHW)
      FTabCPg.SetValue(IDHWFldpg, recNr, theIDHW)
      FTabCPg.SetValue(HWPfFldpg, recNr, theHWPg)
      FTabCPg.SetValue(LgFldpg, recNr, theLage)
      FTabCPg.SetValue(AWFldpg, recNr, theAbst)
      FTabCPg.SetValue(SchichtFldpg, recNr, "QohneT")
    end
  elseif (theQH <> 0) then
    thePg=Polygon.Make({{theRWl@theTH, theRWr@theTH,
                  theRWr@theQH, theRWl@theQH}})
    recNr=recNr+1
    FTabCPg.AddRecord
    FTabCPg.SetValue(ShapeFldpg, recNr, thePg)
    FTabCPg.SetValue(RWFldpg, recNr, theRW)
    FTabCPg.SetValue(HWFldpg, recNr, theHW)
    FTabCPg.SetValue(IDHWFldpg, recNr, theIDHW)
    FTabCPg.SetValue(HWPfFldpg, recNr, theHWPg)
    FTabCPg.SetValue(LgFldpg, recNr, theLage)
    FTabCPg.SetValue(AWFldpg, recNr, theAbst)
    FTabCPg.SetValue(SchichtFldpg, recNr, "QmitD")

    if (theEH < theQH) then
      if (theEH < 0) then
        theEH=(10*HFaktor)
      end
      thePg=Polygon.Make({{theRWl@theQH, theRWr@theQH,
                    theRWr@theEH, theRWl@theEH}})
      recNr=recNr+1
      FTabCPg.AddRecord
      FTabCPg.SetValue(ShapeFldpg, recNr, thePg)
      FTabCPg.SetValue(RWFldpg, recNr, theRW)
      FTabCPg.SetValue(HWFldpg, recNr, theHW)
      FTabCPg.SetValue(IDHWFldpg, recNr, theIDHW)
      FTabCPg.SetValue(HWPfFldpg, recNr, theHWPg)
      FTabCPg.SetValue(LgFldpg, recNr, theLage)
      FTabCPg.SetValue(AWFldpg, recNr, theAbst)
```

```
              FTabCPg.SetValue(SchichtFldpg, recNr, "TrorAe")
            end
          end
        end
      end
    end
    FTabCPg.SetEditable(false)
    thmNew=FTheme.Make(FTabCPg)
    PfView.AddTheme(thmNew)

    'Show percentage complete with enabled stop button
    more=av.SetStatus(N/AnzPg*100)
    if (not more) then
      break
    end
end

'Veränderung der Legende der Bohrungen als Stapel-Arbeit
av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

AnfHW=VHWANr
EndHW=VHWENr
AnzHW=((EndHW-AnfHW)/PrfAbst+1).SetFormat("d")
AnzHWIdx=(AnzHW-1).SetFormat("d")

av.ShowMsg("Veränderung der Legende der Bohrungen...")
av.ShowStopButton

ListofB1={"C", "B"}
ListofFarbD = {"Deck", 53, "QohneD", 26, "QmitD", 20, "QohneT", 14}
For each Thm in 0..AnzHWIdx
  Ng=Thm+1
  aHW=AnfHW+(Thm*PrfAbst)
  aHWInt=aHW.SetFormat("").SetFormat("d")

  for each Bst in ListofB1
     afnStr=Bst+(aHWInt).AsString+".shp"
     theTheme=PfView.FindTheme(afnStr)
     FTabP=theTheme.GetFTab
     AnzRecP=0
     for each aRecP in FTabP
        AnzRecP=AnzRecP+1
     end
     if (AnzRecP <> 0) then

     theLegend=theTheme.GetLegend
     theLegend.SetLegendType(#Legend_Type_Unique)
     theLegend.Unique(theTheme, "Schicht")
     ListofKlasse=theLegend.GetClassifications
     AnzKlasse=ListofKlasse.Count
     IdxKlasse=AnzKlasse-2
     'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
     aListofNr={}
     for each i in 0..IdxKlasse
        theKlasseLb=ListofKlasse.Get(i).GetLabel
        'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
        aLbIdx = ListofFarbD.FindByValue(theKlasseLb)
        if (aLbIdx <> -1) then
          aCNr = ListofFarbD.Get((aLbIdx + 1))
        elseif (aLbIdx = -1) then
          aCNr = 2
```

```
        end
        aListofNr.Add(aCNr)
      end

      'theLegend.SetNullValue("Schicht", "Deck")
      aListofSymbol=theLegend.GetSymbols
      AnzSymb=aListofSymbol.Count
      'MsgBox.Info(AnzSymb.AsString, "Anzahl der Symbole")
      AnzSymbIdx=AnzSymb-2
      aListofColor={}

      for each Nmb in 0..IdxKlasse
        aNumb=aListofNr.Get(Nmb)
        theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
        theRgbList=theColor.GetRgbList
        aColor=Color.Make
        aColor.SetRgbList(theRgbList)
        aListofColor.Add(aColor)
      end

      for each symb in 0..AnzSymbIdx
        aListofSymbol.Get(symb).SetColor(aListofColor.Get(symb))
      end

      theTheme.UpdateLegend
      end
    end
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzHW*100)
    if (not more) then
      break
    end
  end



'pfq2d1fl.ave
'2D-Querprofilschnitte einer geologischen Fläche werden
'für die gesamten Hochwerte mit einem bestimmten Abstand
'in einem aktiven Querprofilschnitt-View hergestellt.
'Die Daten werden aus einem Thema mit 3D-Querprofilschnittlinien
'einer geologischen Fläche in einem Karten-View geholt.
'Ein Menü in einem aktiven Querprofilschnitt-View zum Anklicken.

theProject=av.GetProject
ProfilView=av.GetActiveDoc  'ein aktives Querprofilschnitt-View
myScript=theProject.FindScript("pfq2d1fl")
myScript.SetNumberFormat( "d.dd") ' script default

ListofKtViews={"Kt1_gg", "Kt1_ggd", "Kt1_hg", "Kt1_hgd", "Kt1_tga", "Kt1_tgd"}
theViewStr=MsgBox.ChoiceAsString(ListofKtViews,
          "Der Name eines Karten-Views, welches"
          +NL+"das 3D-Querprofilschnittlinien-Thema enthält",
          "Eingabe eines Karten-Views")
theKtView=theProject.FindDoc(theViewStr)

'Ein Feature-Shape-File (Querprofilschnitt-PolyLineZ)
'für eine geologische Schicht wird geöffnet.
ListofThemen=theKtView.GetThemes
ListofPLThm = {}
for each i in ListofThemen
```

```
   if (i.Is(FTheme)) then
   aFTab = i.GetFtab
     if (aFtab.GetShapeClass.IsSubclassOf(Polyline)) then
       ListofPLThm.Add(i)
     end
   end
end

theThm=MsgBox.ChoiceAsString(ListofPLThm,
          "Der Name der Datei im View"++theKtView.AsString,
          "Auswahl einer Datei (3D-PolyLineZ)")
PLFTab=theThm.GetFTab
PLShpFld=PLFTab.FindField("Shape")
PLIDFld=PLFTab.FindField("IDHW")
PLHWFld=PLFTab.FindField("HW")
PLRWAnfFld=PLFTab.FindField("RWAnf")
PLRWEndFld=PLFTab.FindField("RWEnd")
Listof3DPLFlds={PLShpFld, PLIDFld, PLHWFld, PLRWAnfFld, PLRWEndFld}

'Feststellung der Anzahl der 3D-PolyLineZ ohne Geologie im Thema
oG3DPL=0
for each rec in PLFTab
   oG3DPL=oG3DPL+1
end
oG3DPLIdx=oG3DPL-1
MsgBox.Info(oG3DPL.AsString,
  "Anzahl der 3D-PolyLineZ im Thema"++theThm.AsString)

'Verzeichnis der Querprofilschnitte auf Festplatte,
'die durch dieses Programm entstehen

theWD = av.GetProject.GetWorkDir
theSV = theWD.AsString
MsgBox.Info(theSV, "Verzeichnis der QProfile auf Festplatte")

ListofEB={"G","H","K","L","M","N","P","Q","S","T","U","V","W","X"}

EB=MsgBox.ChoiceAsString(ListofEB,
   "für die Namen der neuen Querprofilschnitte",
   "Auswahl eines ersten Buchstabens")

FtStr=MsgBox.Input("für Überhöhung des Querprofilschnittes",
    "Eingabe eines Faktors","50")
Ft=FtStr.AsNumber

ListofGeoBez = {"Geländeoberfläche", "Terrassenoberkante",
             "NT abgedeckte Fläche", "Quartärbasisfläche"}

aKW = MsgBox.ChoiceAsString(ListofGeoBez,
   "um die neuen Querprofilschnitte zu bezeichnen",
   "Auswahl einer geologischen Fläche")
IdxKW = ListofGeoBez.Find(aKW)

ListofKW = {"GH","NT","NA","NQ"}
GeoStr = ListofKW.Get(IdxKW)

av.ShowMsg("Herstellung der 3D-Quer-Profilschnitte in View"
        ++ProfilView.AsString)
av.ShowStopButton

ListofColorNm={"dunkel-grün", "dunkel-violett", "dunkel-braun",
             "dunkel-blau", "schwarz", "rot"}
```

```
ListofColorNr={17, 35, 53, 22, 5, 9}

ColorNm=MsgBox.ChoiceAsString(ListofColorNm,
  "die Farbe im Symbolwindow für Querprofilschnitte",
  "Auswahl einer Farbe in der RgbList")
aColorIdx=ListofColorNm.FindByValue (ColorNm)
aColorNr=ListofColorNr.Get(aColorIdx)

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

theColor=(thePalette.GetList(#PALETTE_LIST_COLOR).Get(aColorNr))
theRgbList=theColor.GetRgbList
aColor=Color.Make
aColor.SetRgbList(theRgbList)

for each i in 0..oG3DPLIdx
  ListofEbene={}
  ListofList={}
  Ng=i+1
  ListIdx1 = -1

  theHW=PLFTab.ReturnValue(PLHWFld, i)
  HWInt=theHW.SetFormat("d")

  fnStr=EB+(HWInt).AsString+".shp"
  fName=FileName.Make(theSV+fnStr)
  PrfFTab=FTab.MakeNew(fName, PolyLine)
  PrfShpFld=PrfFTab.FindField("shape")
  PrfIDHWFld=Field.Make("IDHW", #Field_Short, 4, 0)
  PrfHWFld=Field.Make("HW", #Field_Float, 10, 2)
  PrfRWAnfFld=Field.Make("RWAnf", #Field_Float, 10, 2)
  PrfRWEndFld=Field.Make("RWEnd", #Field_Float, 10, 2)
  PrfEbeneFld=Field.Make("Flaeche", #Field_Char, 2, 0)
  Listof2DPrfFlds={PrfIDHWFld, PrfHWFld, PrfRWAnfFld, PrfRWEndFld, PrfEbeneFld}
  PrfFTab.AddFields(Listof2DPrfFlds)

  PrfFTab.SetEditable(false)
  PrfFTab.SetEditable(true)

  theShape=PLFTab.ReturnValue(PLShpFld, i)
  the3DPLList={}
  the3DPLList.Add({theShape})
  ListofList.Add(the3DPLList)
  ListofEbene.Add(GeoStr)
  ListIdx1=ListIdx1+1

  theRWAnf=PLFTab.ReturnValue(PLRWAnfFld, i)
  theRWEnd=PLFTab.ReturnValue(PLRWEndFld, i)

  '3D-PolyLineZ wird in Liste der Koordinaten umgewandelt.
  for each j in 0..ListIdx1
    theProfilList=ListofList.Get(j)
    if (theProfilList <> 0) then
      for each q in theProfilList
        theLines=q.Get(0).AsList
        for each m in theLines
          Listofx={}
          'Listofy={}
          Listofz={}
          for each ptx in m
            myx=ptx.Getx
```

```
                'myy=ptx.Gety
                myz=ptx.Getz
                Listofx.Add(myx)
                'Listofy.Add(myy)
                Listofz.Add(myz)
              end
            end
          end
        end
        Anz=Listofx.Count
        Index=Anz-1
        ListofPoint={}
        for each k in 0..Index
          xkrd=Listofx.Get(k)
          zkrd=(Listofz.Get(k))*Ft
          ListofPoint.Add(xkrd@zkrd)
        end
        ListofListofPoint={}
        ListofListofPoint.Add(ListofPoint)
        thePolyLine=PolyLine.Make(ListofListofPoint)
        recNr=0
        PrfFTab.AddRecord
        PrfFTab.SetValue(PrfShpFld, recNr, thePolyLine)
        PrfFTab.SetValue(PrfIDHWFld, recNr, recNr)
        PrfFTab.SetValue(PrfHWFld, recNr, theHW)
        PrfFTab.SetValue(PrfRWAnfFld, recNr, theRWAnf)
        PrfFTab.SetValue(PrfRWEndFld, recNr, theRWEnd)
        theEbene=ListofEbene.Get(j)
        PrfFTab.SetValue(PrfEbeneFld, recNr, theEbene)
      end

      PrfFTab.SetEditable(false)
      thmNew=FTheme.Make(PrfFTab)
      ProfilView.AddTheme(thmNew)

      theTheme=thmNew
      theLegend=theTheme.GetLegend
      theLegend.SingleSymbol
      theLegend.GetSymbols.Get(0).SetColor(aColor)
      theTheme.UpdateLegend

      'Show percentage complete with enabled stop button
      more=av.SetStatus(Ng/oG3DPL*100)
      if (not more) then
        break
      end
    end
```

```
'pfq2d4fl.ave
'Aus 3D-Querprofilschnittlinien in mehren Themen der gologischen Flächen
'in einem Karten-View werden 2D-Querprofilschnitte (PolyLine) in einem aktiven
'Querprofilschnitt-View hergestellt. Jedes neue 2D-Querprofilschnitt-Thema
'enthält alle geologischen Flächen an einer Querprofilschnittlinie im Karten-View.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
```

```
ProfilView=av.GetActiveDoc  'ein aktives View für alle Querprofilschnitte,
                    'welche mit diesem Script neu hergestellt werden.
myScript=theProject.FindScript("pfq2d4fl")
myScript.SetNumberFormat( "d.dd") ' script default

ListofKtViews={"Kt1_gg", "Kt1_ggd", "Kt1_hg",
            "Kt1_hgd", "Kt1_tga", "Kt1_tgd"}
theViewStr=MsgBox.ChoiceAsString(ListofKtViews,
         "das die 3D-Profilschnittlinien-Themen enthält",
         "Eingabe eines Karten-Views")
theKtView=theProject.FindDoc(theViewStr)
ListofSchichten={"Geländeoberfläche", "Basisfläche der Deckschichten (TOK)",
              "NT abgedeckte Fläche", "Quartärbasis"}

ListofKW={"GH", "TOK", "NTabF", "QbF"}
'Anzahl der geologischen Flächen
AnzScht= ListofSchichten.Count
IdxScht= AnzScht-1

'Feature-Shape-Dateien (PolyLineZ) werden im Karten-View geöffnet.
ListofKtThms=theKtView.GetThemes
ListofPLThms={}
ListofFTabs={}
ListofListof3DFlds={}

for each aTh in 0.. IdxScht
   NrStr=((aTh+1).SetFormat("").SetFormat("d")).AsString
   theScht=MsgBox.ChoiceAsString(ListofSchichten,
         "Geologie des"++NrStr++". Datensatzes",
         "Auswahl einer Fläche der Querprofilschnittlinien")
   PLThm=MsgBox.ChoiceAsString(ListofKtThms,
        "Eingabe eines Themas für 3D-Querprofilschnittlinien"
        ++"im Karten-View"++ theKtView.AsString,
        NrStr++". PolyLineZ-Thema für "++theScht.AsString)
   FTab0=PLThm.GetFTab
   Listof3DFlds=FTab0.GetFields
   ListofPLThms.Add(PLThm)
   ListofFTabs.Add(FTab0)
   ListofListof3DFlds.Add(Listof3DFlds)
end

'Definition der Farbe und des Musters der Legende
'Bezeichnung der Datensätze

aListofdfm = {"Deck",53,  "D",53, "De",53, "NT",14, "MT",26,
    "HT",32, "Präq",3, "Präm",3,  "GH",53, "NA",5, "NQ",8,
    "TOK",14, "NTabF",5, "MTabF",8, "QbF",8,

    "Geländeoberfläche",53, "Basisfläche der Deckschichten (TOK)",14,
    "NT abgedeckte Fläche",5, "MT abgedeckte Fläche",8,
    "Quartärbasis",8, "Oberfläche der als MT ältere Schichten",8,
    "Oberfläche der Präquartär-Schichten",8, "Deckschichten",53,
    "Niederterrassen",14, "Mittelterrassen",26,
    "Präquartäre Schichten",3}

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette = av.GetSymbolWin.GetPalette

FTab1=ListofFTabs.Get(0)
'Feststellung der Anzahl der 3D-PolyLine in dem Thema
oG3DPL=0
for each rec in FTab1
```

```
    oG3DPL=oG3DPL+1
end
oG3DPLIdx=oG3DPL-1
PLThm1=ListofPLThms.Get(0)
MsgBox.Info(oG3DPL.AsString, "Anzahl der 3D-PolyLine in dem Thema"
      ++PLThm1.AsString)

'Verzeichnis der Profilschnitte auf Festplatte,
'die durch dieses Programm entstehen, als Sourceverzeichnis

fName00=FileDialog.Show("*.shp", "shape file",
          "Anzeige einer Datei auf Festplatte,"++
           "um Verzeichnis festzustellen")
fNameStr = fName00.AsString
if (fNameStr <> Nil) then
  MsgBox.Info(fNameStr, "Der Name einer Datei mit Verzeichnis")
  Frg1=MsgBox.YesNo("Hat der Name auch einen Dateiname"
      ++"außer Verzeichnis?", "Kontrolle der Daten", True)
  if (Frg1) then
    ListoffNameStr=fNameStr.AsTokens("\")
    AnzfNameStr=ListoffNameStr.Count
    theIdxfNameStr=AnzfNameStr-1
    thefNameStr=ListoffNameStr.Get(theIdxfNameStr)
    MsgBox.Info(thefNameStr, "Der Name einer Datei")
    fNameStr.Substitute (thefNameStr, "")
  end
else
  fNameDefault="C:\Yda\Ganzesg\Demodat2\"
  fNameStr=MsgBox.Input("Verzeichnis der Querprofilschnitte"
        ++"auf Festplatte", "Eingabe der Daten", fNameDefault)
end
MsgBox.Info(fNameStr,"Verzeichnis der Querprofilschnitte auf Festplatte")

Frg2=MsgBox.YesNo("Ist der Name des Verzeichnises richtig?",
              "Kontrolle der Daten", True)
if (Frg2) then
  theSV=fNameStr
else
  theSV=MsgBox.Input("die durch dieses Programm entstehen",
       "Verzeichnis der Profilschnitte auf Festplatte,", fNameStr)
end
MsgBox.Info(theSV,"Verzeichnis der Querprofilschnitte auf Festplatte")

ListofEB={"V", "P"}
EB=MsgBox.ChoiceAsString(ListofEB, " der neuen Querprofilschnitte,",
                   "Auswahl eines ersten Buchstabens")

FtStr=MsgBox.Input("zur Überhöhung des Profilschnittes",
     "Eingabe des Faktors","50")
Ft=FtStr.AsNumber

av.ShowMsg("Herstellung der 3D-Profilschnitte in View"
        ++ProfilView.AsString)
av.ShowStopButton

for each i in 0..oG3DPLIdx
   Ng=i+1
  'Berechnung der neuen 2D-Querprofilschnittlinien in einem neuen Thema
  ListofListofNDaten={}
  for each j in 0..IdxScht
    ListofNDaten={}
    aFTab=ListofFTabs.Get(j)
```

```
    aListof3DFld=ListofListof3DFlds.Get(j)
    aShape=aFTab.ReturnValue(aListof3DFld.Get(0), i)
    theProfilList={}
    theProfilList.Add({aShape})
    theRWAnf=2590000.00
    theRWEnd=2500000.00
    if (theProfilList <> 0) then
      for each q in theProfilList
        theLines=q.Get(0).AsList
        for each m in theLines
          Listofx={}
          Listofy={}
          Listofz={}
          for each ptx in m
            myx=ptx.Getx
            myy=ptx.Gety
            myz=ptx.Getz
            Listofx.Add(myx)
            Listofy.Add(myy)
            Listofz.Add(myz)
            if (j = 0) then
              If (myx < theRWAnf) then
                theRWAnf=myx
              elseif (myx > theRWEnd) then
                theRWEnd=myx
              end
            end
          end
        end
      end
    end
    Anz=Listofx.Count
    Index=Anz-1
    ListofPoint={}
    for each k in 0..Index
      xkrd=Listofx.Get(k)
      zkrd=(Listofz.Get(k))*Ft
      ListofPoint.Add(xkrd@zkrd)
    end
    ListofListofPoint={}
    ListofListofPoint.Add(ListofPoint)
    thePolyLine=PolyLine.Make(ListofListofPoint)
'   MsgBox.report(thePolyLine.AsString, "the PolyLine (2D)")
    ListofNDaten.Add(thePolyLine)
    ListofNDaten.Add(j)
    ListofNDaten.Add(i)
    theHW= Listofy.Get(0)
    ListofNDaten.Add(theHW)
    ListofNDaten.Add(theRWAnf)
    ListofNDaten.Add(theRWEnd)
    theEbene=ListofKW.Get(j)
    ListofNDaten.Add(theEbene)
    ListofListofNDaten.Add(ListofNDaten)
  end

  'Ein Feature-Shape-File für Profilschnitte (PolyLine)
  'wird hergestellt.
  HWInt=theHW.SetFormat("d")
  fnStr=EB+(HWInt).AsString+".shp"
  fName=FileName.Make(theSV+fnStr)
  NewFTab=FTab.MakeNew(fName, PolyLine)
  ShapeField=NewFTab.FindField("shape")
```

```
IDField=Field.Make("ID", #Field_Short, 2, 0)
IDHWField=Field.Make("IDHW", #Field_Short, 4, 0)
HWField=Field.Make("HW", #Field_Float, 10, 2)
RW1Field=Field.Make("RWAnf", #Field_Float, 10, 2)
RW2Field=Field.Make("RWEnd", #Field_Float, 10, 2)
EbeneField=Field.Make("Flaeche", #Field_Char, 6, 0)
ListofNFlds={IDField , IDHWField, HWField,
          RW1Field, RW2Field, EbeneField}
NewFTab.AddFields(ListofNFlds)
ListofNFlds2={ShapeField,IDField,IDHWField,
           HWField,RW1Field,RW2Field,EbeneField}
IdxNFld=5

NewFTab.SetEditable(false)
NewFTab.SetEditable(true)

AnzDatenS= ListofListofNDaten.Count
IdxDatenS= AnzDatenS-1
RecNr=-1
For each aDS in 0.. IdxDatenS
    recNr=recNr+1
    NewFTab.AddRecord
    ListofNDaten= ListofListofNDaten.Get(aDS)
    AnzNDt= ListofNDaten.Count
    IdxNDt= AnzNDt-1
    For each aDt in 0.. IdxNDt
        theFld= ListofNFlds2.Get(aDt)
        theDt = ListofNDaten.Get(aDt)
        NewFTab.SetValue(theFld, recNr, theDt)
    end
end
NewFTab.SetEditable(false)
thmNew=FTheme.Make(NewFTab)
ProfilView.AddTheme(thmNew)

theLegend=thmNew.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(thmNew, "Flaeche")
'theLegend.SetNullValue("Flaeche", "Null")
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofSColor={}
for each aKl in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(aKl).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   aIdxLb = aListofdfm.FindByValue(theKlasseLb)
   if (aIdxLb <> - 1) then
     aCNr = aListofdfm.Get((aIdxLb + 1))
   elseif (aIdxLb = -1) then
     aR = aKl Mod 60
     if (aR < 2) then
       aCNr = 2
     elseif (aR > 1) then
       aCNr = aR
     end
   end

   theColor=(thePalette.GetList(#PALETTE_LIST_COLOR).Get(aCNr))
   theRgbList=theColor.GetRgbList
   aColor=Color.Make
```

```
      aColor.SetRgbList(theRgbList)
      aListofSColor.Add(aColor)
    end
    aListofSymbol=theLegend.GetSymbols
    AnzSymb=aListofSymbol.Count
    AnzSymbIdx=AnzSymb-2

    for each symb in 0..AnzSymbIdx
      aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
    end
    thmNew.UpdateLegend
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/oG3DPL*100)
    if (not more) then
      exit
    end
  end



'pfq2dpg3.ave
'Aus 3D-Querprofilschnittlinien in mehren Themen der gologischen Flächen
'in einem Karten-View werden 2D-Querprofilschnitte (Polygone) in einem
'aktiven Querprofilschnitt-View hergestellt. Jedes neue 2D-
'Querprofilschnitt-Thema enthält alle geologischen Schichten an einer
'Querprofilschnittlinie im Karten-View.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
ProfilView=av.GetActiveDoc  'ein aktives View für alle Querprofilschnitte,
                    'welche mit diesem Script neu hergestellt werden.
myScript=theProject.FindScript("pfq2dpg3")
myScript.SetNumberFormat( "d.dd") ' script default

ListofKtViews={"Kt1_gg", "Kt1_ggd", "Kt1_hg", "Kt1_hgd", "Kt1_tga", "Kt1_tgd"}
theViewStr=MsgBox.ChoiceAsString(ListofKtViews,
        "das die 3D-Profilschnittlinien-Themen enthält",
        "Eingabe eines Karten-Views")
theKtView=theProject.FindDoc(theViewStr)

ListofFlaechen = {"Geländeoberfläche", "Basisfläche der Deckschichten (TOK)",
            "Quartärbasis"}

ListofFlKW = {"GH", "TOK", "QbF"}

'Anzahl der geologischen Flächen
AnzScht= ListofFlaechen.Count
IdxScht= AnzScht-1

'Feature-Shape-Dateien (PolyLineZ) werden im Karten-View geöffnet.
ListofKtThms=theKtView.GetThemes
ListofPLThms={}
ListofFTabs={}
ListofListof3DFlds={}
ListofSchtFlds = {}

for each aTh in 0.. IdxScht
   NrStr=((aTh+1).SetFormat("").SetFormat("d")).AsString
   theScht=MsgBox.ListAsString(ListofFlaechen,
        "Geologie des"++NrStr++". Datensatzes",
        "Auswahl einer Fläche der Querprofilschnittlinien")
```

```
    PLThm=MsgBox.ListAsString(ListofKtThms,
          "Eingabe eines Themas für 3D-Querprofilschnittlinien"
          ++"im Karten-View"++ theKtView.AsString,
          NrStr++". PolyLineZ-Thema für "++theScht.AsString)
    ListofPLThms.Add(PLThm)
    FTab0=PLThm.GetFTab
    Listof3DFlds=FTab0.GetFields

    Anz3DFlds = Listof3DFlds.Count
    Idx3DFlds = Anz3DFlds - 1

   'Feststellung des ersten Datensatzes
    FldStr=""
    DtStr=""
    for each i in 0..Idx3DFlds
       aFld=Listof3DFlds.Get(i)
       aFldStr=aFld.GetName
       if (aFldStr <> "Shape") then
          FldStr = FldStr + aFldStr+"; "
          aValue=FTab0.ReturnValue(aFld, 0)
          DtStr=DtStr+aValue.AsString+"; "
       end
    end

    MsgBox.Report("Die Namen der Felder"+NL+FldStr+NL+
             "Der erste Datensatz:"+NL+DtStr,
             "Information des Themas"++PLThm.AsString)
   aSchtFld = MsgBox.ListAsString(Listof3DFlds,
      "für Klassifizierung des Legendes",
      "Auswahl eines Feldes des Themas"++PLThm.AsString)

   ListofFTabs.Add(FTab0)
   ListofListof3DFlds.Add(Listof3DFlds)
   ListofSchtFlds.Add(aSchtFld)
end

FTab1=ListofFTabs.Get(0)
'Feststellung der Anzahl der 3D-PolyLine in dem Thema
oG3DPL=0
for each rec in FTab1
   oG3DPL=oG3DPL+1
end
oG3DPLIdx=oG3DPL-1
PLThm1=ListofPLThms.Get(0)
MsgBox.Info(oG3DPL.AsString, "Anzahl der 3D-PolyLine in dem Thema"
       ++PLThm1.AsString)

'Verzeichnis der Profilschnitte auf Festplatte,
'die durch dieses Programm entstehen, als Sourceverzeichnis

fName00=FileDialog.Show("*.shp", "shape file",
          "Anzeige einer Datei auf Festplatte,"++
          "um Verzeichnis festzustellen")
fNameStr = fName00.AsString
if (fNameStr <> Nil) then
   MsgBox.Info(fNameStr, "Der Name einer Datei mit Verzeichnis")
   Frg1=MsgBox.YesNo("Hat der Name auch einen Dateiname außer"
      ++"Verzeichnis?","Kontrolle der Daten", True)
   if (Frg1) then
      ListoffNameStr=fNameStr.AsTokens("\")
      AnzfNameStr=ListoffNameStr.Count
      theIdxfNameStr=AnzfNameStr-1
```

```
    thefNameStr=ListoffNameStr.Get(theIdxfNameStr)
    MsgBox.Info(thefNameStr, "Der Name einer Datei")
    fNameStr.Substitute (thefNameStr, "")
  end
else
  fNameDefault="C:\Verz1\Verz2\Verz3\"
  fNameStr=MsgBox.Input("Verzeichnis der Querprofilschnitte auf"
       ++"Festplatte", "Eingabe der Daten", fNameDefault)
end
MsgBox.Info(fNameStr,"Verzeichnis der Querprofilschnitte"
       ++"auf Festplatte")

Frg2=MsgBox.YesNo("Ist der Name des Verzeichnises richtig?",
            "Kontrolle der Daten", True)
if (Frg2) then
  theSV=fNameStr
else
  theSV=MsgBox.Input("die durch dieses Programm entstehen",
      "Verzeichnis der Profilschnitte auf Festplatte,",
      fNameStr)
end
MsgBox.Info(theSV,"Verzeichnis der Querprofilschnitte auf Festplatte")

ListofEB={"Z", "F", "V", "X", "S", "K", "P", "G", "B", "C", "U"}
EB=MsgBox.ChoiceAsString(ListofEB,
      "der neuen Querprofilschnitte (Polygon)",
      "Auswahl des ersten Buchstabens")

FtStr=MsgBox.Input("zur Überhöhung des Profilschnittes",
    "Eingabe des Faktors","50")
Ft=FtStr.AsNumber

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette = av.GetSymbolWin.GetPalette

aListofdfm = {"---",20, "a",43,  "a/Mj",24,  "Aussen",5,
    "d",36, "f",8, "Gy",24,  "H",32, "hg/plRR",3,
    "Hj",32, "Hn",11, "Lf",17, "Lfh/N",14, "Lö",47,
    "Lö/Hj",30, "Lö/Mj",24, "Löy",6,  "Ma",29, "mi-olK",35,
    "miIV",11, "Mj",26, "N",16, "plRR",53, "Rhein",20,
    "Sf",18, "Sfh/N",16,  "sSo",24, "tAb",53, "tTt",51,
    "Deck",53,  "D",53, "De",53, "NT",14, "MT",26,
    "HT",32, "Präq",3, "Präm",3,  "GH",53, "NA",5, "NQ",8,
    "TOK",14, "NTabF",5, "MTabF",8, "QB",8, "QmitD",20,
    "QohneD",26, "QohneT",8, "TrorAe",3,  "UMT",26,
    "UMT III",26, "UMT IV",26, "OMT",41, "rMTl",20,
    "lMTl",26, "rHTl",32, "lNT",14, "rNT",17, "lHTr",32,
    "rHTr",32, "HTr",32, "MTl",26, "MTr",20, "HTl",32,
    "Deck-Mu",53, "Deck-LS",47, "UMT2",26, "Holst-Sd",38,
    "Holst-Tf",35, "UMT1",41, "Holst-Ton",23, "Holst-U",44,
    "SdScht-HR",45, "MMT-R",20, "Geländeoberfläche",53,
    "Basisfläche der Deckschichten (TOK)",14, "NT abgedeckte Fläche",5,
    "MT abgedeckte Fläche",8, "Quartärbasis",8,
    "Oberfläche der als MT ältere Schichten",8,
    "Oberfläche der Präquartär-Schichten",8,
    "Deckschichten",53, "Niederterrassen",14, "Mittelterrassen",26,
    "Präquartäre Schichten",3,
    "Deckschichten (Mutter oder Waldboden)",53,
    "Deckschichten (Lehm oder Sand)",47, "Untere Mittelterrasse 2",26,
    "Untere Mittelterrasse 1",41, "Holstein (Torf, z.T. Tonhaltig)",35,
    "Holstein (Sandschichten mit Tonlagen)",38,
    "Holstein (Tonschichten)",23, "Holstein (Schluffschichten)",44,
```

```
    "Sandschichten (z.T. Holstein, z.T. Rinnenschotter)",45,
    "Die mittlere Mittelterrasse (Rinnenschotter)",20,
    "Untere Mittelterrasse",26, "Obere Mittelterrasse",41,
    "Hauptterrasse",32, "als MT ältere Schichten",3,
    "Präquartär-Schichten",3, "Untere Mittelterrasse III",26,
    "Untere Mittelterrasse IV",26, "Sonst",0}

av.ShowMsg("Berechnung der 2D-Profilschnitte in View"
      ++ProfilView.AsString)
av.ShowStopButton

ListofPgBz = {"Deck", "Terrassen", "Quartaer"}

ListofPgs = {}
ListofHW = {}
ListofRWAnf = {}
ListofRWEnd = {}
ListofScht = {}
ListofBz = {}


for each i in 0..oG3DPLIdx
  Ng=i+1
  ListofListofPoint={}
  ListofTScht = {}

  for each j in 0..IdxScht
    aFTab = ListofFTabs.Get(j)
    aListofFlds = ListofListof3DFlds.Get(j)
    aShpFld = aListofFlds.Get(0)
    aShape=aFTab.ReturnValue(aShpFld, i)
    aSchtFld = ListofSchtFlds.Get(j)
    aScht = aFTab.ReturnValue(aSchtFld, i)
    ListofTScht.Add(aScht)
    theProfilList={}
    theProfilList.Add({aShape})
    Listofx={}
    Listofy={}
    Listofz={}
    if (theProfilList <> 0) then
      for each q in theProfilList
        theLines=q.Get(0).AsList
        for each m in theLines
          for each ptx in m
            myx=ptx.Getx
            myy=ptx.Gety
            myz=ptx.Getz
            Listofx.Add(myx)
            Listofy.Add(myy)
            Listofz.Add(myz)
          end
        end
      end
    end
    Anz=Listofx.Count
    Index=Anz-1
    ListofPoint={}
    for each k in 0..Index
      xkrd=Listofx.Get(k)
      zkrd=(Listofz.Get(k))*Ft
      ListofPoint.Add(xkrd@zkrd)
    end
```

```
  theRWAnf = Listofx.Get(0)
  theRWEnd = Listofx.Get(Index)
  ListofListofPoint.Add(ListofPoint)
  theHW = Listofy.Get(0)
end

ListofQuar = {}
ListofDeck={}
ListofDeckUmk={}

ListofTOK={}
ListofTOKUmk={}

ListofPQ={}

AnzDatenS= ListofListofPoint.Count
IdxDatenS= AnzDatenS-1

for each aPgR in 0.. IdxDatenS
   ListofPoint = ListofListofPoint.Get(aPgR)
   AnzNPt= ListofPoint.Count
   IdxNPt= AnzNPt-1
   for each aPgT in 0.. IdxNPt
      aPt= ListofPoint.Get(aPgT)
      aIdxUmk= IdxNPt- aPgT
      aUmkPt= ListofPoint.Get(aIdxUmk)
      if (aPgR = 0) then
         ListofQuar.Add(aPt)
         ListofDeck.Add(aPt)
      elseif (aPgR = 1) then
         ListofDeckUmk.Add(aUmkPt)
         ListofTOK.Add(aPt)
      elseif (aPgR = 2) then
         ListofTOKUmk.Add(aUmkPt)
         ListofPQ.Add(aPt)
      end
   end
end

AnzDeckUm= ListofDeckUmk.Count
IdxDeckUm= AnzDeckUm-1

for each aPt in 0.. IdxDeckUm
    aUmPt= ListofDeckUmk.Get(aPt)
    ListofDeck.Add(aUmPt)
end

AnzNTUm= ListofTOKUmk.Count
IdxNTUm= AnzNTUm-1

for each aPt in 0.. IdxNTUm
    aUmPt= ListofTOKUmk.Get(aPt)
    ListofTOK.Add(aUmPt)
end

AnzNTUm= ListofTOKUmk.Count
IdxNTUm= AnzNTUm-1

for each aPt in 0.. IdxNTUm
    aUmPt= ListofTOKUmk.Get(aPt)
    ListofQuar.Add(aUmPt)
end
```

```
  ListofListofPgPt={}
  ListofListofPgPt.Add(ListofQuar)
  PgPQ=Polygon.Make(ListofListofPgPt)
  ListofPgs.Add(PgPQ)
  ListofHW.Add(theHW)
  ListofRWAnf.Add(theRWAnf)
  ListofRWEnd.Add(theRWEnd)
  aTScht = ListofTScht.Get(2)
  ListofScht.Add(aTScht)
  ListofBz.Add(ListofPgBz.Get(2))

  ListofListofPgPt={}
  ListofListofPgPt.Add(ListofDeck)
  PgDeck=Polygon.Make(ListofListofPgPt)
  ListofPgs.Add(PgDeck)
  ListofHW.Add(theHW)
  ListofRWAnf.Add(theRWAnf)
  ListofRWEnd.Add(theRWEnd)
  aTScht = ListofTScht.Get(0)
  ListofScht.Add(aTScht)
  ListofBz.Add(ListofPgBz.Get(0))

  ListofListofPgPt={}
  ListofListofPgPt.Add(ListofTOK)
  PgNT=Polygon.Make(ListofListofPgPt)
  ListofPgs.Add(PgNT)
  ListofHW.Add(theHW)
  ListofRWAnf.Add(theRWAnf)
  ListofRWEnd.Add(theRWEnd)
  aTScht = ListofTScht.Get(1)
  ListofScht.Add(aTScht)
  ListofBz.Add(ListofPgBz.Get(1))


  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/oG3DPL*100)
  if (not more) then
    exit
  end
end

'Herstellung der neuen Profilschnitt-Themen für 2D-Polygone

'Anzahl der ganzen Datensätze
AnzgDS = ListofPgs.Count
IdxgDS = AnzgDS - 1

ListofHWSt = {}

for each i in 0..IdxgDS
  aHW = ListofHW.Get(i)
  if (i = 0) then
    HW0 = aHW
    ListofHWSt.Add(HW0)
  elseif (i > 0) then
    if (aHW <> HW0) then
      HW0 = aHW
      ListofHWSt.Add(HW0)
    end
  end
end
```

```
'Anzahl der neuen Themen nach HW
AnzNTh = ListofHWSt.Count
IdxNTh = AnzNTh - 1
'MsgBox.Info(AnzNTh.AsString, "Anzahl der neuen Themen")

'FR11 =MsgBox.YesNo("Sind die Anzahl der neuen Themen richtig?",
'     "Kontrolle", true)

'if (Not FR11) then
'   MsgBox.Error("Die Zahl stimmt nicht!"+NL+
'                "Das Programm wird abgebrochen!", "")
'   exit
'end

av.ShowMsg("Speicherung der 2D-Profilschnitte in View"
       ++ProfilView.AsString)
av.ShowStopButton


AnfIdx = 0
Ng = 0
for each aTh in 0..IdxNTh
    Ng = Ng + 1
    theHW = ListofHWSt.Get(aTh)

  'Ein Feature-Shape-File für Profilschnitte (Polygon) wird hergestellt.
  HWInt=theHW.SetFormat("d")
  fnStr=EB+(HWInt).AsString+".shp"
  fName=FileName.Make(theSV+fnStr)
  NewFTab=FTab.MakeNew(fName, Polygon)
  ShapeField=NewFTab.FindField("shape")
  IDField=Field.Make("ID", #Field_Short, 4, 0)
  IDHWField=Field.Make("IDHW", #Field_Short, 4, 0)
  HWField=Field.Make("HW", #Field_Float, 10, 2)
  RW1Field=Field.Make("RWAnf", #Field_Float, 10, 2)
  RW2Field=Field.Make("RWEnd", #Field_Float, 10, 2)
  StField=Field.Make("Schichten", #Field_Char, 20, 0)
  BzField=Field.Make("Bezeichnung", #Field_Char, 10, 0)
  ListofNFlds={IDField , IDHWField, HWField,
               RW1Field, RW2Field, StField, BzField}
  NewFTab.AddFields(ListofNFlds)

  NewFTab.SetEditable(false)
  NewFTab.SetEditable(true)

  recNr = -1
  Nr = -1

  while (Nr <> -2)
    Nr = Nr + 1
    aIdx = AnfIdx + Nr
    if (aIdx <= IdxgDS) then
      aHW = ListofHW.Get(aIdx)
      if (aHW = theHW) then
        recNr = recNr + 1
        NewFTab.AddRecord
        aPg = ListofPgs.Get(aIdx)
        aRWAnf = ListofRWAnf.Get(aIdx)
        aRWEnd = ListofRWEnd.Get(aIdx)
        aScht = ListofScht.Get(aIdx)
        aBz = ListofBz.Get(aIdx)
```

```
      NewFTab.SetValue(ShapeField, recNr, aPg)
      NewFTab.SetValue(IDField, recNr, recNr)
      NewFTab.SetValue(IDHWField, recNr, aTh)
      NewFTab.SetValue(HWField, recNr, aHW)
      NewFTab.SetValue(RW1Field, recNr, aRWAnf)
      NewFTab.SetValue(RW2Field, recNr, aRWEnd)
      NewFTab.SetValue(StField, recNr, aScht)
      NewFTab.SetValue(BzField, recNr, aBz)
    elseif (aHW <> theHW) then
      AnfIdx = aIdx
      Nr = -2
      break
    end
  elseif (aIdx > IdxgDS) then
      Nr = -2
      break
  end
end


NewFTab.SetEditable(false)
thmNew=FTheme.Make(NewFTab)
ProfilView.AddTheme(thmNew)

theLegend=thmNew.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
SchtFldStr = StField.AsString
theLegend.Unique(thmNew, SchtFldStr)
'theLegend.SetNullValue(SchtFldStr, "Null")
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-1
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofSColor={}
for each aKl in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(aKl).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   aIdxLb = aListofdfm.FindByValue(theKlasseLb)
   if (aIdxLb <> - 1) then
     aCNr = aListofdfm.Get((aIdxLb + 1))
   elseif (aIdxLb = -1) then
     aR = aKl Mod 60
     if (aR < 2) then
       aCNr = 2
     elseif (aR > 1) then
       aCNr = aR
     end
   end

   theColor=(thePalette.GetList(#PALETTE_LIST_COLOR).Get(aCNr))
   theRgbList=theColor.GetRgbList
   aColor=Color.Make
   aColor.SetRgbList(theRgbList)
   aListofSColor.Add(aColor)
end
aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
AnzSymbIdx=AnzSymb-1

for each symb in 0..AnzSymbIdx
   aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
```

```
  end
  thmNew.UpdateLegend
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzNTh*100)
  if (not more) then
    exit
  end

end



'pfqaddth.ave
'Die Themen für Querprofilschnitte werden von einem anderen View
'in einem aktiven View geladen (Add-Funktion).
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
myScript=theProject.FindScript("pfqaddth")
myScript.SetNumberFormat( "d") ' script default

ListofViews={"Qprfg_gg", "Qprf0_gg", "Qprf1_gg", "Qprf2_gg", "Qprf3_gg",
            "Qprf4_gg", "Qprf5_gg", "Qprf6_gg", "Qprf7_gg", "Qprf8_gg",
            "Qprf9_gg", "Qprfg_gd", "Qprf_ggd", "Qprf_gg"}
GP=MsgBox.ChoiceAsString(ListofViews, "Das View mit den umzuladenden Profilen",
                    "Auswahl eines Views")
theView=theProject.FindDoc(GP)
Listof2B={"B5","C5","F5","G5","H5","K5","L5","M5","N5","O5",
         "P5","Q5","S5","T5","U5","V5", "W5","X5","Z5"}
B2=MsgBox.ChoiceAsString(Listof2B, "Die ersten zwei Buchstaben der Profile",
              "Auswahl der Querprofilschnitte zu laden")
AWPrf=B2.Left(1)
theNView=av.GetActiveDoc   'Das aktive View
Frg1=MsgBox.YesNo("Gibt es im aktiven View schon Profilschnitte"+NL+
    "wo jetzt die Profilschnitte geladen werden ?",
    "Vorhandene Querprofilschnitte im aktiven View ?", True)
if (Frg1) then
  V2B=MsgBox.ChoiceAsString(Listof2B,
      "Die ersten zwei Buchstaben der vorhandenen Profilschnitte",
      "Auswahl der Profilschnitte im aktiven View")
  'Bestimmung des minimalen und maximalen Hochwertes im View
  ListofNThemes=theNView.GetThemes
  AnzNThms=ListofNThemes.Count
  ThmsNIdx=AnzNThms-1
  minHW=5642000
  maxHW=5618000
  for each eThm in 0..ThmsNIdx
    theagTh=ListofNThemes.Get(eThm)
    agThStr=theagTh.AsString
    erst2B=agThStr.Left(2)
    if (erst2B = V2B) then
      ListofagThStr=agThStr.AsTokens("BCFGHKLMNPSTUVWXZ.shp")
      aHWBP=ListofagThStr.Get(0)
      aHWBPNr=aHWBP.AsNumber
      if (aHWBPNr < minHW) then
        minHW=aHWBPNr
      end
      if (aHWBPNr > maxHW) then
        maxHW=aHWBPNr
      end
```

```
      end
   end
   minHWStr=minHW.AsString
   maxHWStr=maxHW.AsString
else
   minHWStr=(5618450.00).AsString
   maxHWStr=(5641050.00).AsString
end
AnfHWStr=MsgBox.Input("um die Profilschnitte ins View"
                  ++theNView.AsString++"zu laden",
                   "Eingabe des kleinsten Hochwertes der Profilschnitte",
                    minHWStr)
EndHWStr=MsgBox.Input("um die Profilschnitte ins View"
                  ++theNView.AsString++"zu laden",
                   "Eingabe des größten Hochwertes der Profilschnitte",
                    maxHWStr)
AnfHW=AnfHWStr.AsNumber
EndHW=EndHWStr.AsNumber
AnzThms=(EndHW-AnfHW)/50+1
AnzThmsIdx=AnzThms-1
MsgBox.Info(AnzThms.AsString, "Die Anzahl der umzuladenden Profilschnitte")
av.ShowMsg("Add themes in View"++theNView.AsString++" ...")
av.ShowStopButton
Ng=0
PrfAbstStr=MsgBox.Input("Abstand der Profilschnitte [m]",
       "Auswahl der Profilschnitte zu laden", "50")
PrfAbstNr=PrfAbstStr.AsNumber
for each aNr in 0..AnzThmsIdx
     Ng=Ng+1
     ThNr=(AnfHw+(aNr*PrfAbstNr))
     ThStr=ThNr.AsString
     NThStr=AWPrf+ThStr+".shp"
     PrfTheme=theView.FindTheme(NThStr)
     'aSrcNm = PrfTheme.GetSrcName
     'NPrfTheme = Theme.Make(aSrcNm)
     NPrfTheme = PrfTheme.Clone
     theNView.AddTheme(NPrfTheme)
     'Show percentage complete with enabled stop button
     more=av.SetStatus(Ng/AnzThms*100)
    if (not more) then
      break
    end
end
av.GetProject.SetModified( TRUE )
```

```
'pfqdelet.ave
'Die ausgewälten Profilschnitte werden aus dem aktiven View entladen.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc   'ein aktives View

Listof2B = {"B5","C5","F5","G5","H5","K5","L5","M5",
    "N5","P5","Q5","S5","T5","U5","V5","W5","X5"}
V2B=MsgBox.ChoiceAsString(Listof2B,
   "Die ersten zwei Buchstaben der Profilschnitte",
   "Auswahl des Profilschnittes zu entladen")
```

```
ListofPrf={}
ListofThemes=theView.GetThemes   'Alle Themen im aktiven View
Anz=ListofThemes.Count
AnzIdx=Anz-1
for each aTh in 0..AnzIdx
   athm=ListofThemes.Get(aTh)
   aThmStr=athm.AsString
   B2=aThmStr.Left(2)
   if (B2 = V2B) then
     ListofPrf.Add(athm)
   end
end

AnzPrf=ListofPrf.Count
IdxPrf=AnzPrf-1
Zaeler=IdxPrf
MsgBox.Info(AnzPrf.AsString, "Anzahl der ausgewählten Profilschnitte in TOC")

av.ShowMsg("Delete themes in a TOC ...")
av.ShowStopButton
Ng=0
While (Zaeler > -1)
   Ng=Ng+1
   theTheme=ListofPrf.Get(Zaeler)
   theView.DeleteTheme(theTheme)
   'Show percentage complete with enabled stop button
   more=av.SetStatus(Ng/AnzPrf*100)
   if (not more) then
     break
   end
   Zaeler=Zaeler-1
end
theProject.SetModified(true)



'pfqkothw.ave
'Ein Punkt in einer Tabelle in einem Karten-View, dessen Hochwert mit
'dem HW eines Querprofilschnittes gleich ist, oder der sich in der
'nächesten Entfernung von dem HW des Querprofilschnittes befindet,
'wird herausgesucht. Der gefundene Punkt wird auf dem Bildschirm
'gezeigt und auf eine Querprofilschnitt eingesetzt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc     'ein aktives Querprofilschnitt-View

myScript=theProject.FindScript("pfqkothw")
myScript.SetNumberFormat( "d.dd") ' script default

theTheme=theView.GetActiveThemes.Get(0)  'ein aktives Querprofilschnitt-Thema


Frg11=MsgBox.YesNo("Ist das aktive Thema"++theTheme.AsString
    ++"richtig?", "Kontrolle des aktiven Themas", true)

if (Not Frg11) then

  ListofThms=theView.GetThemes
  ListofPLThms = {}
```

```
  for each aT in ListofThms
     if (aT.Is(FTheme)) then
       aFTab = aT.GetFTab
       if (aFTab.GetShapeClass.IssubclassOf(Polyline)) then
         ListofPLThms.Add(aT)
       end
     end
  end

  theTheme = MsgBox.ChoiceAsString(ListofPLThms,
      "im View"++theView.AsString++"zur Korrektur",
      "Auswahl eines Profilschnitt-Themas")
end

ThStr = theTheme.AsString
B1 = ThStr.Left(1)
BStr = B1 + "."
ListofStr1 = ThStr.AsTokens(BStr)

aPfStr1=ListofStr1.Get(0)
aPfHW=aPfStr1.AsNumber.SetFormat("").SetFormat("d")
MsgBox.Info(aPfHW.AsString, "Der Hochwert des Themas"
     ++theTheme.AsString)

AW2=MsgBox.YesNo("Ist der Hochwert des Themas"
   ++theTheme.AsString++"richtig?", "Kontrolle", TRUE)
if (Not AW2) then
  MsgBox.Error("Der Hochwert ist falsch!", "")
  exit
end
YFtStr=MsgBox.Input("zur Überhöhung der Höhe",
     "Eingabe eines Faktors", "50")
YFt=YFtStr.AsNumber

ListofViews={"Kt1_gg", "Kt1_ggd", "Kt1_hg",
          "Kt1_hgd", "Kt1_tga", "Kt1_tgd"}
View2Str=MsgBox.ChoiceAsString(ListofViews,
 "in dem das Punkt-Thema für Grenze liegt","Auswahl eines Karten-Views")
View2=theProject.FindDoc(View2Str)

ListofThms2 = View2.GetThemes

ListofPtThms = {}

for each aT in ListofThms2
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
       ListofPtThms.Add(aT)
     end
   end
end

PtTheme = MsgBox.ChoiceAsString(ListofPtThms,
     "in dem die Punkte für Grenze liegen",
     "Auswahl eines Themas im Karten-View")

PtFTab = PtTheme.GetFTab
ListofFelds = PtFTab.GetFields
PtShpFld = ListofFelds.Get(0)
PtIDFld = PtFTab.FindField("ID")
PtH50Fld = MsgBox.ChoiceAsString(ListofFelds,
```

```
        "für 50-fach erhöhte Höhen",
        "Auswahl eines Feldes der Punkte")

AnzPt=0
for each Pt in PtFTab
   AnzPt=AnzPt+1
end
AnzPtIdx=AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"
      ++PtTheme.AsString)

FTab1=theTheme.GetFTab
ListofFlds1 = FTab1.GetFields
ShpFld1=FTab1.FindField("Shape")

'Alle Punkte in der Tabelle, deren Hochwerte gleich wie der HW
'des Querprofilschnittes im Querprofilschnitt-View sind, werden gesucht.
ListofPtIdx={}
minAbst = 1000000
minIdx = -1
for each aPt in 0..AnzPtIdx
   thePt = PtFTab.ReturnValue(PtShpFld, aPt)
   aPty = thePt.Gety
   Abst=(aPty - aPfHW).Abs
   if (Abst = 0) then
     ListofPtIdx.Add(aPt)
   elseif (Abst <> 0) then
     if (Abst < minAbst) then
       minAbst = Abst
       minIdx = aPt
     end
   end
end

AnzPfPt=ListofPtIdx.Count
AnzPfPtIdx=AnzPfPt-1

if (AnzPfPt = 0) then
  ListofPtIdx.Add(minIdx)
  AnzPfPt = 1
  AnzPfPtIdx = 0
end

ListofAW3DPt = {}
for each Pt in 0..AnzPfPtIdx
   PtIdx=ListofPtIdx.Get(Pt)

   thePt=PtFTab.ReturnValue(PtShpFld, PtIdx)
   thePtRW=thePt.Getx
   thePtHW=thePt.Gety
   thePtH50=PtFTab.ReturnValue(PtH50Fld, PtIdx)
   thePtHm=thePtH50/ YFt
   PtIdxStr=(PtIdx.SetFormat("").SetFormat("d")).AsString
   MsgBox.Report("Der Rechtswert:    "++thePtRW.AsString+NL+
              "Der Hochwert:      "++thePtHW.AsString+NL+
              "Die 50-fache Höhe: "++thePtH50.AsString
               ++"("+thePtHm.AsString++"[m ü NN])",
              "Der"++(PtIdxStr)++"."++"Punkt des Themas"
               ++PtTheme.AsString)

   ListofAW3DPt.Add(thePtRW@thePtHW@thePtH50)
   thePt=Point.Make(thePtRW, thePtH50)
```

```
    theGraphicPt=GraphicShape.Make(thePt)
    theSymbol=theGraphicPt.GetSymbol
    theSymbol=BasicMarker.Make
    theSymbol.SetStyle(#BASICMARKER_STYLE_PATTERN)
    theSymbol.SetCharacter("+".AsASCII)
    theSymbol.SetSize(20)
    theSymbol.SetColor(Color.GetRed)
    theGraphicPt.SetSymbol(theSymbol)
    theGraphicList=theView.GetGraphics
    theGraphicList.Add(theGraphicPt)
end

'Auswahl eines Punktes
if (AnzPfPt = 1) then
  theAW3DPt = ListofAW3DPt.Get(0)
elseif (AnzPfPt > 1) then
  ListofaDStr = {}
  for each i in 0..AnzPfPtIdx
    aAW3DPt = ListofAW3DPt.Get(i)
    ax = aAW3DPt.Getx.AsString
    ay = aAW3DPt.Gety.AsString
    az = aAW3DPt.Getz
    aHm = ((az / YFt).SetFormat("").SetFormat("d")).AsString
    aDStr = ax +";  "++ay+";  " ++ aHm
    ListofaDStr.Add(aDStr)
  end
  aAWDStr = MsgBox.ListAsString(ListofaDStr,
        "um den in Profilschnitt einzusetzen"
         +NL+"(Felder: x;  y;  z [m])", "Auswahl eines Punktes")
  aAWIdx = ListofaDStr.FindByValue(aAWDStr)
  theAW3DPt = ListofAW3DPt.Get(aAWIdx)
end

'Feststellung der Anzahl der 2D-PolyLine in dem Thema
Anz2DPL=0
for each rec in FTab1
   Anz2DPL=Anz2DPL+1
end
Anz2DPLIdx=Anz2DPL-1

'Feststellung der Anzahl der Felder
AnzFlds1 = ListofFlds1.Count
IdxFlds1 = AnzFlds1 - 1

aSchtFld = MsgBox.ListAsString(ListofFlds1,
  "für Klassifizierung der Legende",
  "Auswahl eines Feldes des Themas"++theTheme.AsString)
aSchFldStr = aSchtFld.AsString

'Feststellung der Datensätze
ListofDt={}

for each j in 0..Anz2DPLIdx
  aValue = FTab1.ReturnValue(aSchtFld, j)
  ListofDt.Add(aValue)
end

aDtS = MsgBox.ListAsString(ListofDt,
      "um einen Abschnitt zu korrigieren",
      "Auswahl eines Datensatzes im Profilschnitt"++ theTheme.AsString)
aDtSIdx = ListofDt.FindByValue(aDtS)
```

```
PLAusg = FTab1.ReturnValue(ShpFld1, aDtSIdx)
MsgBox.Report(PLAusg.AsString, "Die ausgewählte PolyLine")

recNr=Anz2DPLIdx
ListofFLPt = {}

av.ShowMsg("Feststellung der Koordinaten der 2D-Querprofilschnitte in Thema"
    ++theTheme.AsString++"...")

theProfilList1={}
theProfilList1.Add({PLAusg})
'2D-PolyLine wird in Liste der Koordinaten umgewandelt.
if (theProfilList1 <> 0) then
  for each q in theProfilList1
    theLines=q.Get(0).AsList
    for each m in theLines
      for each apt in m
        ListofFLPt.Add(apt)
      end
    end
  end
end

PtAnz = ListofFLPt.Count
PtAnzIdx = PtAnz - 1

'Herausfinden eines Punktes auf dem Querprofilschnitt

minDist=1000
minIdx=-1

aMPtx = theAW3DPt.Getx
aMPty = theAW3DPt.Getz

for each i in 0..PtAnzIdx
  aPt = ListofFLPt.Get(i)
  xkrd= aPt.Getx
  ykrd= aPt.Gety
  xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
  yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
  xyAbst = ((xAbst  + yAbst).sqrt) .Abs
  if (xyAbst < minDist) then
    minDist = xyAbst
    minIdx = i  'Index des nähesten Punktes des Maus-Klickens
  end
end

'Bestimmung der Stelle auf der geologischen Fläche
'als x-, y-Koordinaten

if (minIdx = 0) then
   vIdx = 0     'ein Vertex auf der Linie vor dem Punkt im Punkt_Thema
   nIdx = 1     'ein Vertex auf der Linie nach dem Punkt im Punkt_Thema
elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
   aPfPtv = ListofFLPt.Get((minIdx -1))
   aPfPt = ListofFLPt.Get(minIdx)
   aPfPtn = ListofFLPt.Get((minIdx + 1))

   xkrdv = aPfPtv.Getx  'x-Koord. vom letzten Vertex
   xkrd  = aPfPt.Getx   'x-Koord. vom nächesten Vertex
   xkrdn = aPfPtn.Getx  'x-Koord. vom nächsten Vertex
```

```
ykrdv = aPfPtv.Gety  'y-Koord. vom letzten Vertex
ykrd = aPfPt.Gety        'y-Koord. vom nächesten Vertex
ykrdn = aPfPtn.Gety  'y-Koord. vom nächsten Vertex

xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten Vertex
                        'vom nächesten Vertex
xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten Vertex
                        'vom nähesten Vertex
xML = (xkrd - aMPtx) * (xkrd - aMPtx)
yML = (ykrd - aMPty) * (ykrd - aMPty)
xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Punkt
                        'vom nächesten Vertex
xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Punkt
                         'vom letzten Vertex
xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Punkt
                         'vom nächsten Vertex
vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem
vLpML = xyAv + xyML        'Punkt und dem letzten Vertex

nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen
nLpML = xyAn + xyML        'dem Punkt und dem nächsten Vertex

vM = (vLmML - xyMLv).Abs   'Differenz zwischen der theorischen und der
vP = (vLpML - xyMLv).Abs   'tatsächlichen Länge im Bezug
                    'auf den letzten Vertex
nM = (nLmML - xyMLn).Abs   'Differenz zwischen der theorischen und der
nP = (nLpML - xyMLn).Abs   'tatsächlichen Länge im Bezug
                    'auf den nächsten Vertex
ListofLg = {vM, vP, nM, nP}   'Suche nach der kleinsten Länge
MinLg = 10000
TheLgIdx = -1
For each aLg in 0..3
     theLg = ListofLg.Get(aLg)
     if (theLg < MinLg) then
          MinLg = theLg
          TheLgIdx = aLg
     end
end
if (xyML < 10) then
   vIdx = minIdx - 1
   nIdx = minIdx + 1
end
if (xyML >= 10) then
  if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
    vIdx = minIdx - 1  'ein Vertex auf der Linie vor dem Punkt
    nIdx= minIdx       'ein Vertex auf der Linie nach dem Punkt
  elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
    vIdx = minIdx      'ein Vertex auf der Linie vor dem Punkt
    nIdx = minIdx + 1  'ein Vertex auf der Linie nach dem Punkt
  else
    vIdx = minIdx      'ein Vertex auf der Linie vor dem Punkt
    nIdx = minIdx + 1  'ein Vertex auf der Linie nach dem Punkt
  end
end
```

```
elseif (minIdx = PtAnzIdx)   then
    vIdx = minIdx - 1
    nIdx = minIdx
end

av.ShowMsg("Speicherung der neuen 2D-Profilschnitte in das Thema"
       ++theTheme.AsString++"...")

'Einsetzen des Punktes auf die Polyline oder das Polygon
ListofPoint = {}

for each i in 0..PtAnzIdx
  if (i <= vIdx) then
    aPt = ListofFLPt.Get(i)
    ax = aPt.Getx
    ay = aPt.Gety
    ListofPoint.Add(ax@aY)
  end
end
aNx = theAW3DPt.Getx
aNz = theAW3DPt.Getz
aNPt = Point.Make(aNx, aNz)
ListofPoint.Add(aNPt)
for each i in 0..PtAnzIdx
  if (i >= nIdx) then
    aPt = ListofFLPt.Get(i)
    ax = aPt.Getx
    ay = aPt.Gety
    ListofPoint.Add(ax@aY)
  end
end
ListofListofPoint={}
ListofListofPoint.Add(ListofPoint)
theNShp=PolyLine.Make(ListofListofPoint)

FTab1.SetEditable(false)
FTab1.SetEditable(true)

FTab1.SetValue(ShpFld1, aDtSIdx, theNShp)

FTab1.SetEditable(false)

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

aListofFarbD = {"GH", 53, "NT", 20, "Na", 5, "NQ", 8}

theLegend=theTheme.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(theTheme, aSchFldStr)
'theLegend.SetNullValue(aSchFldStr, "Null")

ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofSColor={}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   aIdxLb = aListofFarbD.FindByValue(theKlasseLb)
   if (aIdxLb <> -1) then
```

```
      aCNr = aListofFarbD.Get((aIdxLb + 1))
    elseif (aIdxLb = -1) then
      aR = i Mod 60
      if (aR < 3) then
        aCNr = 2
      elseif (aR > 2) then
        aCNr = aR
      end
    end
    theColor=(thePalette.GetList(#PALETTE_LIST_COLOR).Get(aCNr))
    theRgbList=theColor.GetRgbList
    aColor=Color.Make
    aColor.SetRgbList(theRgbList)
    aListofSColor.Add(aColor)
end

aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
AnzSymbIdx=AnzSymb-2

for each symb in 0..AnzSymbIdx
    aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
end
theTheme.UpdateLegend
```

```
'pfql2geo.ave
'Querprofilschnittlinien in einem Thema werden durch eine digitale
'geologische Karte in Teillinien mit den geologischen Bezeichnungen unterteilt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc         'Aktives Karten-View
'myScript=theProject.FindScript("pfql2geo")
'myScript.SetNumberFormat("d.dd")
ListofThemes=theView.GetThemes
ListofPLFThm = {}
for each aT in ListofThemes
    if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(PolyLine)) then
        ListofPLFThm.Add(aT)
      end
    end
end
PLTheme=MsgBox.ChoiceAsString(ListofPLFThm,
 "das die 2D-PolyLinien enthält,"+NL+"um geologischen Linien herzustellen",
       "Eingabe eines Themas im View"++theView.AsString)
PLFTab=PLTheme.GetFTab
PLShpFld=PLFTab.FindField("Shape")
PLIDFld=PLFTab.FindField("IDHW")
PLHWFld=PLFTab.FindField("HW")
PLRWAnfFld=PLFTab.FindField("RWAnf")
PLRWEndFld=PLFTab.FindField("RWEnd")
PLRWAnf=PLFTab.ReturnValue(PLRWAnfFld, 0).SetFormat("").SetFormat("d.dd")
PLRWEnd=PLFTab.ReturnValue(PLRWEndFld, 0).SetFormat("").SetFormat("d.dd")
'Feststellung der Anzahl der 2D-PolyLine in dem Thema
Anz2DPL=0
```

```
for each rec in PLFTab
   Anz2DPL=Anz2DPL+1
end
Anz2DPLIdx=Anz2DPL-1
MsgBox.Info(Anz2DPL.AsString, "Anzahl der 2D-PolyLine im Thema"++PLTheme.AsString)

'Eingabe des Themas von der digitalen geologischen Karte
GeoTheme=MsgBox.ChoiceAsString(ListofThemes, "das die Polygone für Geologie enthält",
      "Eingabe eines Themas im View"++theView.AsString) 'z.B. Clip1.shp
GeoFTab=GeoTheme.GetFTab
GeoShpFld=GeoFTab.FindField("Shape")
ListofFlds=GeoFTab.GetFields
GeoFld=MsgBox.ChoiceAsString(ListofFlds, "Das Feld für Geolgie",
     "Auswahl des Feldes im Thema"++GeoTheme.AsString)
'Feststellung der Anzahl der Shapes in dem geologischen Schichten-Polygon-Thema
AnzGR=0
for each rec in GeoFTab
   AnzGR=AnzGR+1
end
AnzGRIdx=AnzGR-1
MsgBox.Info(AnzGR.AsString, "Anzahl der Shapes in der geologischen
Karte"++GeoTheme.AsString)
'Ein Feature-Shape-File für Geolgie (PolyLine) wird hergestellt.
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("pl2dmgeo","shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("pl2dmgeo","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PolyLine)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PLgFTab=FTab.MakeNew(fName, PolyLine)
ShapeField1=PLgFTab.FindField("shape")
IDHWField1=Field.Make("IDHW", #Field_Short, 4, 0)
HWField1=Field.Make("HW", #Field_Float, 10, 2)
IDField1=Field.Make("IDProf", #Field_Short, 2, 0)
RW1Field1=Field.Make("RWAnf", #Field_Float, 10, 2)
RW2Field1=Field.Make("RWEnd", #Field_Float, 10, 2)
GeoField1=Field.Make("Geologie", #Field_Char, 30, 0)
ListofFlds1={IDHWField1, HWField1, IDField1, RW1Field1,
          RW2Field1, GeoField1}
PLgFTab.AddFields(ListofFlds1)
ListofFlds2={ShapeField1, IDHWField1, HWField1,
          IDField1, RW1Field1, RW2Field1, GeoField1}
PLgFTab.SetEditable(false)
PLgFTab.SetEditable(true)
av.ShowMsg("Zuweisung der geologischen Namen in Profil ...")
av.ShowStopButton
recNr=-1
'Ablesen der PolyLine vom Input-Thema
for each gzd in 0..Anz2DPLIdx
Ng=gzd+1
the2DPolyL=PLFTab.ReturnValue(PLShpFld, gzd)
theIDHW=PLFTab.ReturnValue(PLIDFld, gzd)
theHW=PLFTab.ReturnValue(PLHWFld, gzd)
ListofGeolog={}
ListofPg={}
for each aShape in 0..AnzGRIdx
   theShape=GeoFTab.ReturnValue(GeoShpFld, aShape)
   if (the2DPolyL.Intersects(theShape)) then
     theGeologie=GeoFTab.ReturnValue(GeoFld, aShape)
     ListofGeolog.Add(theGeologie)
     ListofPg.Add(theShape)
   end
```

```
end
AnzPg=ListofPg.Count
AnzPgIdx=AnzPg-1
if (AnzPg <> 0) then
  ListofGeo={}
  ListofNewPolyL={}
  for each aShape in 0..AnzPgIdx
    theGeologie=ListofGeolog.Get(aShape)
    theShape=ListofPg.Get(aShape)
    NewPolyL=the2DPolyL.LineIntersection(theShape)
    ListofCheck1=NewPolyL.AsList
    Check1=ListofCheck1.Count
    if (Check1 <> 0) then
      ListofTeile=NewPolyL.Explode
      AnzderTeile=ListofTeile.Count
      AnzdTIdx=AnzderTeile-1
      for each j in 0..AnzdTIdx
        N=j+1
        ListofGeo.Add(theGeologie)
        NewPolyLT=ListofTeile.Get(j)
        ListofNewPolyL.Add(NewPolyLT)
      end
    end
  end
  AnzNewPL=ListofNewPolyL.Count
  AnzNPLIdx=AnzNewPL-1

  RWMin=2583000
  RWMax=0
  ListofRWAnf={}
  ListofRWEnd={}
  for each i in 0..AnzNPLIdx
     N=i+1
     Fg=ListofNewPolyL.Get(i)
     ListofFg=Fg.AsList
     RWAnf=ListofFg.Get(0).Get(0).Getx
     ListofRWAnf.Add(RWAnf)
     RWEnd=ListofFg.Get(0).Get(1).Getx
     ListofRWEnd.Add(RWEnd)
     if (RWAnf < RWMin) then
       RWMin=RWAnf
     end
     if (RWEnd > RWMax) then
       RWMax=RWEnd
     end
  end
  if (RWMin > PLRWAnf) then
    theExtraPL=PolyLine.Make({{PLRWAnf@theHW, RWMin@theHW}})
    ListofNewPolyL.Add(theExtraPL)
    ListofRWAnf.Add(PLRWAnf)
    ListofRWEnd.Add(RWMin)
    ListofGeo.Add("Aussen")
    AnzNPLIdx=AnzNPLIdx+1
  end
  if (RWMax < PLRWEnd) then
    theExtraPL=PolyLine.Make({{RWMax@theHW, PLRWEnd@theHW}})
    ListofNewPolyL.Add(theExtraPL)
    ListofRWAnf.Add(RWMax)
    ListofRWEnd.Add(PLRWEnd)
    ListofGeo.Add("Aussen")
    AnzNPLIdx=AnzNPLIdx+1
  end
```

```
  AnzNPL2=AnzNPLIdx+1
  for each i in 0..AnzNPLIdx
     N=i+1
     M=N.AsString
     theGeologie=ListofGeo.Get(i)
     ListofValue={}
     theShapeV=ListofNewPolyL.Get(i)
     ListofValue.Add(theShapeV)
     ListofValue.Add(theIDHW)
     ListofValue.Add(theHW)
     ListofValue.Add(i)
     ListofValue.Add(ListofRWAnf.Get(i))
     ListofValue.Add(ListofRWEnd.Get(i))
     ListofValue.Add(theGeologie)
     recNr=recNr+1
     PLgFTab.AddRecord
     for each j in 0..6
        PLgFTab.SetValue(ListofFlds2.Get(j), recNr,
                  ListofValue.Get(j))
     end
  end
end
'Show percentage complete with enabled stop button
more=av.SetStatus(Ng/Anz2DPL*100)
if (not more) then
   break
end
end
PLgFTab.SetEditable(false)
thmNew=FTheme.Make(PLgFTab)
theView.AddTheme(thmNew)
```

```
'pfql3dmg.ave
'2D-Querprofilschnittlinien mit geologischen Bezeichnungen in einem
'Thema werden durch Punkte mit Höhenwerten und eine interpolierte Fläche
'in 3D-Querprofilschnittlinien umgewandelt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc     'Aktives Karten-View
myScript=theProject.FindScript("pfql3dmg")
myScript.SetNumberFormat("d.dd")

'Eingabe der Datei, die 2D-Shapes zur Umwandlung in 3D enthält
ListofThemes=theView.GetThemes
ListofPLFThm = {}
ListofPtFThm = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aSCN = aFTab.GetShapeClass.GetClassName
      if (aSCN = "PolyLine") then
         ListofPLFThm.Add(aT)
      elseif (aSCN = "Point") then
         ListofPtFThm.Add(aT)
      end
   end
```

```
end

PLTheme=MsgBox.ChoiceAsString(ListofPLFThm,
      "das die geologischen 2D-PolyLine enthält",
      "Eingabe eines Themas im View"++theView.AsString)
PLFTab=PLTheme.GetFTab
ListofPLFlds = PLFTab.GetFields

PLShpFld=PLFTab.FindField("Shape")
PLIDFld=PLFTab.FindField("IDHW")
PLHWFld=PLFTab.FindField("HW")
PLIDPfFld=PLFTab.FindField("IDProf")
RWAnfFld=PLFTab.FindField("RWAnf")
RWEndFld=PLFTab.FindField("RWEnd")
PLGeoFld = MsgBox.ListAsString(ListofPLFlds,
        "für geologische Kennwörter der Datensätze",
        "Auswahl eines Feldes im"++PLTheme.AsString)

'Feststellung der Anzahl der 2D-PolyLine in dem Thema
Anz2DPL=0
for each rec in PLFTab
   Anz2DPL=Anz2DPL+1
end
Anz2DPLIdx=Anz2DPL-1
MsgBox.Info(Anz2DPL.AsString, "Anzahl der 2D-PolyLine in dem Thema")

'Eingabe der Datei, die Punkte mit Hoehenwerten enthält,
'und in 3D uebernommen werden soll
PTTheme=MsgBox.ChoiceAsString(ListofPtFThm,
       "das die Punkte mit Hoehenwerten enthält",
       " Eingabe eines Themas")
t=PTTheme
PTFTab=PTTheme.GetFTab
ListofFlds=PTFTab.GetFields

PTShpFld=MsgBox.ChoiceAsString(ListofFlds,
       "Eingabe des Feldes für Shape",
       "Auswahl eines Feldes in Thema:"++PTTheme.AsString)
PTHhFld=MsgBox.ChoiceAsString(ListofFlds,
       "Eingabe des Feldes für Höhen [m]",
       "Auswahl eines Feldes in Thema:"++PTTheme.AsString)
ListofPtFld={PTShpFld, PTHhFld}

'Eingabe der interpolierten Datei mit den Höhenwerten (Grid oder Tin)
done=False
While (not done)
  surfaceList = {}
  for each t2 in theView.GetThemes
    if (t2.Is(GTheme) or t2.Is(STheme)) then
      surfaceList.Add(t2)
    end
  end
  if (surfaceList.Count = 0) then
    aSurfFN = SourceManager.GetDataSet({Grid,Tin},"Select Surface :"
          ++ t.GetName)
    if (aSurfFN = NIL) then
      continue
    end
    aSrcName = Grid.MakeSrcName(aSurfFN.AsString)
    if (aSrcName <> NIL) then
      theSurface = Grid.Make(aSrcName)
      surfTheme = GTheme.Make(theSurface)
```

```
    else
      aSrcName = SrcName.Make(aSurfFN.AsString)
      theSurface = Tin.Make(aSrcName)
      surfTheme = STheme.Make(theSurface)
    end
    theView.AddTheme(surfTheme)
  else
    surfTheme = MsgBox.ListAsString(surfaceList,
     "Choose theme to use as surface:","Select Surface :"++ t.GetName)
    if (surfTheme = NIL) then
      continue
    end
    if (surfTheme.Is(GTheme)) then
      theSurface = surfTheme.GetGrid
    elseif (surfTheme.Is(STheme)) then
      theSurface = surfTheme.GetSurface
    else
      continue
    end
  end
  done=True
end

'Ein Feature-Shape-File für Geologie (PolyLineZ) wird hergestellt
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("pl2dmgeo","shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("pl3dmgg1","shp")
fName=FileDialog.Put(fnStr, "*.shp","Output 3D shape File (PolyLineZ)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PL3FTab=FTab.MakeNew(fName, PolyLineZ)
ShapeField1=PL3FTab.FindField("shape")
IDHWField1=Field.Make("IDHW", #Field_Short, 4, 0)
HWField1=Field.Make("HW", #Field_Float, 10, 2)
IDField1=Field.Make("IDProf", #Field_Short, 2, 0)
RW1Field1=Field.Make("RWAnf", #Field_Float, 10, 2)
RW2Field1=Field.Make("RWEnd", #Field_Float, 10, 2)
GeoField1=Field.Make("Geologie", #Field_Char, 20, 0)
Listof3DFlds1={IDHWField1, HWField1, IDField1,
            RW1Field1, RW2Field1, GeoField1}
PL3FTab.AddFields(Listof3DFlds1)
Listof3DFlds2={ShapeField1, IDHWField1, HWField1,
            IDField1, RW1Field1, RW2Field1, GeoField1}

PL3FTab.SetEditable(false)
PL3FTab.SetEditable(true)

av.ShowMsg("Zuweisung der Z Werten in Profil ...")
av.ShowStopButton

aPrj=Prj.MakeNull
recNr=-1
'Ablesen der PolyLine vom Input-Theme
for each gzd in 0..Anz2DPLIdx
  Ng=gzd+1
  the2DPolyL=PLFTab.ReturnValue(PLShpFld, gzd)
  theIDHW=PLFTab.ReturnValue(PLIDFld, gzd)
  theHW=PLFTab.ReturnValue(PLHWFld, gzd)
  theIDPf=PLFTab.ReturnValue(PLIDPfFld, gzd)
  RWAnf=PLFTab.ReturnValue(RWAnfFld, gzd)
  RWEnd=PLFTab.ReturnValue(RWEndFld, gzd)
  aGeolog=PLFTab.ReturnValue(PLGeoFld, gzd)
```

```
Listof2DValue={the2DPolyL, theIDHW, theHW,
            theIDPf, RWAnf, RWEnd, aGeolog}

'Die Punkte für die PolyLine werden bestimmt
ListofRWPt={}
ListofZValue={}

VGI=((RWAnf-2558600)/50).Floor
RWAnfVInt=2558600+(VGI*50)
ListofRWPt.Add(RWAnfVInt)

NKI=((RWAnf-2558600)/50).Ceiling
RWAnfInt=2558600+(NKI*50)
VGI=((RWEnd-2558600)/50).Floor
RWEndInt=2558600+(VGI*50)
RWEndVInt=RWEndInt
RWIntv=VGI-NKI

for each Abst in 0..RWIntv
  RWPtInt=RWAnfInt+(Abst*50)
  ListofRWPt.Add(RWPtInt)
end

NKI=((RWEnd-2558600)/50).Ceiling
RWEndNInt=2558600+(NKI*50)
ListofRWPt.Add(RWEndNInt)

IntPIdx=(ListofRWPt.Count)-1

ListofPtFld={PTShpFld, PTHhFld}
for each Pt in 0..IntPIdx
  RWPt=ListofRWPt.Get(Pt)
  RWIdxAbst=((RWPt-2558600)/50)
  TabIdx=((theHW-5618450)/50)*478+RWIdxAbst
  TIdx=TabIdx.Truncate
  TIdxDf=TabIdx-TIdx
  if (TIdxDf=0) then
    aShapeValue=PTFTab.ReturnValue(PTShpFld, TabIdx)
    RWValue=aShapeValue.Getx
    HWValue=aShapeValue.Gety
    if ((RWValue=RWPt) and (HWValue=theHW)) then
      theZValue=PTFTab.ReturnValue(PTHhFld, TabIdx).SetFormat("d.dd")
    else
      MsgBox.Error("Falsche Daten in der Tabelle wurde gesucht!",
            "Fehler Meldung")
      MsgBox.Info(RWPt.AsString, "RWPt von Liste"
            ++"vom"++(RWPt).AsString)
      MsgBox.Info(RWValue.AsString, "RWValue vom Point"
            ++"vom"++(RWPt).AsString)
      MsgBox.Info(theHW.AsString, "theHW von der List"
            ++"vom"++(theHW).AsString)
      MsgBox.Info(HWValue.AsString, "HWValue vom Point"
            ++"vom"++(theHW).AsString)
      exit
    end
  else
    New2DPt=(RWPt@theHW)
    theZValue=theSurface.PointValue(New2DPt, aPrj).SetFormat("d.dd")
  end
  ListofZValue.Add(theZValue)
end
```

```
RWAnfVIntHoehe=ListofZValue.Get(0)
RWAnfNIntHoehe=ListofZValue.Get(1)
RWEndVIntHoehe=ListofZValue.Get(IntPIdx-1)
RWEndNIntHoehe=ListofZValue.Get(IntPIdx)
RWAnfHoehe=(((RWAnf-RWAnfVInt)/50)*(RWAnfNIntHoehe-RWAnfVIntHoehe))
        +RWAnfVIntHoehe
RWEndHoehe=(((RWEnd-RWEndVInt)/50)*(RWEndNIntHoehe-RWEndVIntHoehe))
        +RWEndVIntHoehe

ListofZValue.Set(0, RWAnfHoehe)
ListofRWPt.Set(0, RWAnf)
ListofZValue.Set(IntPIdx, RWEndHoehe)
ListofRWPt.Set(IntPIdx, RWEnd)

Idx3DPL=ListofRWPt.Count-1
ListofRW3D={}
ListofVL3D={}
ListofNew3DPt={}
RW1Koord=ListofRWPt.Get(0)
RWLKoord=ListofRWPt.Get(Idx3DPL)
Value1=ListofZValue.Get(0)
ValueL=ListofZValue.Get(Idx3DPL)

ListofRWPt.Remove(Idx3DPL)
ListofRWPt.Remove(0)
ListofZValue.Remove(Idx3DPL)
ListofZValue.Remove(0)

ListofRW3D.Add(RW1Koord)
ListofVL3D.Add(Value1)

IdxPL3D=ListofRWPt.Count-1
for each Ptx in 0..IdxPL3D
   RWKoord=ListofRWPt.Get(Ptx)
   if (RWKoord <> RW1Koord)  then
     if (RWKoord <> RWLKoord) then
       ListofRW3D.Add(RWKoord)
       RWHoehe=ListofZValue.Get(Ptx)
       ListofVL3D.Add(RWHoehe)
     end
   end
end

ListofRW3D.Add(RWLKoord)
ListofVL3D.Add(ValueL)

IdxPL3D=ListofRW3D.Count-1
for each Ptx in 0..IdxPL3D
   RWKoord=ListofRW3D.Get(Ptx)
   RWHoehe=ListofVL3D.Get(Ptx)
   New3DPt=(RWKoord@theHW@RWHoehe)
   ListofNew3DPt.Add(New3DPt)
end

'Aus den Punkten wird eine 3D-PolyLine hergestellt.
ListofListofPt={}
ListofListofPt.Add(ListofNew3DPt)
New3DPolyL=PolyLineZ.Make(ListofListofPt)
Listof3DFlds2={ShapeField1, IDHWField1, HWField1, IDField1,
          RW1Field1, RW2Field1, GeoField1}
Listof2DValue={the2DPolyL, theIDHW, theHW, theIDPf,
          RWAnf, RWEnd, aGeolog}
```

```
  Listof3DValue={New3DPolyL, theIDHW, theHW, theIDPf,
                 RWAnf, RWEnd, aGeolog}

  recNr=recNr+1
  PL3FTab.AddRecord
  PL3FTab.SetValue(ShapeField1, recNr, New3DPolyL)
  PL3FTab.SetValue(IDHWField1, recNr, theIDHW)
  PL3FTab.SetValue(HWField1, recNr, theHW)
  PL3FTab.SetValue(IDField1, recNr, theIDPf)
  PL3FTab.SetValue(RW1Field1, recNr, RWAnf)
  PL3FTab.SetValue(RW2Field1, recNr, RWEnd)
  PL3FTab.SetValue(GeoField1, recNr, aGeolog)

  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/Anz2DPL*100)
  if (not more) then
    break
  end
end
PL3FTab.SetEditable(false)
thmNew=FTheme.Make(PL3FTab)
theView.AddTheme(thmNew)




'pfql3dog.ave
'Alle Zweidimensionalen Querprofilschnittlinien werden durch Punkte
'mit Höhenwerten in 3D-Profilschnittlinien umgewandelt. Die Profilschnittlinien
'und die Punkte des Punkt-Themas liegen in einem Karten-View.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theKtView=av.GetActiveDoc    'Ein aktives Karten-View
myScript=theProject.FindScript("pfql3dog")
myScript.SetNumberFormat("d.dd")

'Eingabe der Datei, die 2D-Shapes zur Umwandlung in 3D enthält
ListofThemes=theKtView.GetThemes
ListofPLThms = {}
ListofPtThms = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     theClass=aFTab.GetShapeClass
     if (theClass.IsSubclassOf(PolyLine)) then
       ListofPLThms.Add(aT)
     elseif (theClass.IsSubclassOf(Point)) then
       ListofPtThms.Add(aT)
     end
   end
end
PLTheme=MsgBox.ChoiceAsString(ListofPLThms,
       "das die 2D-Querprofillinien enthält",
       " Eingabe eines Themas im View"++ theKtView.AsString)
PLFTab=PLTheme.GetFTab
PLShpFld=PLFTab.FindField("Shape")
PLIDFld=PLFTab.FindField("IDHW")
PLHWFld=PLFTab.FindField("HW")
RWAnfFld=PLFTab.FindField("RWAnf")
RWEndFld=PLFTab.FindField("RWEnd")
```

```
'Feststellung der Anzahl, des kleinsten und des
'größten HWs der 2D-PolyLine in dem Thema
Anz2DPL=0
maxHW=0.00
minHW=5642000.00
for each rec in PLFTab
   Anz2DPL=Anz2DPL+1
   theHW=PLFTab.ReturnValue(PLHWFld, rec)
   if (theHW < minHW) then
     minHW=theHW
   end
   if (theHW > maxHW) then
     maxHW=theHW
   end
end
Anz2DPLIdx=Anz2DPL-1
MsgBox.Info(Anz2DPL.AsString, "Anzahl der 2D-PolyLine in dem Thema")
'Eingabe der Datei, die Punkte mit Hoehenwerten enthält
PTTheme=MsgBox.ChoiceAsString(ListofPtThms,
       "das die Punkte mit Hoehenwerten enthält",
       " Eingabe eines Themas im View"++ theKtView.AsString)
PTFTab=PTTheme.GetFTab
ListofFldNm=PTFTab.GetFields
PTHhFld=MsgBox.ListAsString(ListofFldNm,"in der Punkt-Datei"
    ++ PTTheme.AsString, "Auswahl eines Feldes mit Höhenwerten")
PTShpFld=PTFTab.FindField("Shape")

'Eingabe der interpolierten Datei mit den Höhenwerten (Grid oder Tin)
done=False
While (not done)
  surfaceList = {}
  for each t2 in ListofThemes
    if (t2.Is(GTheme) or t2.Is(STheme)) then
      surfaceList.Add(t2)
    end
  end
  if (surfaceList.Count = 0) then
    aSurfFN = SourceManager.GetDataSet({Grid,Tin},
            "Select Surface vom:" ++ PTTheme.GetName)
    if (aSurfFN = NIL) then
      continue
    end
    aSrcName = Grid.MakeSrcName(aSurfFN.AsString)
    if (aSrcName <> NIL) then
      theSurface = Grid.Make(aSrcName)
      surfTheme = GTheme.Make(theSurface)
    else
      aSrcName = SrcName.Make(aSurfFN.AsString)
      theSurface = Tin.Make(aSrcName)
      surfTheme = STheme.Make(theSurface)
    end
    theKtView.AddTheme(surfTheme)
  else
    surfTheme = MsgBox.ListAsString(surfaceList,
         "Auswahl eines Themas, um als surface zu benutzen:",
         "Auswahl des Surface für:" ++ PTHhFld.AsString)
    if (surfTheme = NIL) then
      continue
    end
    if (surfTheme.Is(GTheme)) then
      theSurface = surfTheme.GetGrid
```

```
    elseif (surfTheme.Is(STheme)) then
      theSurface = surfTheme.GetSurface
    else
      continue
    end
  end
  done=True
end

'Ein Feature-Shape-File für einen Profilschnitt (3D) wird hergestellt
WDStr=av.GetProject.GetWorkDir.AsString
FnStr0= "D"+PLTheme.AsString
fnStr=FileName.Make(WDStr).MakeTmp(FnStr0,"shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(FnStr0,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output 3D shape File (PolyLineZ)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PL3dFTab=FTab.MakeNew(fName, PolyLineZ)
ShapeField1=PL3dFTab.FindField("shape")
IDHWField1=Field.Make("IDHW", #Field_Short, 4, 0)
HWField1=Field.Make("HW", #Field_Float, 10, 2)
RW1Field1=Field.Make("RWAnf", #Field_Float, 10, 2)
RW2Field1=Field.Make("RWEnd", #Field_Float, 10, 2)
ListofFlds1={IDHWField1, HWField1, RW1Field1, RW2Field1}
PL3dFTab.AddFields(ListofFlds1)

av.ShowMsg("Zuweisung der Z Werten in Profilschnitt ...")
av.ShowStopButton
aPrj=Prj.MakeNull
recNr=-1
PrfAbst=MsgBox.Input("Rasterabstand der Punkte auf einem Profil [m]:",
                "Eingabe der Daten:", "50")
PrfAbNr=PrfAbst.AsNumber
PL3dFTab.SetEditable(false)
PL3dFTab.SetEditable(true)
for each gzd in 0..Anz2DPLIdx
  Ng=gzd+1
  the2DPolyL=PLFTab.ReturnValue(PLShpFld, gzd)
  theIDHW=PLFTab.ReturnValue(PLIDFld, gzd)
  theHW=PLFTab.ReturnValue(PLHWFld, gzd)
  RWAnf=PLFTab.ReturnValue(RWAnfFld, gzd)
  RWEnd=PLFTab.ReturnValue(RWEndFld, gzd)
  AnzSpalt=((RWEnd-RWAnf)/PrfAbNr)+1

  'Die Zwischen Punkte für die PolyLine werden bestimmt
  ListofNew3DPt={}
  PtIdx=(RWEnd-RWAnf)/PrfAbNr
  for each SpIdx in 0..PtIdx
    RWPt=RWAnf+(SpIdx*PrfAbNr)
    TabIdx=(((theHW-minHW)/PrfAbNr)*AnzSpalt)+SpIdx
    TIdx=TabIdx.Truncate
    TIdxDf=TabIdx-TIdx
    if (TIdxDf=0) then
      aShapeValue=PTFTab.ReturnValue(PTShpFld, TabIdx)
      RWValue=aShapeValue.Getx
      HWValue=aShapeValue.Gety
      if ((RWValue=RWPt) and (HWValue=theHW)) then
        theZValue=PTFTab.ReturnValue(PTHhFld, TabIdx)
      else
        MsgBox.Error("Falsches Daten in der Tabelle wurde gesucht!","")
        exit
      end
```

```
    else
      New2DPt=(RWPt@theHW)
      theZValue=theSurface.PointValue(New2DPt, aPrj).SetFormat("d.dd")
    end
    New3DPt=(RWPt@theHW@theZValue)
    ListofNew3DPt.Add(New3DPt)
  end
  'Aus den Punkten wird eine 3D-PolyLine hergestellt
  ListofListofPt={}
  ListofListofPt.Add(ListofNew3DPt)
  New3DPolyL=PolyLineZ.Make(ListofListofPt)
  recNr=recNr+1
  PL3dFTab.AddRecord
  PL3dFTab.SetValue(ShapeField1, recNr, New3DPolyL)
  PL3dFTab.SetValue(IDHWField1, recNr, theIDHW)
  PL3dFTab.SetValue(HWField1, recNr, theHW)
  PL3dFTab.SetValue(RW1Field1, recNr, RWAnf)
  PL3dFTab.SetValue(RW2Field1, recNr, RWEnd)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/Anz2DPL*100)
  if (not more) then
    break
  end
end
PL3dFTab.SetEditable(false)
thmNew=FTheme.Make(PL3dFTab)
theKtView.AddTheme(thmNew)
```

```
'pfqlhrst.ave
'2D-Querprofilschnittlinien (z.B. 453 Linien) werden als PolyLine in
'einem Karten-View hergestellt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("pfqlhrst")
myScript.SetNumberFormat("d.dd")

'Ein Feature-Shape-File für Querprofilschnittlinien wird hergestellt
WDStr=av.GetProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("pl2d453","shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("pl2d453","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PolyLine)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PlFTab=FTab.MakeNew(fName, PolyLine)
ShapeField1=PlFTab.FindField("shape")
IDHWField1=Field.Make("IDHW", #Field_Short, 4, 0)
HWField1=Field.Make("HW", #Field_Float, 10, 2)
RW1Field1=Field.Make("RWAnf", #Field_Float, 10, 2)
RW2Field1=Field.Make("RWEnd", #Field_Float, 10, 2)
ListofFlds1={IDHWField1, HWField1, RW1Field1, RW2Field1}
PlFTab.AddFields(ListofFlds1)
ListofFlds2={ShapeField1, IDHWField1, HWField1, RW1Field1, RW2Field1}

av.ShowMsg("Herstellung der Querprofilschnittlinien ...")
av.ShowStopButton
```

```
ListofPL2d={}
ListofIDHW={}
ListofHWNr={}
KHWStr=MsgBox.Input("Der kleinste Hochwert der Profilschnttlinien",
                "Eingabe der Daten der Profile", "5618450.00")
GHWStr=MsgBox.Input("Der größte Hochwert der Profilschnittlinien",
                "Eingabe der Daten der Profile", "5641050.00")
AbstStr=MsgBox.Input("Der HW-Abstand zwischen Profilschnittlinien",
                 "Eingabe der Daten der Profile", "50.00")
KHWNr=KHWStr.AsNumber
GHWNr=GHWStr.AsNumber
AbstNr=AbstStr.AsNumber
AnzPL=((GHWNr-KHWNr)/AbstNr)+1
AnzPLIdx=AnzPL-1
ARWStr=MsgBox.Input("Der kleinste Rechtswert der Profilschnittlinien",
                "Eingabe der Daten der Profile", "2558600.00")
ERWStr=MsgBox.Input("Der größte Rechtswert der Profilschnittlinien",
                "Eingabe der Daten der Profile", "2582450.00")
RWAnfNr=ARWStr.AsNumber
RWEndNr=ERWStr.AsNumber

for each i in 0..AnzPLIdx
   N=i+1
   HWNr=KHWNr+(AbstNr*i)
   thePolyL=PolyLine.Make({{RWAnfNr@HWNr, RWEndNr@HWNr}})
   IDHWNr=i
   ListofPL2d.Add(thePolyL)
   ListofIDHW.Add(IDHWNr)
   ListofHWNr.Add(HWNr)
   'Show percentage complete with enabled stop button
   more=av.SetStatus(N/AnzPL*100)
   if (not more) then
      break
   end
   if (HWNr > GHWNr) then
      break
   end
end

av.ShowMsg("Speicherung der Profile in der Tabelle...")
av.ShowStopButton

AnzPL=ListofPL2d.Count
AnzPLIdx=AnzPL-1
MsgBox.Info(AnzPL.AsString, "Anzahl der 2D PolyLine in der Liste")
AnzFlds=ListofFlds2.Count
FldIdx=AnzFlds-1

PlFTab.SetEditable(false)
PlFTab.SetEditable(true)

for each i in 0..AnzPLIdx
   N=i+1
   ListofValue={}
   PlFTab.AddRecord
   theShapeV=ListofPL2d.Get(i)
   ListofValue.Add(theShapeV)
   IDHWNr=ListofIDHW.Get(i)
   ListofValue.Add(IDHWNr)
   HWNr=ListofHWNr.Get(i)
   ListofValue.Add(HWNr)
```

```
    ListofValue.Add(RWAnfNr)
    ListofValue.Add(RWEndNr)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(N/AnzPL*100)
    if (not more) then
      break
    end
    for each j in 0..FldIdx
       PlFTab.SetValue(ListofFlds2.Get(j), i, ListofValue.Get(j))
    end
    if (HWNr > GHWNr) then
      break
    end
end
PlFTab.SetEditable(false)
thmNew=FTheme.Make(PlFTab)
theView.AddTheme(thmNew)
```

```
'pgaus4pl.ave
'Ein Polyline-Thema mit mehreren Datensätzen in einem aktiven
'Profilschnitt-View wird in Polygone umgewandelt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc      'ein aktives View
myScript=theProject.FindScript("pgaus4pl")
myScript.SetNumberFormat( "d.dd") ' script default

theTheme=theView.GetActiveThemes.Get(0)  'Aktives Thema
ThStr=theTheme.AsString

AW11 = MsgBox.YesNo("Ist das aktive Thema"+" "++ThStr++"richtig?",
          "Feststellung des Themas", true)
if (Not AW11) then
  MsgBox.Error("Das aktive Thema ist falsch!"+NL+
            "Das Programm wird abgebrochen!", "")
  return (nil)
end

PLFTab = theTheme.GetFTab
PLShpFld = PLFTab.FindField("Shape")
ListofFlds = PLFTab.GetFields
PLKWFld = MsgBox.ChoiceAsString(ListofFlds,"für Kennwort des Datensatzes",
       "Auswahl eines Feldes im Thema"++theTheme.AsString)
IdxKWFld = ListofFlds.FindByValue(PLKWFld)

MsgBox.Info(theTheme.AsString, "Der Name des Themas")
'Feststellung der Anzahl der PolyLine in dem Thema
AnzPL=0
for each rec in PLFTab
   AnzPL=AnzPL+1
end
AnzPLIdx=AnzPL-1

ListofFlaeche = {}
for each i in 0..AnzPLIdx
   aFlaeche = PLFTab.ReturnValue(PLKWFld, i)
```

```
   ListofFlaeche.Add(aFlaeche)
end

ListofReihen = {}
for each i in 0..AnzPLIdx
  aNr = i+1
  aNrStr = aNr.SetFormat("").SetFormat("d").AsString
  aAzStr = AnzPL.SetFormat("").SetFormat("d").AsString
  aGFL = MsgBox.ListAsString(ListofFlaeche,
    "welche die"++aNrStr++"."++"obere Fläche ist",
    "Auswahl einer von"++aAzStr++"Flächen")
  aIdx = ListofFlaeche.FindByValue(aGFL)
  ListofReihen.Add(aIdx)
end

minYStr=MsgBox.Input("die kleinste Höhe"+NL+ "der gesammten Polygone [m]",
      "Bestimmung der Koordinaten des Polygons", "0.00")
minY = minYStr.AsNumber
FaktStr=MsgBox.Input("zur Überhöhung des Profilschnttes",
      "Eingabe eines Faktors", "50")
minYF = (FaktStr.AsNumber)*(minY)

recNr=AnzPLIdx
ListofListofPt = {}
minx = 3000000
maxx = 0

av.ShowMsg("Feststellung der Koordinaten des Themas im View"
    ++(theView.AsString)++"...")
av.ShowStopButton
for each i in 0..AnzPLIdx
  Ng=i+1
  theShape = PLFTab.ReturnValue(PLShpFld, i)
  theProfilList1={}
  theProfilList1.Add({theShape})
  ListofPt = {}
  '2D PolyLine wird in List der Koordinaten umgewandelt.
    if (theProfilList1 <> 0) then
      for each q in theProfilList1
        theLines=q.Get(0).AsList
        for each m in theLines
          for each ptx in m
            myx=ptx.Getx
            myy=ptx.Gety
            ListofPt.Add(myx@myy)
            if (myx < minx) then
              minx = myx
            end
            if (myx > maxx) then
              maxx = myx
            end
          end
        end
      end
    end
    ListofListofPt.Add(ListofPt)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzPL*100)
    if (not more) then
      exit
    end
end
```

```
av.ShowMsg("Umwandlung der Profilschnitte"++theTheme.AsString++"in Polygone ...")
ListofListofPgPts={}
ListofFLBz = {}
for each i in 0..AnzPLIdx
  aIdx = ListofReihen.Get(i)
  aListofPt = ListofListofPt.Get(aIdx)
  aNr = i + 1
  aFL = ListofFlaeche.Get(aIdx)
  if (i < AnzPLIdx) then
    aIdx2 = ListofReihen.Get(aNr)
    aListofPt2 = ListofListofPt.Get(aIdx2)
    AnzPt2 = aListofPt2.Count
    IdxPt2 = AnzPt2 - 1
    aFL2 = ListofFlaeche.Get(aIdx2)
    for each j in 0..IdxPt2
      UmkIdx = IdxPt2 - j
      aPt2 = aListofPt2.Get(UmkIdx)
      aListofPt.Add(aPt2)
    end
  elseif (i = AnzPLIdx) then
    aListofPt.Add(maxx@minYF)
    aListofPt.Add(minx@minYF)
    aFL2 = minYStr
  end
  ListofListofPgPts.Add(aListofPt)
  ListofFLBz.Add(aFL++"und"++aFL2)
end

av.ShowMsg("Speicherung der neuen 2D-Polygone"++"...")
ListofPolygs = {}
ListofGeol = {"Deckschichten", "Niederterrassen",
          "Mittelterrassen", "Präquartäre Schichten"}
ListofKW = {"D", "NT", "MT", "Präq"}
ListofPgBz = {}
ListofaKW = {}
AnzPg = ListofListofPgPts.Count
IdxPg = AnzPg - 1
for each aPg in 0..IdxPg
  aNr = aPg + 1
  aNrStr= aNr.SetFormat("").SetFormat("d").AsString
  aFL = ListofFLBz.Get(aPg)
  aBZ = ListofGeol.Get(aPg)
  aPGName = MsgBox.Input("für die"++aNrStr++"."++"obere Schicht"
          +NL+"(für das Polygon von den Flächen"++aFL+")",
          "Eingabe der Bezeichnung", aBZ)
  ListofPgBz.Add(aPGName)
  aBZIdx = ListofGeol.FindByValue(aPGName)
  aKW = ListofKW.Get(aBZIdx)
  ListofaKW.Add(aKW)
  ListofListofPoint = {}
  myList = ListofListofPgPts.Get(aPg)
  ListofListofPoint.Add(myList)
  thePolyg=Polygon.Make(ListofListofPoint)
  ListofPolygs.Add(thePolyg)
end

'Ein Feature-Shape-File für Schichten (Polygon) wird hergestellt.
WDStr=theProject.GetWorkDir.AsString
AnzStr = ThStr.Count
aNStr = ThStr.Right(AnzStr - 1)
aListofStr = aNStr.AsTokens(".shp")
```

```
fnStr = "N"+(aListofStr.Get(0))
fnName=FileName.Make(WDStr).MakeTmp(fnStr,"shp")
'fnName=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fnStr, "shp")
fnPg=FileDialog.Put(fnName, "*.shp", "Output shape File (Polygon)")
if (fnPg=nil) then exit end
fnPg.SetExtension("shp")
FTabPg=FTab.MakeNew(fnPg, Polygon)
ShpFld=FTabPg.FindField("shape")

AnzFlds = ListofFlds.Count
IdxFlds = AnzFlds - 1
ListofNFld1 = {}
for each i in 0..IdxFlds
  if (i > 0) then
    aFLFld = ListofFlds.Get(i)
    aPgFld = aFLFld.Clone
    if (i = IdxKWFld) then
      aPgFld = Field.Make("Schichten", #Field_Char, 23, 0)
    end
    ListofNFld1.Add(aPgFld)
  end
end
aPgFld = Field.Make("Kennwort", #Field_Char, 6, 0)
ListofNFld1.Add(aPgFld)
FTabPg.AddFields(ListofNFld1)

ListofListofValue = {}
for each j in 0..AnzPLIdx
 ListofValue = {}
 aIdx = ListofReihen.Get(j)
 for each i in 0..IdxFlds
   if (i > 0) then
     if (i <> IdxKWFld) then
       aValue = PLFTab.ReturnValue(ListofFlds.Get(i), aIdx)
     elseif (i = IdxKWFld) then
       aValue = ListofPgBz.Get(j)
     end
     ListofValue.Add(aValue)
   end
 end
 ListofValue.Add(ListofaKW.Get(j))
 ListofListofValue.Add(ListofValue)
end

FTabPg.SetEditable(false)
FTabPg.SetEditable(true)
ListofPgFlds = FTabPg.GetFields
AnzPgFlds = ListofPgFlds.Count
IdxPgFlds = AnzPgFlds - 1
for each aRec in 0..IdxPg
   FTabPg.AddRecord
   myPolyg=ListofPolygs.Get(aRec)
   aListofValue = ListofListofValue.Get(aRec)
   for each j in 0..IdxPgFlds
     if (j = 0) then
       aValue = myPolyg
     elseif (j > 0) then
       aNr = j - 1
       aValue = aListofValue.Get(aNr)
     end
     FTabPg.SetValue(ListofPgFlds.Get(j), aRec, aValue)
   end
```

```
end
FTabPg.SetEditable(false)
thmNew=FTheme.Make(FTabPg)
theView.AddTheme(thmNew)

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette
aListofColor={}
aListofLb = {"Deckschichten","Niederterrassen",
          "Mittelterrassen","Quartärbasis","Präquartäre Schichten"}
aListofNr={47, 14, 26, 3, 3, 32, 26, 1}
for each Nmb in 0..5
  aNumb=aListofNr.Get(Nmb)
  theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
  theRgbList=theColor.GetRgbList
  aColor=Color.Make
  aColor.SetRgbList(theRgbList)
  aListofColor.Add(aColor)
end
theLegend=thmNew.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(thmNew, "Schichten")
'theLegend.SetNullValue("Schichten", "Null")
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofSColor={}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   aLbIdx = aListofLb.FindByValue(theKlasseLb)
   if (aLbIdx <> -1) then
     aColor = aListofColor.Get(aLbIdx)
   elseif (aLbIdx = -1) then
     aColor = aListofColor.Get(5)
   end
   aListofSColor.Add(aColor)
end
aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
AnzSymbIdx=AnzSymb-2
for each symb in 0..AnzSymbIdx
   aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
end
thmNew.UpdateLegend




'pgaus4pt.ave
'Ein Polygon-Thema mit mehreren Datensätzen wird aus Punkten mit
'mehreren Höhen in einem View für ein beliebiges Profil hergestellt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
'ein aktives Profilschnitt-View zur Herstellung der Polygone
PfView=av.GetActiveDoc 'ein aktives Profilschnitt-View

'Eingabe der Datei, die Punkte auf einer Profillinie enthält.
```

```
ListofKtViews={"Kt1_gg", "Kt1_ggd", "Kt1_hg", "Kt1_hgd", "Kt1_tga",
            "Kt1_tgd","Kt2_gg", "Kt2_ggd", "Kt2_hg", "Kt2_hgd",
            "Kt2_tga", "Kt2_tgd"}

KtViewStr = MsgBox.ChoiceAsString(ListofKtViews,
     "wo sich ein Punkt-Thema mit den Höhen-Feldern"
     ++"zur Herstellung der Polygone befindet",
     "Auswahl eines Views")
KtView = theProject.FindDoc(KtViewStr)

ListofThemes = KtView.GetThemes
ListofPtThms = {}
ListofPLThms = {}
ListofPgThms = {}

for each aT in ListofThemes
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aCNm = aFTab.GetShapeClass.GetClassName
      if (aCNm = "Point") then
        ListofPtThms.Add(aT)
      elseif (aCNm = "Polyline") then
        ListofPLThms.Add(aT)
      elseif (aCNm = "Polygon") then
        ListofPgThms.Add(aT)
      end
   end
end

PTThm=MsgBox.ChoiceAsString(ListofPtThms,
     "das die Punkte auf einer Profillinie"
     +NL+"zur Herstellung der Polygone enthält",
     "Eingabe eines Themas im View"++KtView.AsString)

ThStr = PTThm.AsString
PTFTab = PTThm.GetFTab

AnzPt = 0
for each rec in PTFTab
  AnzPt = AnzPt + 1
end
IdxPt = AnzPt - 1

ListofFlds=PTFTab.GetFields

PTShpFld=MsgBox.ChoiceAsString(ListofFlds, "Eingabe des Feldes für Shape",
       "Auswahl des Feldes in Theme:"++PTThm.AsString)
PTEfFld=MsgBox.ChoiceAsString(ListofFlds,
       "Eingabe des Feldes für Entfernungen vom Anfang",
       "Auswahl des Feldes in Theme:"++PTThm.AsString)

AnzPLStr = MsgBox.Input("zur Herstellung der Polygone",
        "Anzahl der Höhen der Flächen", "4")
AnzPL = AnzPLStr.AsNumber
AnzPLIdx = AnzPL - 1

minYStr=MsgBox.Input("die kleinste Höhe"+NL+ "der gesammten Polygone [m]",
      "Bestimmung der Koordinaten des Polygons", "0.00")
minY = minYStr.AsNumber
FaktStr=MsgBox.Input("zur Überhöhung des Profilschnttes",
      "Eingabe eines Faktors", "50")
```

```
Fakt = FaktStr.AsNumber
minYF = Fakt * minY

ListofGeoFl = {"Geländehöhe", "Deckschichtenbasis (TOK)",
               "NT abgedeckte Höhe", "MT abgedeckte Höhe (QB)"}

ListofReihen = {}
ListofFlaeche = {}
ListofListofPt = {}
minx = 1000000
maxx = 0
for each j in 0..AnzPLIdx
  Nr = j + 1
  ListofReihen.Add(j)
  aNrStr = Nr.AsString
  PtHFld=MsgBox.ChoiceAsString(ListofFlds, "Eingabe eines Feldes für Höhen [m]"
      +NL+"für die"++aNrStr+". obere Fläche",
      "Auswahl eines Feldes im Thema:"++PTThm.AsString)
  aGFl = MsgBox.ListAsString(ListofGeoFl,
       "welche die"++aNrStr++"."++"obere Flaeche darstellt",
       "Auswahl einer von"++AnzPLStr++"Bezeichnungen")
  ListofFlaeche.Add(aGFl)

  aListofPt = {}
  for each i in 0..IdxPt
    aEf = PTFTab.ReturnValue(PTEfFld, i)
    aHm = PTFTab.ReturnValue(PtHFld, i)
    aHFakt = Fakt * aHm
    aPt = Point.Make(aEf, aHFakt)
    aListofPt.Add(aPt)
    if (aEf < minx) then
      minx = aEf
    end
    if (aEf > maxx) then
      maxx = aEf
    end
  end
  ListofListofPt.Add(aListofPt)
end

ListofListofPgPts={}
ListofFLBz = {}
for each i in 0..AnzPLIdx
  aIdx = ListofReihen.Get(i)
  aListofPt = ListofListofPt.Get(aIdx)
  aNr = i + 1
  aFL = ListofFlaeche.Get(aIdx)
  if (i < AnzPLIdx) then
    aIdx2 = ListofReihen.Get(aNr)
    aListofPt2 = ListofListofPt.Get(aIdx2)
    AnzPt2 = aListofPt2.Count
    IdxPt2 = AnzPt2 - 1
    aFL2 = ListofFlaeche.Get(aIdx2)
    for each j in 0..IdxPt2
      UmkIdx = IdxPt2 - j
      aPt2 = aListofPt2.Get(UmkIdx)
      aListofPt.Add(aPt2)
    end
  elseif (i = AnzPLIdx) then
    aListofPt.Add(maxx@minYF)
    aListofPt.Add(minx@minYF)
    aFL2 = minYStr
```

```
  end
  ListofListofPgPts.Add(aListofPt)
  ListofFLBz.Add(aFL++"und"++aFL2)
end

av.ShowMsg("Speicherung der neuen 2D-Polygone"++"...")
ListofPolygs = {}
ListofGeol = {"Deckschichten", "Niederterrassen",
          "Mittelterrassen", "Präquartäre Schichten"}
ListofKW = {"D", "NT", "MT", "Präq"}
ListofPgBz = {}
ListofaKW = {}
AnzPg = ListofListofPgPts.Count
IdxPg = AnzPg - 1
for each aPg in 0..IdxPg
  aNr = aPg + 1
  aNrStr= aNr.SetFormat("").SetFormat("d").AsString
  aFL = ListofFLBz.Get(aPg)
  aBZ = ListofGeol.Get(aPg)
  aPGName = MsgBox.Input("für die"++aNrStr++"."++"obere Schicht"
          +NL+"(für das Polygon von den Flächen"++aFL+")",
          "Eingabe der Bezeichnung", aBZ)
  ListofPgBz.Add(aPGName)
  aBZIdx = ListofGeol.FindByValue(aPGName)
  aKW = ListofKW.Get(aBZIdx)
  ListofaKW.Add(aKW)
  ListofListofPoint = {}
  myList = ListofListofPgPts.Get(aPg)
  ListofListofPoint.Add(myList)
  thePolyg=Polygon.Make(ListofListofPoint)
  ListofPolygs.Add(thePolyg)
end


'Auswahl eines Themas zur Speicherung der Punkte
ListofAW={"ein neues Thema", "ein vorhandenes Thema"}
AW11=MsgBox.ChoiceAsString(ListofAW, "zur Speicherung der Punkte",
      "Auswahl eines Punkt-Themas")

if (AW11 = "ein neues Thema") then
  'Ein Feature-Shape-File für Schichten (Polygon) wird hergestellt.
  WDStr=theProject.GetWorkDir.AsString
  AnzStr = ThStr.Count
  aNStr = ThStr.Right(AnzStr - 1)
  aListofStr = aNStr.AsTokens(".shp")
  fnStr = "N"+(aListofStr.Get(0))
  fnName=FileName.Make(WDStr).MakeTmp(fnStr,"shp")
  'fnName=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fnStr, "shp")
  fnPg=FileDialog.Put(fnName, "*.shp", "Output shape File (Polygon)")
  if (fnPg=nil) then exit end
  fnPg.SetExtension("shp")
  PgFTab = FTab.MakeNew(fnPg, Polygon)
  PgShpFld = PgFTab.FindField("shape")
  PgIdFld = Field.Make("ID", #Field_Short, 6, 0)
  PgSchtFld = Field.Make("Schichten", #Field_Char, 60, 0)
  PgKwFld = Field.Make("Kennwort", #Field_Char, 10, 0)

  ListofPgFlds={PgIdFld, PgSchtFld, PgKwFld}
  PgFTab.AddFields(ListofPgFlds)
  recNr = -1

  PgTheme = FTheme.Make(PgFTab)
```

```
  PfView.AddTheme(PgTheme)

elseif (AW11 = "ein vorhandenes Thema") then
  PgTheme =MsgBox.ChoiceAsString(ListofPgThms, "Polygon-Thema",
                  "Auswahl eines Themas")
  PgFTab=PgTheme.GetFTab
  PgShpFld=PgFTab.FindField("Shape")
  PgIdFld=PgFTab.FindField("Id")
  PgSchtFld= PgFTab.FindField("Schichten")
  PgKWFld= PgFTab.FindField("Kennwort")

  'Feststellung der Anzahl der Datensätze im Thema
  AnzPgRec=0
  for each arec in PgFTab
    AnzPgRec = AnzPgRec +1
  end
  IdxPgRec= AnzPgRec -1
  recNr = IdxPgRec + 1
end

PgFTab.SetEditable(false)
PgFTab.SetEditable(true)

for each aRec in 0..IdxPg
   recNr = recNr + 1
   PgFTab.AddRecord
   myPolyg=ListofPolygs.Get(aRec)
   theGBz = ListofPgBz.Get(aRec)
   theKw = ListofaKW.Get(aRec)
   PgFTab.SetValue(PgShpFld, recNr, myPolyg)
   PgFTab.SetValue(PgIdFld, recNr, recNr)
   PgFTab.SetValue(PgSchtFld, recNr, theGBz)
   PgFTab.SetValue(PgKWFld, recNr, theKw)
end

PgFTab.SetEditable(false)
PgTheme.UpdateLegend


av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette
aListofColor={}
aListofLb = {"Deckschichten","Niederterrassen",
          "Mittelterrassen","Quartärbasis","Präquartäre Schichten"}
aListofNr={47, 14, 26, 3, 3, 32, 26, 1}
for each Nmb in 0..5
  aNumb=aListofNr.Get(Nmb)
  theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
  theRgbList=theColor.GetRgbList
  aColor=Color.Make
  aColor.SetRgbList(theRgbList)
  aListofColor.Add(aColor)
end
theLegend = PgTheme.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(PgTheme, "Schichten")
'theLegend.SetNullValue("Schichten", "Null")
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofSColor={}
```

```
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   aLbIdx = aListofLb.FindByValue(theKlasseLb)
   if (aLbIdx <> -1) then
     aColor = aListofColor.Get(aLbIdx)
   elseif (aLbIdx = -1) then
     aColor = aListofColor.Get(5)
   end
   aListofSColor.Add(aColor)
end
aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
AnzSymbIdx=AnzSymb-2
for each symb in 0..AnzSymbIdx
   aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
end
PgTheme.UpdateLegend




'pgauslkt.ave
'Eine PolyLine wird in ein Polygon umgewandelt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("pgauslkt")
myScript.SetNumberFormat( "d.dd") ' script default

ListofThemes=theView.GetThemes
ListofPLFThm = {}
ListofPgFThm = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     aClassNm= aFTab.GetShapeClass.GetClassName
     if (aClassNm = "PolyLine") then
       ListofPLFThm.Add(aT)
     elseif (aClassNm = "Polygon") then
       ListofPgFThm.Add(aT)
     end
   end
end

PLTheme=MsgBox.ChoiceAsString(ListofPLFThm,
  "PolyLine-Thema für Grenze", "Auswahl eines Themas")
PLFTab=PLTheme.GetFTab
PLShpFld=PLFTab.FindField("Shape")
PLIDFld= PLFTab.FindField("ID")
AnzPLRec = 0                'Anzahl der Datensätze im Thema
for each i in PLFTab
  AnzPLRec= AnzPLRec +1
end
IdxPLRec= AnzPLRec -1
'MsgBox.Info(AnzPLRec.AsString, "Anzahl der Datensätze im Thema"
'      ++ PLTheme.AsString)

'Auswahl eines Datensatzes
```

```
ListofPLFlds=PLFTab.GetFields
AnzPLFld=ListofPLFlds.Count
IdxPLFld=AnzPLFld-1
ListofDtStr={}
ListofFldStr = {}
FldStr = ""
for each j in 0..IdxPLRec
  DtStr=""
  for each i in 0..IdxPLFld
    aFld=ListofPLFlds.Get(i)
    aFldStr=aFld.GetName
    if (aFldStr <> "Shape") then
      FldStr = FldStr + aFldStr+"; "
      aValue=PLFTab.ReturnValue(aFld, j)
      DtStr=DtStr+aValue.AsString+"; "
    end
  end
  ListofFldStr.Add(FldStr)
  ListofDtStr.Add(DtStr)
end
aFldStr = ListofFldStr.Get(0)
aDtSStr=MsgBox.ListAsString(ListofDtStr,
     "im Thema"++ PLTheme.AsString+NL+
     "(Feldname: "++aFldStr+")", "Auswahl eines Datensatzes")
aDtSIdx=ListofDtStr.FindByValue(aDtSStr)
PLAusg=PLFTab.ReturnValue(PLShpFld, aDtSIdx)
'MsgBox.Report(PLAusg.AsString, "Das ausgewählte Shape")

'Umwandlung der PolyLine in die Liste der Punkte
av.ShowMsg("Feststellung der Koordinaten der PolyLine in Thema"
        ++(PLTheme.AsString)++"...")

theProfilList={}
theProfilList.Add({PLAusg })
ListofPt={}
'2D-Shape wird in Liste der Punkte umgewandelt.
if (theProfilList <> 0) then
  for each q in theProfilList
      theLines=q.Get(0).AsList
      for each m in theLines
         for each apt in m
            ListofPt.Add(apt)
         end
      end
  end
end

Fr1=MsgBox.YesNo("Soll noch einige zusätzliche Punkte"
   ++"am Ende eingefügt werden?","Eingabe von Stützpunkten", true)
if (Fr1) then
  ZSZahlStr=MsgBox.Input("Die Anzahl der zusätzlichen Zahlen",
                  "Eingabe der zusätzlichen Zahlen", "3")
  ZSZahl=ZSZahlStr.AsNumber.SetFormat("").SetFormat("d")
  ZSZIdx=(ZSZahl-1).SetFormat("").SetFormat("d")
  for each aZSZ in 0..ZSZIdx
     ZZ=(aZSZ+1).SetFormat("").SetFormat("d")
     strzsx1=MsgBox.Input("x-Krd. des"++ZZ.AsString+". Punktes von"
          ++ZSZahl.AsString,"Ein zusatzpunkt (im Uhrzeigersinn)",
          "2558600.00")
     strzsy1=MsgBox.Input("y-Krd. des"++ZZ.AsString+". Punktes von"
          ++ZSZahl.AsString,"Ein zusatzpunkt (im Uhrzeigersinn)",
          "5618450.00")
```

```
      zsx1=strzsx1.AsNumber
      zsy1=strzsy1.AsNumber
      ListofPt.Add(zsx1@zsy1)
  end
end
ListofListofPt={}
ListofListofPt.Add(ListofPt)
thePolyg=Polygon.Make(ListofListofPt)
if (AnzPLFld > 2) then
  Frg1=MsgBox.YesNo("fuer das neue Polygon",
       "Gibt es schon ein Kennwort in"++PLTheme.AsString, true)
  if (Frg1) then
    aKWFld=MsgBox.ChoiceAsString(ListofPLFlds, "ein Feld für Kennwort",
         "Auswahl eines Feldes in"++PLTheme.AsString)
    aKW= PLFTab.ReturnValue(aKWFld, aDtSIdx)
  elseif (Not Frg1) then
    aKW=MsgBox.Input("fuer das neue Polygon",
      "Eingabe eines Kennwortes","z.B. GrPg_Mtlo1")
  end
end
theKW=MsgBox.Input("fuer das neue Polygon",
       "Eingabe eines Kennwortes", aKW)
theBZ= MsgBox.Input("fuer das neue Polygon"+NL+
      "z. B. linksrheinische Untere Mittelterrasse oben",
      "Eingabe einer Bezeichnung", "lUMTo")

'Auswahl eines Themas zur Speicherung der Punkte
ListofAW={"ein neues Thema", "ein vorhandenes Thema"}
AW11=MsgBox.ChoiceAsString(ListofAW, "zur Speicherung der Punkte",
      "Auswahl eines Punkt-Themas")

if (AW11 = "ein neues Thema") then
  'Ein Feature-Shape-File für Polygon wird hergestellt.
  WDStr=av.GetProject.GetWorkDir.AsString
  FnStr0="Pg"+ PLTheme.AsString
  fnStr=FileName.Make(WDStr).MakeTmp(FnStr0,"shp")
  'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(FnStr0,"shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Polygon)")
  if (fName =nil) then exit end
  fName.SetExtension("shp")
  PgFTab =FTab.MakeNew(fName, Polygon)
  PgShpFld = PgFTab.FindField("shape")
  PgIdFld =Field.Make("ID", #Field_Short, 4, 0)
  PgKWFld=Field.Make("Kennwort", #Field_Char, 20, 0)
  PgBZFld=Field.Make("Beschreibung", #Field_Char, 60, 0)
  ListofPgFlds={ PgIdFld, PgBZFld, PgKWFld}
  PgFTab.AddFields(ListofPgFlds)
  recNr=-1

  PgTheme = FTheme.Make(PgFTab)
  theView.AddTheme(PgTheme)

elseif (AW11 = "ein vorhandenes Thema") then
  PgTheme =MsgBox.ChoiceAsString(ListofPgFThm, "Polygon-Thema für Grenze",
                  "Auswahl eines Themas")
  PgFTab=PgTheme.GetFTab
  PgShpFld=PgFTab.FindField("Shape")
  PgIdFld=PgFTab.FindField("Id")
  PgKWFld= PgFTab.FindField("Kennwort")
  PgBZFld= PgFTab.FindField("Beschreibung")

  'Feststellung der Anzahl der Datensätze im Thema
```

```
  AnzPgRec=0
  for each arec in PgFTab
    AnzPgRec = AnzPgRec +1
  end
  IdxPgRec= AnzPgRec -1

  'Bestimmung von Index des Datensatzes, um die Punkte zu speichern
  if (AnzPgRec = 0) then
   recNr=-1
  elseif (AnzPgRec > 0) then
   ListofAW22={"am Ende der vorhandenen Datensätze",
              "am bestimmten Stelle"}
   AW22=MsgBox.ChoiceAsString(ListofAW22,
      "zur Speicherung des neuen Polygons",
      "Auswahl einer Stelle in Datensätzen")
   if (AW22 = "am Ende der vorhandenen Datensätze") then
      recNr= IdxPgRec 'am Ende der vorhandenen Datensätze
   elseif (AW22 = "am bestimmten Stelle") then
     aDtSIdxStr= aDtSIdx.AsString
     nIdxStr=MsgBox.Input("zur Speicherung des Polygons",
          "Index-Nummer der Stelle",aDtSIdxStr)
     nIdxNr=nIdxStr.AsNumber
     recNr= nIdxNr-1
     qt2=MsgBox.YesNo("Ist die Stellennummer der Datensätze sicher?",
          "Feststellung der Nummer der Stelle", true)
     if (Not qt2) then
       MsgBox.Error("Die Stellennummer der Datensätze ist unsicher!"
          +NL+"Deshalb wird das Programm jetzt abgebrochen!", "")
       exit
     end
   end
  end
end

recNr=recNr+1
PgFTab.SetEditable(false)
PgFTab.SetEditable(true)
if (AW11 = "ein neues Thema") then
  PgFTab.AddRecord
elseif (AW11 = "ein vorhandenes Thema") then
  if (AW22 = "am Ende der vorhandenen Datensätze") then
    PgFTab.AddRecord
  end
end

PgFTab.SetValue(PgShpFld, recNr, thePolyg)
PgFTab.SetValue(PgIdFld, recNr, recNr)
PgFTab.SetValue(PgKWFld, recNr, theKW)
PgFTab.SetValue(PgBZFld, recNr, theBZ)
PgFTab.SetEditable(false)

PgTheme.UpdateLegend




'pgaustkt.ave
'Ein Polygon-Thema wird aus einem Punkt-Thema hergestellt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.
```

```
theProject=av.GetProject
theView=av.GetActiveDoc    'ein aktives View
ListofThms=theView.GetThemes
ListofFThms = {}

for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofFThms.Add(aT)
      end
   end
end

av.ShowMsg("Eingabe eines Punkt-Themas und"
     ++"Feststellung der Anzahl der Punkte in dem Thema ...")
PtTheme=MsgBox.ChoiceAsString(ListofFThms,
      "um die Punkte in Polygon umzuwandeln.", "Auswahl eines Punkt-Themas")
PtFTab=PtTheme.GetFTab
PtShpFld=PtFTab.FindField("Shape")
AnzPt=0
for each Pt in PtFTab
   AnzPt=AnzPt+1
end
AnzPtIdx=AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"++PtTheme.AsString)

'Die Rheihenfolge der Punkte wird kontrolliert.
Rheih={"Uhrzeigersinn", "Gegenuhrzeigersinn", "noch nicht kontrolliert"}
AW1=MsgBox.ChoiceAsString(Rheih, "Die Rheihenfolge der Punkte:",
     "Kontrolle der Punkte")

'Die Punkte werden in Polygon umgewandelt
ListofPt={}
if (AW1 = "Uhrzeigersinn") then
  for each aPt in 0..AnzPtIdx
     theShp=PtFTab.ReturnValue(PtShpFld, aPt)
     ListofPt.Add(theShp)
  end
elseif (AW1 = "Gegenuhrzeigersinn") then
  for each aPt in 0..AnzPtIdx
     neuIdx=AnzPtIdx-aPt
     theShp=PtFTab.ReturnValue(PtShpFld, neuIdx)
     ListofPt.Add(theShp)
  end
elseif (AW1 = "noch nicht kontrolliert") then
  MsgBox.Error("Die Rheihenfolge der Punkte im Theme"++PtTheme.AsString+NL+
            "muss noch festgetellt werden!"+NL+NL+
            "Das Programm wird jetzt abgebrochen!", "")
  exit
end
Fr1=MsgBox.YesNo("Soll noch einige zusätzliche Punkte eingefügt werden?", "Vertex", true)
if (Fr1) then
  MsgBox.Report("Die Rheihenfolge der Punkte für Polygon"+NL+
             "muss im Uhrzeigersinn stehen."+NL+NL+
             "Die zusätzlichen Punkte werden am Ende der Punkte"+NL+
             "auch im Uhrzeigersinn eingefügt.",
           "Hinweise zur Herstellung eines Polygons")
  ZSZahlStr=MsgBox.Input("Die Anzahl der zusätzlichen Punkte",
                    "Eingabe der zusätzlichen Punkte", "2")
  ZSZahl=ZSZahlStr.AsNumber.SetFormat("").SetFormat("d")
  ZSZIdx=(ZSZahl-1).SetFormat("").SetFormat("d")
```

```
  for each aZSZ in 0..ZSZIdx
    ZZ=(aZSZ+1).SetFormat("").SetFormat("d")
    strzsx1=MsgBox.Input("RW des"++ZZ.AsString+". Punktes von"++ZSZahl.AsString,
                "Koordinaten eines Zusatzpunktes", "2563000.00")
    strzsy1=MsgBox.Input("HW des"++ZZ.AsString+". Punktes von"++ZSZahl.AsString,
                "Koordinaten eines Zusatzpunktes", "5637000.00")
    zsx1=strzsx1.AsNumber
    zsy1=strzsy1.AsNumber
    ListofPt.Add(zsx1@zsy1)
  end
end
ListofListofPt={}
ListofListofPt.Add(ListofPt)
thePolyg=Polygon.Make(ListofListofPt)
MsgBox.report(thePolyg.AsString, "Das neue Polygon")

av.ShowMsg("Speicherung des hergestellten Polygons ...")
'Ein Feature-Shape-File eine Polygons wird hergestellt.
theWDStr = theProject.GetWorkDir.AsString
fnStr=FileName.Make(theWDStr).MakeTmp("Grpgmls1","shp")
fnGrMt=FileDialog.Put(fnStr, "*.shp", "Output 2D shape File (Polygon)")
if (fnGrMt=nil) then exit end
fnGrMt.SetExtension("shp")
GrMtFTab=FTab.MakeNew(fnGrMt, Polygon)
ShapeFld=GrMtFTab.FindField("shape")
IDFld=Field.Make("ID", #Field_Short, 2, 0)
ListofFlds1={IDFld}
GrMtFTab.AddFields(ListofFlds1)

GrMtFTab.SetEditable(false)
GrMtFTab.SetEditable(true)
recNr=0
GrMtFTab.AddRecord
GrMtFTab.SetValue(ShapeFld, recNr, thePolyg)
GrMtFTab.SetValue(IDFld, recNr, recNr)
GrMtFTab.SetEditable(false)

thmNew=FTheme.Make(GrMtFTab)
theView.AddTheme(thmNew)




'pgin2pl1.ave
'Das Polygon in einem Karten-View wird in zwei Polyline zerlegt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("pgin2pl1")
myScript.SetNumberFormat( "d.dd") ' script default

ListofThemes=theView.GetThemes
ListofPgFThm = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(Polygon)) then
       ListofPgFThm.Add(aT)
     end
```

```
   end
end
PgTheme=MsgBox.ChoiceAsString(ListofPgFThm, "ein Polygon-Thema zur Zerlegung",
      "Auswahl eines Themas (Input) im View"++theView.AsString)
PgFTab=PgTheme.GetFTab
ShpFld=PgFTab.FindField("Shape")

av.ShowMsg("Feststellung der Koordinaten des Polygons in"
        ++PgTheme.AsString++"...")

theShape=PgFTab.ReturnValue(ShpFld, 0)
MsgBox.Report(theShape.AsString, "Die ganzen Punkte des Polygons"
         ++PgTheme.AsString)
ListofShapes={}
ListofShapes.Add({theShape})

'Erkennung der Koordinaten auf dem Polygon

ListofgPgPt={}
Ng=0
maxHW=5618000
minHW=5642000
Stelle=-1
minStelle=0
maxStelle=0

'2D-Polygon wird in Liste der Koordinaten umgewandelt.
if (ListofShapes <> 0) then
  for each aShp in ListofShapes
     aListofListofPts=aShp.Get(0).AsList
     for each aList in aListofListofPts
        for each myPt in aList
           Stelle=Stelle+1
           myx=myPt.Getx
           myy=myPt.Gety
           ListofgPgPt.Add(myx@myy)
           if (myy < minHW) then
             minHW=myy
             minStelle=Stelle
           elseif (myy > maxHW) then
             maxHW=myy
             maxStelle=Stelle
           end
        end
     end
  end
end

AnzgPt=ListofgPgPt.Count
AnzgPl=AnzgPt-1
gPtIdx=AnzgPt-1-1

AnfHW=ListofgPgPt.Get(0).Gety
EndHW=ListofgPgPt.Get(gPtIdx).Gety  'Der vorletzte Punkt des Polygons

MsgBox.Report("Anzahl der Punkte im Polygon:"++AnzgPt.AsString+NL+
           "Anzahl der Punkte in Polyline (gesamt)"++AnzgPl.AsString+NL+NL+
           "Hochwert des ersten Punktes im Polygon:"++AnfHW.AsString+NL+
           "Hochwert des vorletzten Punktes im Polygon"++EndHW.AsString+NL+NL+
           "Der kleinste Hochwert im Polygon"++minHW.AsString+NL+
           "Index des kleinsten Hochwertes im Polygon"++minStelle.AsString+NL+
           "Der größte Hochwert im Polygon"++maxHW.AsString+NL+
```

```
            "Index des größten Hochwertes im Polygon"++maxStelle.AsString,
            "Information")

ListoflPgPt={}
ListofrPgPt={}

if ((AnfHW = maxHW) or (EndHW = maxHW)) then
   for each Pt in 0..gPtIdx
      aPt=ListofgPgPt.Get(Pt)
      aPtHWSt=Pt
      if (aPtHWSt < minStelle) then
        ListofrPgPt.Add(aPt)
      elseif (aPtHWSt = minStelle) then
        ListofrPgPt.Add(aPt)
        ListoflPgPt.Add(aPt)
      elseif (aPtHWSt > minStelle) then
        ListoflPgPt.Add(aPt)
      end
   end
elseif ((AnfHW = minHW) or (EndHW = minHW)) then
   for each Pt in 0..gPtIdx
      aPt=ListofgPgPt.Get(Pt)
      aPtHWSt=Pt
      if (aPtHWSt < maxStelle) then
        ListoflPgPt.Add(aPt)
      elseif (aPtHWSt = maxStelle) then
        ListoflPgPt.Add(aPt)
        ListofrPgPt.Add(aPt)
      elseif (aPtHWSt > maxStelle) then
        ListofrPgPt.Add(aPt)
      end
   end
elseif (((AnfHW > minHW) and (AnfHW < maxHW))
   and ((EndHW > minHW) and (EndHW < maxHW)))then
  if (EndHW > AnfHW) then
   for each Pt in 0..gPtIdx
      aPt=ListofgPgPt.Get(Pt)
      aHWSt=Pt
      if (aHWSt < minStelle) then
        ListofrPgPt.Add(aPt)
      elseif (aHWSt = minStelle) then
        ListofrPgPt.Add(aPt)
        ListoflPgPt.Add(aPt)
      elseif ((aHWSt > minStelle) and (aHWSt < maxStelle)) then
        ListoflPgPt.Add(aPt)
      elseif (aHWSt = maxStelle) then
        ListoflPgPt.Add(aPt)
        ListofrPgPt.Add(aPt)
      elseif (aHWSt > maxStelle) then
        ListofrPgPt.Add(aPt)
      end
   end
  elseif (AnfHW > EndHW) then
   for each Pt in 0..gPtIdx
      aPt=ListofgPgPt.Get(Pt)
      aHWSt=Pt
      if (aHWSt < maxStelle) then
        ListoflPgPt.Add(aPt)
      elseif (aHWSt = maxStelle) then
        ListoflPgPt.Add(aPt)
        ListofrPgPt.Add(aPt)
      elseif ((aHWSt > maxStelle) and (aHWSt < minStelle)) then
```

```
        ListofrPgPt.Add(aPt)
      elseif (aHWSt = minStelle) then
        ListofrPgPt.Add(aPt)
        ListoflPgPt.Add(aPt)
      elseif (aHWSt > minStelle) then
        ListoflPgPt.Add(aPt)
      end
    end
  end
end

AnzrPg=ListofrPgPt.Count
rPgIdx=AnzrPg-1

LtofLtofrPgPt={}
LtofLtofrPgPt.Add(ListofrPgPt)
RPolyL=PolyLine.Make(LtofLtofrPgPt)

MsgBox.Report("Anzahl der Punkte der rechten Seite des Polygons"
          ++AnzrPg.AsString+NL+RPolyL.AsString,
          "Die Punkte der rechten Seite des Polygons")

AnzlPg=ListoflPgPt.Count
lPgIdx=AnzlPg-1

LtofLtoflPgPt={}
LtofLtoflPgPt.Add(ListoflPgPt)
LPolyL=PolyLine.Make(LtofLtoflPgPt)

MsgBox.Report("Anzahl der Punkte der linken Seite des Polygons"
          ++AnzlPg.AsString+NL+LPolyL.AsString,
          "Die Punkte der linken Seite des Polygons")

rHWmax=5618000
rHWmin=5645000
rHWmaxSt=0
rHWminSt=0
for each aPt in 0..rPgIdx
   rPtHW=ListofrPgPt.Get(aPt).Gety
   rPtSt=aPt
   if (rPtHW < rHWmin) then
     rHWmin=rPtHW
     rHWminSt=rPtSt
   end
   if (rPtHW > rHWmax) then
     rHWmax=rPtHW
     rHWmaxSt=rPtSt
   end
end

AnfrPgHW=ListofrPgPt.Get(0).Gety
EndrPgHW=ListofrPgPt.Get(rPgIdx).Gety
MsgBox.Report("Anzahl der Punkte:"++AnzrPg.AsString++NL+NL+
          "Hochwert des ersten Punktes:"++AnfrPgHW.AsString+NL+
          "Hochwert des letzten Punktes:"++EndrPgHW.AsString+NL+NL+
          "Der kleinste Hochwert:"++rHWmin.AsString+NL+
          "Index des kleinsten Hochwertes:"++rHWminSt.AsString+NL+NL+
          "Der größte Hochwert:"++rHWmax.AsString+NL+
          "Index des größten Hochwertes:"++rHWmaxSt.AsString,
          "Information auf der rechten PolyLine")
```

```
ListofrPLPt={}
if ((AnfrPgHW = rHWmax) and (EndrPgHW = rHWmin)) then
  for each aPtIdx in 0..rPgIdx
     myPtIdx=rPgIdx-aPtIdx
     rPLPt=ListofrPgPt.Get(myPtIdx)
     ListofrPLPt.Add(rPLPt)
  end
elseif ((AnfrPgHW = rHWmin) and (EndrPgHW = rHWmax)) then
  for each aPtIdx in 0..rPgIdx
     rPLPt=ListofrPgPt.Get(aPtIdx)
     ListofrPLPt.Add(rPLPt)
  end
elseif (((AnfrPgHW > rHWmin) and (AnfrPgHW < rHWmax)) and
       ((EndrPgHW > rHWmin) and (EndrPgHW < rHWmax))) then
 if (EndrPgHW > AnfrPgHW) then
  for each aPtIdx in 0..rPgIdx
     einPtHWSt=aPtIdx
     if ((einPtHWSt < rHWminSt) or (einPtHWSt = rHWminSt)) then
       myPtIdx=rHWminSt-aPtIdx
     elseif (einPtHWSt > rHWminSt) then
       myPtIdx=rPgIdx+((rHWminSt+1)-aPtIdx)
     end
     if ((myPtIdx < rPgIdx) or (myPtIdx = rPgIdx)) then
       rPLPt=ListofrPgPt.Get(myPtIdx)
       ListofrPLPt.Add(rPLPt)
     end
  end
 elseif (AnfrPgHW > EndrPgHW) then
  AbstIdx=EndrPgHW-rHWminSt
  for each aPtIdx in 0..rPgIdx
     einPtHWSt=aPtIdx
     if ((einPtHWSt < AbstIdx) or (einPtHWSt = AbstIdx)) then
       myPtIdx=aPtIdx+rHWminSt
     elseif (einPtHWSt > AbstIdx) then
       myPtIdx=aPtIdx-(AbstIdx+1)
     end
     rPLPt=ListofrPgPt.Get(myPtIdx)
     ListofrPLPt.Add(rPLPt)
   end
 end
end

AnzlPg=ListoflPgPt.Count
lPgIdx=AnzlPg-1
lHWmax=5618000
lHWmin=5645000
lHWmaxSt=0
lHWminSt=0
for each aPt in 0..lPgIdx
   lPtHW=ListoflPgPt.Get(aPt).Gety
   lPtSt=aPt
   if (lPtHW < lHWmin) then
     lHWmin=lPtHW
     lHWminSt=lPtSt
   end
   if (lPtHW > lHWmax) then
     lHWmax=lPtHW
     lHWmaxSt=lPtSt
   end
end
AnflPgHW=ListoflPgPt.Get(0).Gety
EndlPgHW=ListoflPgPt.Get(lPgIdx).Gety
```

```
MsgBox.Report("Anzahl der Punkte:"++AnzlPg.AsString++NL+NL+
            "Hochwert des ersten Punktes:"++AnflPgHW.AsString+NL+
            "Hochwert des letzten Punktes:"++EndlPgHW.AsString+NL+NL+
            "Der kleinste Hochwert:"++lHWmin.AsString+NL+
            "Index des kleinsten Hochwertes:"++lHWminSt.AsString+NL+NL+
            "Der größte Hochwert:"++lHWmax.AsString+NL+
            "Index des größten Hochwertes:"++lHWmaxSt.AsString,
            "Information auf der linken PolyLine")


ListoflPLPt={}
if ((AnflPgHW = lHWmax) and (EndlPgHW = lHWmin)) then
  for each aPtIdx in 0..lPgIdx
     myPtIdx=lPgIdx-aPtIdx
     lPLPt=ListoflPgPt.Get(myPtIdx)
     ListoflPLPt.Add(lPLPt)
  end
elseif ((AnflPgHW = lHWmin) and (EndlPgHW = lHWmax)) then
  for each aPtIdx in 0..lPgIdx
     lPLPt=ListoflPgPt.Get(aPtIdx)
     ListoflPLPt.Add(lPLPt)
  end
elseif (((AnflPgHW > lHWmin) and (AnflPgHW < lHWmax)) and
      ((EndlPgHW > lHWmin) and (EndlPgHW < lHWmax))) then
 if (EndlPgHW > AnflPgHW) then
  for each aPtIdx in 0..lPgIdx
     einPtHWSt=aPtIdx
     if ((einPtHWSt < lHWminSt) or (einPtHWSt = lHWminSt)) then
       myPtIdx=lHWminSt-aPtIdx
     elseif (einPtHWSt > lHWminSt) then
       myPtIdx=lPgIdx+((lHWminSt+1)-aPtIdx)
     end
     if ((myPtIdx < lPgIdx) or (myPtIdx = lPgIdx)) then
       lPLPt=ListoflPgPt.Get(myPtIdx)
       ListoflPLPt.Add(lPLPt)
     end
  end
 elseif (AnflPgHW > EndlPgHW) then
  AbstIdx=lPgIdx-lHWminSt
  for each aPtIdx in 0..lPgIdx
     einPtHWSt=aPtIdx
     if ((einPtHWSt < AbstIdx) or (einPtHWSt = AbstIdx)) then
       myPtIdx=aPtIdx+lHWminSt
     elseif (einPtHWSt > AbstIdx) then
       myPtIdx=aPtIdx-(AbstIdx+1)
     end
     lPLPt=ListoflPgPt.Get(myPtIdx)
     ListoflPLPt.Add(lPLPt)
   end
 end
end

ListofListofrPLPt={}
ListofListofrPLPt.Add(ListofrPLPt)
therPolyL=PolyLine.Make(ListofListofrPLPt)
MsgBox.Report(therPolyL.AsString, "Die Punkte der rechten PolyLine in Output-Thema")
therBz="rechte Seite (Polyline) des Polygons"++ PgTheme.AsString
therKW="R-"+ PgTheme.AsString

ListofListoflPLPt={}
ListofListoflPLPt.Add(ListoflPLPt)
```

```
thelPolyL=PolyLine.Make(ListofListoflPLPt)
MsgBox.Report(thelPolyL.AsString, "Die Punkte der linken PolyLine in Output-Thema")
thelBz="linke Seite (Polyline) des Polygons"++ PgTheme.AsString
thelKW="L-"+ PgTheme.AsString

'Auswahl eines Themas zur Speicherung der neuen PolyLines
ListofAW={"ein neues Thema", "ein vorhandenes Thema"}
AW11=MsgBox.ChoiceAsString(ListofAW, "zur Speicherung des neuen PolyLines",
      "Auswahl eines PL-Themas")

rNeuerDatensatz="ja"
lNeuerDatensatz="ja"
if (AW11 = "ein neues Thema") then
  'Ein Feature-Shape-File für PolyLine wird hergestellt.
  WDStr=av.GetProject.GetWorkDir.AsString
  fnStr=FileName.Make(WDStr).MakeTmp("Plrrin11","shp")
  'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("Plrrin11","shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (rechte PolyLine)")
  if (fName =nil) then exit end
  fName.SetExtension("shp")
  rPLFTab=FTab.MakeNew(fName, PolyLine)
  rPLShpFld= rPLFTab.FindField("shape")
  rPLIDFld =Field.Make("ID", #Field_Short, 3, 0)
  rPLBZFld=Field.Make("Beschreibung", #Field_Char, 60, 0)
  rPLKWFld=Field.Make("Kennwort", #Field_Char, 20, 0)

  ListofrPLFlds={rPLIDFld, rPLBZFld, rPLKWFld}
  rPLFTab.AddFields(ListofrPLFlds)
  rRecNr=0

  WDStr=av.GetProject.GetWorkDir.AsString
  lfnStr=FileName.Make(WDStr).MakeTmp("Pllrin11","shp")
  'lfnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("Pllrin11","shp")
  lfName=FileDialog.Put(lfnStr, "*.shp", "Output shape File (linke PolyLine)")
  if (lfName =nil) then exit end
  lfName.SetExtension("shp")
  lPLFTab=FTab.MakeNew(lfName, PolyLine)
  lPLShpFld= lPLFTab.FindField("shape")
  lPLIDFld =Field.Make("ID", #Field_Short, 3, 0)
  lPLBZFld=Field.Make("Beschreibung", #Field_Char, 60, 0)
  lPLKWFld=Field.Make("Kennwort", #Field_Char, 20, 0)

  ListoflPLFlds={lPLIDFld, lPLBZFld, lPLKWFld}
  lPLFTab.AddFields(ListoflPLFlds)
  lRecNr=0

  rPLNewTh=FTheme.Make(rPLFTab)
  theView.AddTheme(rPLNewTh)
  therTheme= rPLNewTh

  lPLNewTh=FTheme.Make(lPLFTab)
  theView.AddTheme(lPLNewTh)
  thelTheme= lPLNewTh

elseif (AW11 = "ein vorhandenes Thema") then
  'Ein schon vorhandenes Thema wird ausgewählt, um eine neue PolyLine zu speichern.
  'Auswahl eines PolyLine-Themas
  ListofPLFThm = {}
  for each aT in ListofThemes
     if (aT.Is(FTheme)) then
       aFTab = aT.GetFTab
       if (aFTab.GetShapeClass.IsSubclassOf(PolyLine)) then
```

```
          ListofPLFThm.Add(aT)
        end
      end
  end
  rPLTh=MsgBox.ChoiceAsString(ListofPLFThm,
                      "R-PolyLine-Thema für die Rinne"++PgTheme.AsString,
                      "Auswahl eines Themas (Output)")
  rPLFTab=rPLTh.GetFTab
  rPLShpFld=rPLFTab.FindField("Shape")
  rPLIDFld= rPLFTab.FindField("ID")
  rPLBZFld= rPLFTab.FindField("Beschreibung")
  rPLKWFld= rPLFTab.FindField("Kennwort")

  'Feststellung der Anzahl der Records im Thema
  AnzrPLRec=0
  for each arec in rPLFtab
     AnzrPLRec=AnzrPLRec+1
  end
  IdxrPLRec=AnzrPLRec-1
  'Bestimmung des Datensatzes, um das neue Polygon zu speichern
  IdxrSt= IdxrPLRec+1      'am Ende der vorhandenen Datensätze
  for each arec in 0.. IdxrPLRec
    arKW= rPLFtab.ReturnValue(rPLKWFld, arec)
    If (arKW = therKW) then
      IdxrSt=arec            'Ein vorhandener Datensatz wird korrigiert.
      rNeuerDatensatz="nein"
    end
  end
  rRecNr= IdxrSt

  lPLTh=MsgBox.ChoiceAsString(ListofPLFThm,
                      "L-PolyLine-Thema für die Rinne"++PgTheme.AsString,
                      "Auswahl eines Themas (Output)")
  lPLFTab=lPLTh.GetFTab
  lPLShpFld=lPLFTab.FindField("Shape")
  lPLIDFld= lPLFTab.FindField("ID")
  lPLBZFld= lPLFTab.FindField("Beschreibung")
  lPLKWFld= lPLFTab.FindField("Kennwort")

  'Feststellung der Anzahl der Records im Thema
  AnzlPLRec=0
  for each arec in lPLFtab
     AnzlPLRec=AnzlPLRec+1
  end
  IdxlPLRec=AnzlPLRec-1
  'Bestimmung des Datensatzes, um das neue Polygon zu speichern
  IdxlSt= IdxlPLRec+1      'am Ende der vorhandenen Datensätze
  for each arec in 0.. IdxlPLRec
     alKW= lPLFtab.ReturnValue(lPLKWFld, arec)
     If (alKW = thelKW) then
       IdxlSt=arec            'Ein vorhandener Datensatz wird korrigiert.
       lNeuerDatensatz="nein"
     end
  end
  lRecNr= IdxlSt
  therTheme= rPLTh
  thelTheme= lPLTh

end

rPLFTab.SetEditable(False)
rPLFTab.SetEditable(true)
```

```
If (rNeuerDatensatz = "ja") then
  rPLFTab.AddRecord
end
rPLFTab.SetValue(rPLShpFld, rRecNr, therPolyL)
rPLFTab.SetValue(rPLIDFld, rRecNr, rRecNr)
rPLFTab.SetValue(rPLBZFld, rRecNr, therBz)
rPLFTab.SetValue(rPLKWFld, rRecNr, therKW)
rPLFTab.SetEditable(False)

lPLFTab.SetEditable(False)
lPLFTab.SetEditable(true)
If (lNeuerDatensatz = "ja") then
  lPLFTab.AddRecord
end
lPLFTab.SetValue(lPLShpFld, lRecNr, thelPolyL)
lPLFTab.SetValue(lPLIDFld, lRecNr, lRecNr)
lPLFTab.SetValue(lPLBZFld, lRecNr, thelBz)
lPLFTab.SetValue(lPLKWFld, lRecNr, thelKW)
lPLFTab.SetEditable(False)

therTheme.UpdateLegend

thelTheme.UpdateLegend




'pglaustm.ave
'Ein Polyline- oder Polygon-Thema wird durch Punkte in einem aktiven View
'hergestellt oder korrigiert, die sich in einer Punkt-Datei
'in der Nähe der Maus-Klicken befinden.
'Dieses Script wird als ein Werkzeug (Tool) im aktiven View benutzt.

theProject = av.GetProject
theView = av.GetActiveDoc   'Ein aktives View

'mehrmalige Klicken auf das Bildschirm im aktiven View zur Bestimmung der Stelle
aEingPolyL = theView.GetDisplay.ReturnUserPolyLine

ListofListofEingPt = aEingPolyL.AsList
ListofEingPt = ListofListofEingPt.Get(0)
AnzM = ListofEingPt.Count
AnzMIdx = AnzM - 1
ListofMPt = {}
ListofMCPg = {}
for each mPt in 0..AnzMIdx
  aMPt = ListofEingPt.Get(mPt)
  ListofMPt.Add(aMPt)
  aCircle = Circle.Make(aMPt, 50)
  aMCPg = aCircle.AsPolygon
  ListofMCPg.Add(aMCPg)
end

ListofAufgaben = {"Herstellung einer Polyline", "Korrektur einer Polyline",
            "Herstellung eines Polygons", "Korrektur eines Poygons"}
aAufg = MsgBox.ListAsString(ListofAufgaben,
     "mit den Punkten an Maus-Klicken","Auswahl einer Aufgabe")

ListofThemes=theView.GetThemes
ListofPLThms = {}
```

```
ListofPgThms = {}
ListofPtThms = {}

av.ShowMsg("Auswahl eines Themas ...")
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     aShpClassStr = aFTab.GetShapeClass.GetClassName
     if (aShpClassStr = "Polyline") then
        ListofPLThms.Add(aT)
     elseif (aShpClassStr = "Polygon") then
        ListofPgThms.Add(aT)
     elseif (aShpClassStr = "Point") then
       ListofPtThms.Add(aT)
     end
   end
end

'Auswahl eines Themas (Polyline oder Polygon) im View zur Korrektur
if ((aAufg = "Korrektur einer Polyline") or
   (aAufg = "Korrektur eines Poygons")) then
  aAktThm = theView.GetActiveThemes.Get(0)

  Frg11=MsgBox.YesNo("Ist das aktive Thema"++aAktThm.AsString
     ++"richtig?", "Kontrolle des aktiven Themas", true)

  if (Frg11) then
    PglThm = aAktThm
  elseif (Not Frg11) then
    'Auswahl eines Themas
    if (aAufg = "Korrektur einer Polyline") then
      PglThm = MsgBox.ChoiceAsString(ListofPLThms,
        "Name eines Polyline-Themas zur Korrektur:",
        "Auswahl eines Themas im View"++theView.AsString)
    elseif (aAufg = "Korrektur eines Poygons") then
      PglThm = MsgBox.ChoiceAsString(ListofPgThms,
        "Name eines Polygon-Themas zur Korrektur:",
        "Auswahl eines Themas im View"++theView.AsString)
    end

    if (PglThm = nil) then
      MsgBox.Error("Der Name der Datei fehlt!"+NL+
                "Das Programm wird abgebrochen!", "")
      exit
    end
  end

  'MsgBox.Info(PglThm.AsString, "Die Polyline oder das Polygon")

  PglFTab = PglThm.GetFTab
  PglTStr = PglThm.AsString

  PglShpClassStr = PglFTab.GetShapeClass.GetClassName

  ListofPglFlds = PglFTab.GetFields
  AnzPglFld = ListofPglFlds.Count
  IdxPglFld = AnzPglFld - 1
  PglShpFld = ListofPglFlds.Get(0)
  PglIDFld = PglFTab.FindField("ID")

  AnzDs = 0          'Anzahl der Datensätze
  for each rec in PglFTab
```

```
  AnzDs = AnzDs + 1
end
IdxDs = AnzDs - 1

'Feststellung des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxPglFld
  aFld = ListofPglFlds.Get(i)
  aFldStr=aFld.GetName
  if (aFldStr <> "Shape") then
       FldStr = FldStr + aFldStr+"; "
       aValue = PglFTab.ReturnValue(aFld, 0)
       DtStr=DtStr+aValue.AsString+"; "
  end
end

MsgBox.Report("vom Thema"++PglTStr+","+NL+
      "um zum Teil zu korrigieren"+NL+NL+
      "Die Namen der Felder ohne Shape-Felder"
       +NL+FldStr+NL+NL+
      "Der erste Datensatz:"+NL+DtStr,
      "Information eines Polygons oder einer Polyline ")

PglKWFld = MsgBox.ListAsString(ListofPglFlds,
  "des Themas"++PglTStr+NL+
  "für Bezeichnung der Datensäze und"+NL+
  "für Klassifizierung der Legende zur Korrektur",
  "Auswahl eines Feldes")

PglKWFldStr = PglKWFld.AsString

PglSchtFld = MsgBox.ListAsString(ListofPglFlds,
  "des Themas"++PglTStr+NL+
  "für Beschrteibung der Datensäze",
  "Auswahl eines Feldes")

aLegend = PglThm.GetLegend
aorgLTyp = aLegend.GetLegendType
'MsgBox.Info(aorgLTyp.AsString, "eigentliche Legende")

aListofSymbol = aLegend.GetSymbols
AnzSymb = aListofSymbol.Count
AnzSymbIdx = AnzSymb-1

ListoforgColor = {}
for each symb in 0..AnzSymbIdx
    aorgColor = aListofSymbol.Get(symb).GetColor
    ListoforgColor.Add(aorgColor)
end

aLegend.SetLegendType(#Legend_Type_Unique)
aLegend.Unique(PglThm, PglKWFldStr)
ListofKlasse = aLegend.GetClassifications
AnzKlasse = ListofKlasse.Count
IdxKlasse = AnzKlasse-1
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")

ListofKlBez = {}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   ListofKlBez.Add(theKlasseLb)
```

```
end

theKlBz=MsgBox.ListAsString(ListofKlBez,
  "im"++PglTStr++"zur Auswahl der Polyline"
  +NL+"oder des Polygons zur Korrektur",
  "Auswahl einer Bezeichnung der Datensätze")
if (theKlBz = nil) then
  MsgBox.Error("Die Bezeichnung der Polyline, des Polygons oder"
    +NL+"die Polyline oder das Polygon mit der Bezeichnung fehlt!"
    +NL+"Das Programm wird abgebrochen!", "")
 exit
end

'Selection der Datensätze im Thema mit den Maus-Klicken
ListofSBz = {}
ListofIdxS = {}

if (PglThm.CanSelect) then
  'MsgBox.Info("Das Thema"++PglTStr++"ist selectierbar.","CanSelect?")
  for each rec in 0..AnzMIdx
    aMCPg = ListofMCPg.Get(rec)
    if (rec = 0) then
      PglThm.SelectByPolygon(aMCPg, #VTAB_SELTYPE_NEW)
    elseif (rec > 0) then
      PglThm.SelectByPolygon(aMCPg, #VTAB_SELTYPE_OR)
    end
  end
  'die selektierten Datensätze
  AnzS = 0
  for each rec in PglFTab.GetSelection
    'MsgBox.Info("von"++PglTStr++":  "++rec.AsString,
    '    "Index-Nummer der selektierten Datensätze")
    aSBz = PglFTab.ReturnValue(PglKWFld, rec)
    ListofSBz.Add(aSBz)
    AnzS = AnzS + 1
    ListofIdxS.Add(rec.AsString)
  end
  'MsgBox.Info(AnzS.AsString, "Anzahl der selektierten Datensätze")
  'MsgBox.ListAsString(ListofIdxS, "Index-Nummer der selektierten"
  '                 ++"Datensätze"++"von"++PglTStr, "Information")

  'Anzahl der selektierten Datensätze
  AnzSDS = ListofSBz.Count
  if (AnzSDS = 0) then
    for each i in 0..IdxDs
      aBz = PglFTab.ReturnValue(PglKWFld, i)
      if (aBz = theKlBz) then
        ListofSBz.Add(aBz)
        ListofIdxS.Add(i.AsString)
      end
    end
  end
else
  for each i in 0..IdxDs
    aBz = PglFTab.ReturnValue(PglKWFld, i)
    if (aBz = theKlBz) then
      ListofSBz.Add(aBz)
      ListofIdxS.Add(i.AsString)
    end
  end
end
```

```
  aSBz = MsgBox.ListAsString(ListofSBz,
       "von"++PglTStr++","++
       "um die Polyline oder das Polygon zum Teil zu korrigieren",
       "Auswahl eines Datensatzes")
  aSBzIdx = ListofSBz.FindByValue(aSBz)
  aSRIdx = ListofIdxS.Get(aSBzIdx).AsNumber

  MsgBox.Info("von"++PglTStr++":  "++aSRIdx.AsString,
       "Index des ausgewählten Datensatzes")
  theShp = PglFTab.ReturnValue(PglShpFld, aSRIdx)
  'MsgBox.report("des Themas"++PglTStr+NL+
  '             "Index-Nummer:"++aSRIdx.AsString+NL+NL+
  '             theShp.AsString, "Selektiertes Shape")

  PglThm.ClearSelection

  av.ShowMsg("Feststellung der Koordinaten der Polyline"
         ++"oder des Polygons"++PglThm.AsString++"...")

  theProfilList1={}
  theProfilList1.Add({theShp})

  'Liste der Vertices
  ListofFLPt = {}

  if (theProfilList1 <> 0) then
    for each q in theProfilList1
         theLines=q.Get(0).AsList
         for each m in theLines
             for each ptx in m
                myx=ptx.Getx
                myy=ptx.Gety
                ListofFLPt.Add(myx@myy)
             end
         end
    end
  end
end

'Auswahl eines Punktthemas im View,
'um Punkte im Thema in der nächesten Entfernung
'von Maus-Klicken zu holen.

thePtTheme=MsgBox.Choice(ListofPtThms, "Name eines Punkt-Themas:"
         +NL+"um Punkte an Maus-Klicken zu übernehmen",
         "Auswahl eines Themas im View"++theView.AsString)
'MsgBox.Info(thePtTheme.AsString, "Der Name des Punkt-Themas")

PtFTab=thePtTheme.GetFTab
ListofPtFlds = PtFTab.GetFields
PtShpFld = ListofPtFlds.Get(0) 'Feldauswahl für evtl. Ereignisthema

'Feststellung der Anzahl der Punkte in dem Thema
AnzThPt=0
for each rec in PtFTab
   AnzThPt=AnzThPt+1
end
ThPtIdx=AnzThPt-1

av.ShowMsg("Bestimmung der Punkte in der Nähe der Mausklicken in"
      ++thePtTheme.AsString++"...")
```

```
av.ShowStopButton
ListofPgPT={}
MinAbst=22600.00
Ng=0
For each aMPt in 0..AnzMIdx
   Ng=Ng+1
   aPt = ListofMPt.Get(aMPt)
   axMPt=aPt.Getx
   ayMPt=aPt.Gety
   For each aThPt in 0..ThPtIdx
     thePtShape=PtFTab.ReturnValue(PtShpFld, aThPt)
     axThPt=thePtShape.Getx
     ayThPt=thePtShape.Gety
     xAbst=(axMPt-axThPt).Abs
     if (xAbst < 1000) then
       yAbst=(ayMPt-ayThPt)
       PtAbst=(((xAbst*xAbst)+(yAbst*yAbst)).Sqrt).Abs
       if (PtAbst < MinAbst) then
         MinAbst=PtAbst
         xMinThPt=axThPt
         yMinThPt=ayThPt
       end
     end
   end
   ListofPgPT.Add(xMinThPt@yMinThPt)
   MinAbst=22600.00
   'Show percentage complete with enabled stop button
   more=av.SetStatus(Ng/AnzM*100)
   if (not more) then
     break
   end
end

AnzPgPt = ListofPgPT.Count
IdxPgPt = AnzPgPt - 1
MsgBox.Info(AnzPgPt.AsString, "AnzPgPt")


if ((aAufg = "Korrektur einer Polyline") or
   (aAufg = "Korrektur eines Poygons")) then

  'Alle ausgewählten Punkte werden als ein Abschnitt
  'in Polyline oder Polygon eingesetzt.

  'Der erste, letzte und mittlere Punkt der ausgewälten Punkte
  erstPt = ListofPgPT.Get(0)
  letztPt = ListofPgPT.Get(IdxPgPt)

  ListofAbschPt = {erstPt, letztPt}

  'Anzahl der Stützpunkte der Polyline oder des Polygons
  PtAnz= ListofFLPt.Count
  PtAnzIdx= PtAnz-1

  'Suche nach einem Abschnitt auf Polyline oder Polygon
  'Bestimmung des Abschnittes als Index der Stützpunkte
  'mit dem kleinsten Abstand von den ausgewählten Punkten
  av.ShowMsg("Feststellung der Abschnitte der Polyline"
        ++"oder des Polygons ...")
  av.ShowStopButton

  ListofGrPt = {}
```

```
ListofListofGrIdx = {}

for each aM in 0..1   ' Anfang der Schleife für einen Abschnitt
  Ng = Ng + 1
  minDist=1000000
  minIdx=-1

  aMPt = ListofAbschPt.Get(aM)
  aMPtx = aMPt.Getx
  aMPty = aMPt.Gety

  for each i in 0..PtAnzIdx
    aPt = ListofFLPt.Get(i)
    xkrd= aPt.Getx
    ykrd= aPt.Gety
    xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
    yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
    xyAbst = ((xAbst  + yAbst).sqrt) .Abs
    if (xyAbst < minDist) then
      minDist = xyAbst
      minIdx = i  'Index des nähesten Punktes des Maus-Klickens
    end
  end

  'Bestimmung des Abschnittes des Shapes
  'als x-, y-Koordinaten

  if (minIdx = 0) then
    vIdx = 0     'ein Vertex auf der Linie vor dem Maus-Klicken
    nIdx = 1     'ein Vertex auf der Linie nach dem Maus-Klicken
  elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
    aPtv = ListofFLPt.Get((minIdx - 1))
    aPt0 = ListofFLPt.Get(minIdx)
    aPtn = ListofFLPt.Get((minIdx + 1))

    xkrdv = aPtv.Getx   'x-Koord. des letzten Punktes
    xkrd  = aPt0.Getx   'x-Koord. des nähesten Punktes
    xkrdn = aPtn.Getx   'x-Koord. des nächsten Punktes

    ykrdv = aPtv.Gety   'y-Koord. des letzten Punktes
    ykrd  = aPt0.Gety          'y-Koord. des nähesten Punktes
    ykrdn = aPtn.Gety   'y-Koord. des nächsten Punktes

    xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
    yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
    xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten Punkt vom nähesten Punkt

    xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
    yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
    xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten Punkt vom nähesten Punkte

    xML = (xkrd - aMPtx) * (xkrd - aMPtx)
    yML = (ykrd - aMPty) * (ykrd - aMPty)
    xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Maus-Klicken vom nähesten Punkt

    xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
    yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
    xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Maus-Klicken vom letzten Punkt

    xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
    yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
    xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Maus-Klicken vom nächsten Punkt
```

```
  vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken und dem
  vLpML = xyAv + xyML        'letzten Punkt

  nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken und dem
  nLpML = xyAn + xyML        'nächsten Punkt

  vM = (vLmML - xyMLv).Abs   'Differenz zwischen der theorischen und der
  vP = (vLpML - xyMLv).Abs   'tatsächlichen Länge im Bezug auf den letzten Punkt

  nM = (nLmML - xyMLn).Abs   'Differenz zwischen der theorischen und der
  nP = (nLpML - xyMLn).Abs   'tatsächlichen Länge im Bezug auf den nächsten Punkt

  ListofLg = {vM, vP, nM, nP}  'Suche nach der kleinsten Länge
  MinLg = 10000
  TheLgIdx = -1
  For each aLg in 0..3
     theLg = ListofLg.Get(aLg)
     if (theLg < MinLg) then
          MinLg = theLg
          TheLgIdx = aLg
     end
  end
  if (xyML < 10) then
    vIdx = minIdx - 1
    nIdx = minIdx + 1
  elseif (xyML >= 10) then
    if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
         vIdx = minIdx - 1     'ein Vertex auf der Linie vor dem Maus-Klicken
         nIdx= minIdx          'ein Vertex auf der Linie nach dem Maus-Klicken
    elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
         vIdx = minIdx         'ein Vertex auf der Linie vor dem Maus-Klicken
         nIdx = minIdx + 1     'ein Vertex auf der Linie nach dem Maus-Klicken
    else
         vIdx = minIdx         'ein Vertex auf der Linie vor dem Maus-Klicken
         nIdx = minIdx + 1     'ein Vertex auf der Linie nach dem Maus-Klicken
    end
  end
elseif (minIdx = PtAnzIdx)   then
   vIdx = minIdx - 1
   nIdx = minIdx
end

'Bestimmung der Koordinaten des Maus-Klickens auf dem Profilschnitt
vPt = ListofFLPt.Get(vIdx)
vPtx = vPt.Getx
vPty = vPt.Gety
nPt = ListofFLPt.Get(nIdx)
nPtx = nPt.Getx
nPty = nPt.Gety

if ((minDist = 0) or (minDist < 10)) then 'Der näheste Punkt liegt innerhalb
   mPt = ListofFLPt.Get(minIdx)
   Ptx = mPt.Getx    '10 m vom dem Maus-Klicken auf der Linie
   Pty = mPt.Gety
   if ( minIdx = 0) then
     vGrIdx = minIdx
     nGrIdx = minIdx + 1
   elseif (( minIdx > 0) and ( minIdx < PtAnzIdx)) then
     vGrIdx = minIdx -1
     nGrIdx = minIdx + 1
   elseif ( minIdx = PtAnzIdx) then
```

```
      vGrIdx = minIdx -1
      nGrIdx = minIdx
    end
  elseif (minDist >= 10) then  'Berechnung der Koordinaten des Punktes
    vGrIdx = vIdx      'auf der Linie als Projektion des Maus-Klickens
    nGrIdx = nIdx          'auf die Linie (Profilschnitt)
    if  (vPtx  = aMPtx) then
        Ptx = vPtx
        Pty = vPty
    elseif ((vPtx  <> aMPtx) and (aMPtx <> nPtx)) then
        if (vPty = nPty) then
              Ptx = aMPtx
              Pty = vPty
        elseif (vPty < nPty) then
              if (nPtx = vPtx) then
                    Ptx = nPtx
                    Pty = nPty
              elseif (nPtx <> vPtx) then
                    xnvDiff = nPtx - vPtx
                    ynvDiff = nPty - vPty
                    xMvDiff = aMPtx - vPtx
                    Ptx = aMPtx
                    Pty  = (ynvDiff / xnvDiff) * xMvDiff + vPty
              end
        elseif (vPty > nPty) then
              if (nPtx = vPtx) then
                    Ptx = nPtx
                    Pty = nPty
              elseif (nPtx <> vPtx) then
                    xnvDiff = nPtx - vPtx
                    yvnDiff = vPty - nPty
                    xnMDiff = nPtx - aMPtx
                    Ptx = aMPtx
                    Pty  = (yvnDiff / xnvDiff) * xnMDiff + nPty
              end
        end
    elseif (nPtx  = aMPtx)  then
        Ptx = nPtx
        Pty = nPty
    end
  end          ' Ende von (if ((minDist = 0) or (minDist < 10)) then)
  ListofGrPt.Add(Ptx@Pty)       'Ein Grenz-Punkt für den Abschnitt

  ListofGrIdx = {}
  ListofGrIdx.Add(vGrIdx)
  ListofGrIdx.Add(nGrIdx)
  ListofListofGrIdx.Add(ListofGrIdx)

  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/3*100)
  if (not more) then
    break
  end
end          ' Ende der Schleife für Abschnitt


'Neues Polygon oder neue Polyline des Datensatzes wird
'durch Ersetzen des Abschnittes hergestellt.
ListofNT = {}

'Liste der Stützunkte der Polyline oder des Polygons
'zur Korrektur des Abschnittes
```

```
AnzFLPt = ListofFLPt.Count
IdxFLPt = AnzFLPt - 1

ListofTPt11 = {}   'Teilliste der Polyline oder des Polygons
ListofTPt22 = {}
ListofTPt33 = {}

AnfFLPt = ListofGrPt.Get(0)
EndFLPt = ListofGrPt.Get(1)

ListofGrIdxAnf = ListofListofGrIdx.Get(0)
ListofGrIdxEnd = ListofListofGrIdx.Get(1)

vGrIdxAnf = ListofGrIdxAnf.Get(0)
nGrIdxAnf = ListofGrIdxAnf.Get(1)

vGrIdxEnd = ListofGrIdxEnd.Get(0)
nGrIdxEnd = ListofGrIdxEnd.Get(1)

'MsgBox.Report("vGrIdxAnf:"++vGrIdxAnf.AsString+NL+
'             "nGrIdxAnf:"++nGrIdxAnf.AsString+NL+
'             "vGrIdxEnd:"++vGrIdxEnd.AsString+NL+
'             "nGrIdxEnd:"++nGrIdxEnd.AsString, "Kontrolle")


if (vGrIdxAnf < vGrIdxEnd) then
  vKlIdx = vGrIdxAnf
  vgrIdx = vGrIdxEnd
  nKlIdx = nGrIdxAnf
  ngrIdx = nGrIdxEnd
  klPt = AnfFLPt
  grPt = EndFLPt
elseif (vGrIdxAnf > vGrIdxEnd) then
  vKlIdx = vGrIdxEnd
  vgrIdx = vGrIdxAnf
  nKlIdx = nGrIdxEnd
  ngrIdx = nGrIdxAnf
  klPt = EndFLPt
  grPt = AnfFLPt
end

for each i in 0..IdxFLPt
   aPt = ListofFLPt.Get(i)
   if (i <= vKlIdx) then
      ListofTPt11.Add(aPt)
   elseif ((i >= nKlIdx) and (i <= vgrIdx)) then
      ListofTPt22.Add(aPt)
   elseif (i >= ngrIdx) then
      ListofTPt33.Add(aPt)
   end
end

ListofListofTPt1100 = {}
ListofListofTPt2200 = {}
ListofListofTPt3300 = {}

ListofListofTPt1100.Add(ListofTPt11)
ListofListofTPt2200.Add(ListofTPt22)
ListofListofTPt3300.Add(ListofTPt33)

aPL1100 = Polyline.Make(ListofListofTPt1100)
aPL2200 = Polyline.Make(ListofListofTPt2200)
```

```
aPL3300 = Polyline.Make(ListofListofTPt3300)

'Einsetzen der ausgewählten Punkte auf Polyline oder Polygon

if (PglShpClassStr = "Polyline") then
  for each i in 0..IdxFLPt
    if (i <= vKlIdx) then
      aPt = ListofFLPt.Get(i)
      ListofNT.Add(aPt)
    end
  end
  ListofNT.Add(klPt)

  AnzTPt2211 = ListofPgPT.Count
  IdxTPt2211 = AnzTPt2211 - 1

  'Feststellung der Reihenfolge
  minklIdx = 1000000
  minAbst = 1000000
  for each i in 0..IdxTPt2211
    aPt = ListofPgPT.Get(i)
    aPtx = aPt.Getx
    aPty = aPt.Gety
    klPtx = klPt.Getx
    klPty = klPt.Gety
    Abst = (((aPtx - klPtx) ^ 2) + ((aPty - klPty) ^ 2)).Sqrt.Abs
    if (Abst < minAbst) then
      minAbst = Abst
      minklIdx = i
    end
  end

  if (minklIdx < 5) then
    for each i in 0..IdxTPt2211
      aPt = ListofPgPT.Get(i)
      ListofNT.Add(aPt)
    end
  elseif (minklIdx > (IdxTPt2211-4)) then
    Ng = -1
    for each i in 0..IdxTPt2211
      Ng = Ng + 1
      umkIdx = IdxTPt2211 - Ng
      aPt = ListofPgPT.Get(umkIdx)
      ListofNT.Add(aPt)
    end
  end
  ListofNT.Add(grPt)
  for each i in 0..IdxFLPt
    if (i >= ngrIdx) then
      aPt = ListofFLPt.Get(i)
      ListofNT.Add(aPt)
    end
  end

elseif (PglShpClassStr = "Polygon") then

  'Auswahl einer Teilliste des Polygons zur Korrektur
  nklPt = ListofFLPt.Get(nKlIdx)
  vgrPt = ListofFLPt.Get(vgrIdx)
  aCnkl = Circle.Make(nklPt, 5)
  aCvgr = Circle.Make(vgrPt, 5)
```

```
kTest11 = aCnkl.Intersects(aPL1100)
kTest22 = aCnkl.Intersects(aPL2200)
kTest33 = aCnkl.Intersects(aPL3300)

gTest11 = aCvgr.Intersects(aPL1100)
gTest22 = aCvgr.Intersects(aPL2200)
gTest33 = aCvgr.Intersects(aPL3300)


'MsgBox.Report("kTest11:"++"  "++kTest11.AsString+NL+
'             "kTest22:"++"  "++kTest22.AsString+NL+
'             "kTest33:"++"  "++kTest33.AsString,
'             "Ausgewählte Teilliste des Polygons")

'MsgBox.Report("gTest11:"++"  "++gTest11.AsString+NL+
'             "gTest22:"++"  "++gTest22.AsString+NL+
'             "gTest33:"++"  "++gTest33.AsString,
'             "Ausgewählte Teilliste des Polygons")

'MsgBox.ListAsString(ListofPgPT, "ListofPgPT", "Information")

'Herstellung eines neuen Polygons (zum Teil ersetzt)

if ((kTest22) or (gTest22)) then
  for each i in 0..IdxFLPt
    if (i <= vKlIdx) then
      aPt = ListofFLPt.Get(i)
      ListofNT.Add(aPt)
    end
  end
  ListofNT.Add(klPt)

  AnzTPt2211 = ListofPgPT.Count
  IdxTPt2211 = AnzTPt2211 - 1

  'Feststellung der Reihenfolge
  minklIdx = 1000000
  minAbst = 1000000
  for each i in 0..IdxTPt2211
    aPt = ListofPgPT.Get(i)
    aPtx = aPt.Getx
    aPty = aPt.Gety
    klPtx = klPt.Getx
    klPty = klPt.Gety
    Abst = (((aPtx - klPtx) ^ 2) + ((aPty - klPty) ^ 2)).Sqrt.Abs
    if (Abst < minAbst) then
       minAbst = Abst
       minklIdx = i
    end
  end

  'MsgBox.Info(minklIdx.AsString, "minklIdx")
  if (minklIdx < 6) then
    for each i in 0..IdxTPt2211
       aPt = ListofPgPT.Get(i)
       ListofNT.Add(aPt)
    end
  elseif (minklIdx > (IdxTPt2211-4)) then
    Ng = -1
    for each i in 0..IdxTPt2211
       Ng = Ng + 1
       umkIdx = IdxTPt2211 - Ng
```

```
          aPt = ListofPgPT.Get(umkIdx)
          ListofNT.Add(aPt)
        end
      end
      ListofNT.Add(grPt)
      for each i in 0..IdxFLPt
        if (i >= ngrIdx) then
          aPt = ListofFLPt.Get(i)
          ListofNT.Add(aPt)
        end
      end

    elseif (((kTest11) or (kTest33)) or ((gTest11) or (gTest33))) then
      ListofNT.Add(klPt)
      for each i in 0..IdxFLPt
        if ((i >= nKlIdx) and (i <= vgrIdx)) then
          aPt = ListofFLPt.Get(i)
          ListofNT.Add(aPt)
        end
      end
      ListofNT.Add(grPt)

      AnzTPt2211 = ListofPgPT.Count
      IdxTPt2211 = AnzTPt2211 - 1

      'Feststellung der Reihenfolge
      minklIdx = 1000000
      minAbst = 1000000
      for each i in 0..IdxTPt2211
        aPt = ListofPgPT.Get(i)
        aPtx = aPt.Getx
        aPty = aPt.Gety
        klPtx = klPt.Getx
        klPty = klPt.Gety
        Abst = (((aPtx - klPtx) ^ 2) + ((aPty - klPty) ^ 2)).Sqrt.Abs
        if (Abst < minAbst) then
          minAbst = Abst
          minklIdx = i
        end
      end

      if (minklIdx < 6) then
        for each i in 0..IdxTPt2211
          aPt = ListofPgPT.Get(i)
          ListofNT.Add(aPt)
        end
      elseif (minklIdx > (IdxTPt2211-4)) then
        Ng = -1
        for each i in 0..IdxTPt2211
          Ng = Ng + 1
          umkIdx = IdxTPt2211 - Ng
          aPt = ListofPgPT.Get(umkIdx)
          ListofNT.Add(aPt)
        end
      end
    end
  end
end

'Herstellung einer neuen Polyline oder eines neuen Polygons im View
ListofPLSchtN={"Geländeoberfläche", "Basisfläche der Deckschichten (TOK)",
   "NT abgedeckte Fläche","MT abgedeckte Fläche","Quartärbasis",
```

```
  "Oberfläche der als MT ältere Schichten",
  "Oberfläche der Präquartär-Schichten"}

ListofPLKW = {"GH", "TOK", "NTabF", "MTabF", "QB", "Präm", "Präq"}
aListofPLNr = {53, 14, 5, 8, 8, 8, 8}

ListofPgSchtN={"Deckschichten (Mutter oder Waldboden)",
  "Deckschichten (Lehm oder Sand)","Untere Mittelterrasse 2",
  "Untere Mittelterrasse 1", "Holstein (Torf, z.T. Tonhaltig)",
  "Holstein (Sandschichten mit Tonlagen)", "Holstein (Tonschichten)",
  "Holstein (Schluffschichten)",
  "Sandschichten (z.T. Holstein, z.T. Rinnenschotter)",
  "Die mittlere Mittelterrasse (Rinnenschotter)",
  "Niederterrasse", "Mittelterrasse","Untere Mittelterrasse",
  "Obere Mittelterrasse", "Hauptterrasse", "als MT ältere Schichten",
  "Präquartär-Schichten","Untere Mittelterrasse III",
  "Untere Mittelterrasse IV"}
ListofPgKW = {"Deck-Mu", "Deck-LS", "UMT2", "UMT1", "Holst-Tf",
          "Holst-Sd", "Holst-Ton", "Holst-U", "SdScht-HR", "MMT-R",
          "NT", "MT", "UMT", "OMT", "HT", "Präm", "Präq", "UMT III",
          "UMT IV"}
aListofPgNr ={53, 47, 26, 41, 35, 38, 23, 44, 45, 20,
           14, 20, 26, 41, 32, 3, 3, 26, 26}

ListofListofnPglPT={}

if (aAufg = "Herstellung einer Polyline") then
  ListofListofnPglPT.Add(ListofPgPT)
  theNShp = Polyline.Make(ListofListofnPglPT)
  theScht = MsgBox.ChoiceAsString(ListofPLSchtN, "Auswahl der Name:",
        "Bezeichnung der Polyline")
  SchtIdx = ListofPLSchtN.FindByValue(theScht)
  theKW = ListofPLKW.Get(SchtIdx)

elseif (aAufg = "Korrektur einer Polyline") then
  ListofListofnPglPT.Add(ListofNT)
  theNShp = Polyline.Make(ListofListofnPglPT)
  if (PglSchtFld <> nil) then
    theScht = PglFTab.ReturnValue(PglSchtFld, aSRIdx)
  elseif (PglSchtFld = nil) then
    theScht = MsgBox.ChoiceAsString(ListofPLSchtN, "Auswahl der Name:",
          "Bezeichnung der Polyline")
  end
  theKW = aSBz

elseif (aAufg = "Herstellung eines Polygons") then
  ListofListofnPglPT.Add(ListofPgPT)
  theNShp = Polygon.Make(ListofListofnPglPT)
  theScht = MsgBox.ChoiceAsString(ListofPgSchtN, "Auswahl der Name:",
        "Bezeichnung des Polygons")
  SchtIdx = ListofPgSchtN.FindByValue(theScht)
  theKW = ListofPgKW.Get(SchtIdx)

elseif (aAufg = "Korrektur eines Poygons") then
  ListofListofnPglPT.Add(ListofNT)
  theNShp = Polygon.Make(ListofListofnPglPT)
  if (PglSchtFld <> nil) then
    theScht = PglFTab.ReturnValue(PglSchtFld, aSRIdx)
  elseif (PglSchtFld = nil) then
    theScht = MsgBox.ChoiceAsString(ListofPgSchtN, "Auswahl der Name:",
          "Bezeichnung der Polyline")
  end
```

```
  theKW = aSBz

end

'Speicherung der neuen Polyline oder des neuen Polygons
ListofSp11 = {"Herstellung als ein neues Thema",
            "Speicherung in einem vorhandenen Thema"}
ListofSp22 = {"das zur Korrektur ausgewählte Thema",
            "ein im View vorhandenes Thema"}

AW11 = MsgBox.ListAsString(ListofSp11,
     "zur Speicherung der neuen Polyline oder des Polygons",
     "Auswahl eines Themas")

if (AW11 = "Herstellung als ein neues Thema") then
  WDStr=av.GetProject.GetWorkDir.AsString
  if ((aAufg = "Herstellung einer Polyline") or
      (aAufg = "Korrektur einer Polyline")) then
    fn00 = "Pl"+thePtTheme.AsString.Left(4)
    fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
    'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
    fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Polyline)")
    if (fName =nil) then exit end
    fName.SetExtension("shp")
    NFTab=FTab.MakeNew(fName, Polyline)

  elseif ((aAufg = "Herstellung eines Polygons") or
        (aAufg = "Korrektur eines Poygons")) then
    fn00 = "Pg"+thePtTheme.AsString.Left(4)
    fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
    'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
    fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Polygon)")
    if (fName =nil) then exit end
    fName.SetExtension("shp")
    NFTab=FTab.MakeNew(fName, Polygon)

  end

  NShpFld = NFTab.FindField("shape")
  NIDFld=Field.Make("ID", #Field_Short, 3, 0)
  NSchtFld=Field.Make("Schichten", #Field_Char, 60, 0)
  NKWFld=Field.Make("Kennwort", #Field_Char, 10, 0)
  ListofNFlds={NIDFld, NSchtFld, NKWFld}
  NFTab.AddFields(ListofNFlds)
  NThm=FTheme.Make(NFTab)
  theView.AddTheme(NThm)
  recNr = -1

elseif (AW11 = "Speicherung in einem vorhandenen Thema") then
   AW22 = MsgBox.ListAsString(ListofSp22,
      "zur Speicherung der neuen Polyline oder des Polygons",
      "Auswahl eines Themas")
  if (AW22 = "das zur Korrektur ausgewählte Thema") then
      recNr = aSRIdx

   elseif (AW22 = "ein im View vorhandenes Thema") then

     if (PglShpClassStr = "Polyline") then
       NThm=MsgBox.ChoiceAsString(ListofPLThms,
         "zur Speicherung der neuen Polyline",
         "Auswahl eines Themas im View"++theView.AsString)
```

```
    elseif (PglShpClassStr = "Polygon") then
      NThm=MsgBox.ChoiceAsString(ListofPgThms,
        "zur Speicherung des neuen Polygons",
        "Auswahl eines Themas im View"++theView.AsString)
    end

    NFTab = NThm.GetFTab
    NShpFld = NFTab.FindField("Shape")
    NIDFld = NFTab.FindField("ID")

    'Anzahl der Datensätze im Thema
    AnzNDs=0
    for each rec in NFTab
      AnzNDs = AnzNDs + 1
    end
    IdxNDs = AnzNDs - 1
    recNr = IdxNDs

    'Anzahl der Felder im Thema
    ListofNFlds = NFTab.GetFields
    AnzNFlds = ListofNFlds.Count
    IdxNFlds = AnzNFlds - 1

   'Feststellung der Datensätze im Thema
    ListofPgDatens = {}
    FldStr=""
    for each j in 0..IdxNDs
      DtStr=""
      for each i in 0..IdxNFlds
        aFld = ListofNFlds.Get(i)
        aFldStr = aFld.GetName
        if (aFldStr <> "Shape") then
          if (j = 0) then
            FldStr = FldStr + aFldStr+"; "
          end
          aValue = NFTab.ReturnValue(aFld, j)
          DtStr=DtStr+aValue.AsString+"; "
        end
      end
      ListofPgDatens.Add(DtStr)
    end

    MsgBox.ListAsString(ListofPgDatens, FldStr,
       "Die Namen der Felder und Datensätze")
    NSchtFld = MsgBox.ListAsString(ListofNFlds,
      "für Beschreibung der Schichten",
      "Auswahl eines Feldes des Themas"++NThm.AsString)
    NKWFld = MsgBox.ListAsString(ListofNFlds,
      "für Kennwort der Schichten",
      "Auswahl eines Feldes des Themas"++NThm.AsString)
  end
end

av.ShowMsg("Herstellung des neuen Themas für Schichten ...")
av.ShowStopButton


if ((AW11 = "Speicherung in einem vorhandenen Thema") and
   (AW22 = "das zur Korrektur ausgewählte Thema")) then
  PglFTab.SetEditable(false)
  PglFTab.SetEditable(true)
```

```
  PglFTab.SetValue(PglShpFld, recNr, theNShp)

  PglFTab.SetEditable(false)
  theTheme = PglThm
else
  NFTab.SetEditable(false)
  NFTab.SetEditable(true)

  recNr = recNr + 1
  NFTab.AddRecord
  NFTab.SetValue(NShpFld, recNr, theNShp)
  NFTab.SetValue(NIDFld, recNr, recNr)
  NFTab.SetValue(NSchtFld, recNr, theScht)
  NFTab.SetValue(NKWFld, recNr, theKW)

  NFTab.SetEditable(false)
  theTheme = NThm
end

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

'Definition der Farbe der Legende
'Bezeichnung der Datensätze
aListofgB={"---","a","a/Mj","Aussen","d","f","Gy","H",
    "hg/plRR","Hj","Hn","Lf","Lfh/N","Lö","Lö/Hj","Lö/Mj",
    "Löy","Ma","mi-olK","miIV","Mj","N","plRR","Rhein",
    "Sf","Sfh/N","sSo","tAb","tTt","Sonst",
    "Deck","D","De","NT","MT","HT","Präq","Präm",
    "GH","NA","NQ","NT",
    "Deck","QmitD","QohneD","QohneT","TrorAe",
    "Deckschichten","Niederterrassen","Mittelterrassen",
    "Präquartäre Schichten",
    "rMTl", "lMTl", "rHTl", "lNT", "rNT", "lHTr", "rHTr",
    "HTr", "MTl", "MTr", "HTl",
    "UMT", "UMT2", "UMT3", "UMT4", "OMT",
    "TOK", "NTabF", "MTabF", "QB", "Deck-LS"}
'Die Nummer der Farbe im Symbolwindow
aListofNr={20,43,24,5,36,8,24,32,3,32,11,17,14,47,30,
        24,6,29,35,11,26,16,53,20,18,16,24,53,51,54,
        53,53,53,14,26,32,3,3,53,5,8,14,53,20,26,8,3,
        53,14,26,3,20,26,32,14,17,32,32,32,26,20,32,
        26,26,26,26,20,14,5,8,8,47}
Legendeinfo = 0


av.ShowMsg("Veränderung der Legenden des Themas"
      ++theTheme.AsString+"...")

theLegend = theTheme.GetLegend

if ((AW11 = "Speicherung in einem vorhandenen Thema") and
   (AW22 = "das zur Korrektur ausgewählte Thema")) then

 if (aorgLTyp = #LEGEND_TYPE_SIMPLE) then
  theLegend.SetLegendType(#LEGEND_TYPE_SIMPLE)
  theLegend.SingleSymbol
  theSymbol=theLegend.GetSymbols.Get(0)
  theSymbol.SetColor(ListoforgColor.Get(0))
 elseif (aorgLTyp = #Legend_Type_Unique) then
  theLegend.SetLegendType(#Legend_Type_Unique)
  theLegend.Unique(theTheme, PglKWFldStr)
```

```
    ListofKlasse=theLegend.GetClassifications

    aListofSymbol=theLegend.GetSymbols
    AnzSymb=aListofSymbol.Count
    AnzSymbIdx=AnzSymb-2

    for each symb in 0..AnzSymbIdx
      aColor = ListoforgColor.Get(symb)
      aListofSymbol.Get(symb).SetColor(aColor)
    end
  end

else
    theLegend.SetLegendType(#Legend_Type_Unique)
    NKWFldStr = NKWFld.AsString
    theLegend.Unique(theTheme, NKWFldStr)
    ListofKlasse=theLegend.GetClassifications
    AnzKlasse=ListofKlasse.Count
    IdxKlasse=AnzKlasse-2
    'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")

    aListofSColor = {}
    for each i in 0..IdxKlasse
      theKlasseLb=ListofKlasse.Get(i).GetLabel
      'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
      if (PglShpClassStr = "Polyline") then
        aIdxLbPL = ListofPLKW.FindByValue(theKlasseLb)
        if (aIdxLbPL <> -1) then
          aCNr = aListofPLNr.Get(aIdxLbPL)
        elseif (aIdxLbPL = -1) then
          aIdxLbPL2 = aListofgB.FindByValue(theKlasseLb)
          if (aIdxLbPL2 <> -1) then
            aCNr = aListofNr.Get(aIdxLbPL2)
          elseif (aIdxLbPL2 = -1) then
            if (i < 60) then
              aCNr = i
            elseif (i > 59) then
              aCNr = i - 59
            end
            Legendeinfo = 1
          end
        end
      elseif (PglShpClassStr = "Polylgon") then
        aIdxLbPg = ListofPgKW.FindByValue(theKlasseLb)
        if (aIdxLbPg <> -1) then
          aCNr = aListofPgNr.Get(aIdxLbPg)
        elseif (aIdxLbPg = -1) then
          aIdxLbPg2 = aListofgB.FindByValue(theKlasseLb)
          if (aIdxLbPg2 <> -1) then
            aCNr = aListofNr.Get(aIdxLbPg2)
          elseif (aIdxLbPg2 = -1) then
            if (i < 60) then
              aCNr = i
            elseif (i > 59) then
              aCNr = i - 59
            end
            Legendeinfo = 1
          end
        end
      end
      theColor=(thePalette.GetList(#PALETTE_LIST_COLOR).Get(aCNr))
      theRgbList=theColor.GetRgbList
```

```
      aColor=Color.Make
      aColor.SetRgbList(theRgbList)
      aListofSColor.Add(aColor)
    end

    aListofSymbol=theLegend.GetSymbols
    AnzSymb=aListofSymbol.Count
    AnzSymbIdx=AnzSymb-2

    for each symb in 0..AnzSymbIdx
      aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
    end
    if (Legendeinfo = 1) then
      MsgBox.Report("Die Farbe der Legende ist zum Teil"
        +NL+"oder gar nicht definiert!"
        +NL+"Die Definition der Farbe"
        +NL+"in diesem Programm ist zu ändern.",
      "Information")
    end
end

theTheme.UpdateLegend




'pglkoabm.ave
'Ein Abschnitt eines Polygons oder einer Polyline in einem Thema wird
'durch einen Abschnitt eines Polygons oder einer Polyline
'(z.B. einer Höhenlinie, C1 S(G) Gtqpt111.shp) in einem gleichen
'oder in einem anderen Thema ersetzt.
'Dieses Script wird als ein Werkzeug (Tool) im aktiven View benutzt.
'Wenn die beiden Formen Polyline sind, wird die Maus 2-mal geklickt.
'Wenn eine von den beiden ein Polygon ist, wird die Maus 3-mal geklickt.
'Wenn die beiden Polygone sind, wird die Maus 4-mal geklickt.
'Die Funktion des Maus-Klickens:
'Bei den beiden Polyline: 2-mal: Bestimmung eines Abschnittes.
'Bei einem Polygon von beiden: 3-mal:
'1. und 3. Klicken: Bestimmung eines Abschnittes;
'2. Klicken: Auswahl eines Teils des Polygons
'Bei den beiden Polygonen: 4-mal:
'1. und 3. Klicken: Bestimmung eines Abschnittes;
'2. und 4. Klicken: Auswahl der Teile der Polygone.

theProject=av.GetProject
AkView=av.GetActiveDoc     'Ein aktives View-Thema

'Maus-Klicken auf das Bildschirm
aEingPolyL = AkView.GetDisplay.ReturnUserPolyLine

ListofListofEingPt = aEingPolyL.AsList
ListofEingPt = ListofListofEingPt.Get(0)
AnzM = ListofEingPt.Count
AnzMIdx = AnzM - 1
ListofMPt = {}
ListofMCPg = {}
for each mPt in 0..AnzMIdx
  aMPt = ListofEingPt.Get(mPt)
  ListofMPt.Add(aMPt)
  aCircle = Circle.Make(aMPt, 50)
  aMCPg = aCircle.AsPolygon
```

```
  ListofMCPg.Add(aMCPg)
end

AkTheme=AkView.GetActiveThemes.Get(0)
AkStr = AkTheme.AsString
qt00=MsgBox.YesNo("Ist das aktive Thema"++AkStr
     +NL+"zur Korrektur richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not qt00) then
  MsgBox.Error("Das aktive Thema"++AkStr++"ist falsch!"+NL+
           "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

ListofThemen = {}
ListofThemen.Add(AkTheme)
ListofZiel = {"zu korrigieren", "zu übernehmen"}
ListofGrPt = {}
ListofListofFLPt = {}
ListofRecIdx = {}
ListofListofGrIdx = {}
ListofField = {}
ListofLegendTyp = {}
ListofListoforgColor = {}
ListofaShpClassStr = {}
ListofMKlicken = {"Uhrzeigersinn", "Gegenuhrzeigersinn"}

ListofThms=AkView.GetThemes
ListofPLgThm = {}     'Auswahl der Themen (PolyLine oder Polygone)
for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aSCl = aFTab.GetShapeClass
      if ((aSCl.IsSubclassOf(PolyLine)) or
         (aSCl.IsSubclassOf(Polygon))) then
        ListofPLgThm.Add(aT)
      end
   end
end

CtTheme=MsgBox.ChoiceAsString(ListofPLgThm,
      "um einen Abschnitt zu übernehmen"+NL+
      "Der Name einer Polyline oder eines Polygons",
      "Auswahl eines Themas,")
ListofThemen.Add(CtTheme)

av.ShowMsg("Feststellung der Abschnitte der Polyline"
        ++"oder des Polygons ...")
av.ShowStopButton
Ng = 0
for each j in 0..1    'Schleife für Themen
  aThm = ListofThemen.Get(j)
  aTStr = aThm.AsString
  aFTab = aThm.GetFTab
  aShpClassStr = aFTab.GetShapeClass.GetClassName
  ListofaShpClassStr.Add(aShpClassStr)

  aListofFlds = aFTab.GetFields
  AnzFld = aListofFlds.Count
  IdxFld = AnzFld - 1
  aShpFld = aListofFlds.Get(0)
```

```
AnzDs = 0          'Anzahl der Datensätze
for each rec in aFTab
  AnzDs = AnzDs + 1
end
IdxDs = AnzDs - 1

'Feststellung des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxFld
  aFld = aListofFlds.Get(i)
  aFldStr=aFld.GetName
  if (aFldStr <> "Shape") then
       FldStr = FldStr + aFldStr+"; "
       aValue = aFTab.ReturnValue(aFld, 0)
       DtStr=DtStr+aValue.AsString+"; "
  end
end
aZiel = ListofZiel.Get(j)
MsgBox.Report("vom Thema"++aTStr+","+NL+
      "um ein Polygon oder eine Polyline zum Teil"+NL+aZiel+NL+NL+
      "Die Namen der Felder ohne Shape-Felder"
       +NL+FldStr+NL+NL+
      "Der erste Datensatz:"+NL+DtStr,
      "Information")
aSchtFld = MsgBox.ListAsString(aListofFlds,
  "des Themas"++aTStr+NL+
  "für Bezeichnung der Datensäze und"+NL+
  "für Klassifizierung der Legende, um"++aZiel,
  "Auswahl eines Feldes")
ListofField.Add(aSchtFld)
aSchtFldStr = aSchtFld.AsString

aLegend = aThm.GetLegend
aorgLTyp = aLegend.GetLegendType
ListofLegendTyp.Add(aorgLTyp)
'MsgBox.Info(aorgLTyp.AsString, "eigentliche Legende")
aListofSymbol = aLegend.GetSymbols
AnzSymb = aListofSymbol.Count
AnzSymbIdx = AnzSymb-1
ListoforgColor = {}

for each symb in 0..AnzSymbIdx
    aorgColor = aListofSymbol.Get(symb).GetColor
    ListoforgColor.Add(aorgColor)
end
ListofListoforgColor.Add(ListoforgColor)

aLegend.SetLegendType(#Legend_Type_Unique)
aLegend.Unique(aThm, aSchtFldStr)
ListofKlasse = aLegend.GetClassifications
AnzKlasse = ListofKlasse.Count
IdxKlasse = AnzKlasse-1
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")


ListofKlBez = {}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   ListofKlBez.Add(theKlasseLb)
end
```

```
theKlBz=MsgBox.ListAsString(ListofKlBez,
  "im"++aTStr++"zur Auswahl der Polyline oder des Polygons, um"
    ++aZiel,
  "Auswahl einer Bezeichnung der Datensätze")
if (theKlBz = nil) then
  MsgBox.Error("Die Bezeichnung der Polyline, des Polygons oder"
     +NL+"die Polyline oder das Polygon mit der Bezeichnung fehlt!"
     +NL+"Das Programm wird abgebrochen!", "")
 exit
end

'Selection der Datensätze im Thema mit den Maus-Klicken
ListofSBz = {}
ListofIdxS = {}

if (aThm.CanSelect) then
  'MsgBox.Info("Das Thema"++aTStr++"ist selectierbar.","CanSelect?")
  for each rec in 0..AnzMIdx
     aMCPg = ListofMCPg.Get(rec)
     if (rec = 0) then
        aThm.SelectByPolygon(aMCPg, #VTAB_SELTYPE_NEW)
     elseif (rec > 0) then
        aThm.SelectByPolygon(aMCPg, #VTAB_SELTYPE_OR)
     end
  end
  'die selektierten Datensätze
  AnzS = 0
  for each rec in aFTab.GetSelection
    'MsgBox.Info("von"++aTStr++":  "++rec.AsString,
    '    "Index-Nummer der selektierten Datensätze")
    aSBz = aFTab.ReturnValue(aSchtFld, rec)
    ListofSBz.Add(aSBz)
    AnzS = AnzS + 1
    ListofIdxS.Add(rec.AsString)
  end
  'MsgBox.Info(AnzS.AsString, "Anzahl der selektierten Datensätze")
  'MsgBox.ListAsString(ListofIdxS, "Index-Nummer der selektierten"
  '                  ++"Datensätze"++"von"++aTStr, "Information")
else
  for each i in 0..IdxDs
     aBz = aFTab.ReturnValue(aSchtFld, i)
     if (aBz = theKlBz) then
        ListofSBz.Add(aBz)
        ListofIdxS.Add(i.AsString)
     end
  end
end

aSBz = MsgBox.ListAsString(ListofSBz,
      "von"++aTStr++","++
      "um die Polyline oder das Polygon zum Teil"++aZiel,
      "Auswahl eines Datensatzes")
aSBzIdx = ListofSBz.FindByValue(aSBz)
aSRIdx = ListofIdxS.Get(aSBzIdx).AsNumber

'MsgBox.Info("von"++aTStr++":  "++aSRIdx.AsString,
'      "Index des ausgewählten Datensatzes")
theShp = aFTab.ReturnValue(aShpFld, aSRIdx)
'MsgBox.report("des Themas"++aTStr+NL+
'             "Index-Nummer:"++aSRIdx.AsString+NL+NL+
'             theShp.AsString, "Selektiertes Shape")
```

```
aThm.ClearSelection

'Feststellung der Koordinaten der ausgewählten Figur im Thema

ListofRecIdx.Add(aSRIdx)
theProfilList1={}
theProfilList1.Add({theShp})

'Liste der Vertices
ListofFLPt = {}
minxkrd = 100000000
minykrd = 100000000
maxxkrd = 0
maxykrd = 0

if (theProfilList1 <> 0) then
  for each q in theProfilList1
      theLines=q.Get(0).AsList
      for each m in theLines
          for each ptx in m
            myx=ptx.Getx
            myy=ptx.Gety
            ListofFLPt.Add(myx@myy)
            if (myx < minxkrd) then
              minxkrd = myx
            end
            if (myx > maxxkrd) then
              maxxkrd = myx
            end
            if (myy < minykrd) then
              minykrd = myy
            end
            if (myy > maxykrd) then
              maxykrd = myy
            end
          end
      end
  end
end

PtAnz= ListofFLPt.Count  'Anzahl der Vertices
PtAnzIdx= PtAnz-1
ListofListofFLPt.Add(ListofFLPt)

'Bestimmung des Abschnittes als Index der Vertices
'mit dem kleinsten Abstand von den Maus-Klicken

for each aM in 0..AnzMIdx   ' Anfang der Schleife für einen Abschnitt
 Ng = Ng + 1
 minDist=1000
 minIdx=-1

 aMPt = ListofMPt.Get(aM)
 aMPtx = aMPt.Getx
 aMPty = aMPt.Gety

 if (aMPtx < minxkrd) then
   aMPtx = minxkrd
 elseif (aMPtx > maxxkrd) then
   aMPtx = maxxkrd
 end
 if (aMPty < minykrd) then
```

```
  aMPty = minykrd
elseif (aMPty > maxykrd) then
  aMPty = maxykrd
end

for each i in 0..PtAnzIdx
   aPt = ListofFLPt.Get(i)
   xkrd= aPt.Getx
   ykrd= aPt.Gety
   xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
   yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
   xyAbst = ((xAbst  + yAbst).sqrt) .Abs
   if (xyAbst < minDist) then
      minDist = xyAbst
      minIdx = i  'Index des nähesten Punktes des Maus-Klickens
   end
end

'Bestimmung des Abschnittes des Shapes
'als x-, y-Koordinaten

if (minIdx = 0) then
  vIdx = 0     'ein Vertex auf der Linie vor dem Maus-Klicken
  nIdx = 1     'ein Vertex auf der Linie nach dem Maus-Klicken
elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
  aPtv = ListofFLPt.Get((minIdx - 1))
  aPt0 = ListofFLPt.Get(minIdx)
  aPtn = ListofFLPt.Get((minIdx + 1))

  xkrdv = aPtv.Getx   'x-Koord. des letzten Punktes
  xkrd  = aPt0.Getx   'x-Koord. des nähesten Punktes
  xkrdn = aPtn.Getx   'x-Koord. des nächsten Punktes

  ykrdv = aPtv.Gety   'y-Koord. des letzten Punktes
  ykrd  = aPt0.Gety          'y-Koord. des nähesten Punktes
  ykrdn = aPtn.Gety   'y-Koord. des nächsten Punktes

  xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
  yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
  xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten Punkt vom nähesten Punkt

  xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
  yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
  xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten Punkt vom nähesten Punkte

  xML = (xkrd - aMPtx) * (xkrd - aMPtx)
  yML = (ykrd - aMPty) * (ykrd - aMPty)
  xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Maus-Klicken vom nähesten Punkt

  xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
  yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
  xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Maus-Klicken vom letzten Punkt

  xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
  yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
  xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Maus-Klicken vom nächsten Punkt

  vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken und dem
  vLpML = xyAv + xyML            'letzten Punkt

  nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken und dem
  nLpML = xyAn + xyML             'nächsten Punkt
```

```
  vM = (vLmML - xyMLv).Abs   'Differenz zwischen der theorischen und der
  vP = (vLpML - xyMLv).Abs     'tatsächlichen Länge im Bezug auf den letzten Punkt

  nM = (nLmML - xyMLn).Abs   'Differenz zwischen der theorischen und der
  nP = (nLpML - xyMLn).Abs     'tatsächlichen Länge im Bezug auf den nächsten Punkt

  ListofLg = {vM, vP, nM, nP}   'Suche nach der kleinsten Länge
  MinLg = 10000
  TheLgIdx = -1
  For each aLg in 0..3
     theLg = ListofLg.Get(aLg)
     if (theLg < MinLg) then
          MinLg = theLg
          TheLgIdx = aLg
     end
  end
  if (xyML < 10) then
    vIdx = minIdx - 1
    nIdx = minIdx + 1
  elseif (xyML >= 10) then
    if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
         vIdx = minIdx - 1     'ein Vertex auf der Linie vor dem Maus-Klicken
         nIdx= minIdx          'ein Vertex auf der Linie nach dem Maus-Klicken
    elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
         vIdx = minIdx         'ein Vertex auf der Linie vor dem Maus-Klicken
         nIdx = minIdx + 1     'ein Vertex auf der Linie nach dem Maus-Klicken
    else
         vIdx = minIdx         'ein Vertex auf der Linie vor dem Maus-Klicken
         nIdx = minIdx + 1     'ein Vertex auf der Linie nach dem Maus-Klicken
    end
  end
elseif (minIdx = PtAnzIdx)   then
   vIdx = minIdx - 1
   nIdx = minIdx
end

'Bestimmung der Koordinaten des Maus-Klickens auf dem Profilschnitt
vPt = ListofFLPt.Get(vIdx)
vPtx = vPt.Getx
vPty = vPt.Gety
nPt = ListofFLPt.Get(nIdx)
nPtx = nPt.Getx
nPty = nPt.Gety

if ((minDist = 0) or (minDist < 10)) then 'Der näheste Punkt liegt innerhalb
   mPt = ListofFLPt.Get(minIdx)
   Ptx = mPt.Getx    '10 m vom dem Maus-Klicken auf der Linie
   Pty = mPt.Gety
   if ( minIdx = 0) then
     vGrIdx = minIdx
     nGrIdx = minIdx + 1
   elseif (( minIdx > 0) and ( minIdx < PtAnzIdx)) then
     vGrIdx = minIdx -1
     nGrIdx = minIdx + 1
   elseif ( minIdx = PtAnzIdx) then
     vGrIdx = minIdx -1
     nGrIdx = minIdx
   end
elseif (minDist >= 10) then  'Berechnung der Koordinaten des Punktes
   vGrIdx = vIdx      'auf der Linie als Projektion des Maus-Klickens
   nGrIdx = nIdx          'auf die Linie (Profilschnitt)
```

```
      if  (vPtx  = aMPtx) then
          Ptx = vPtx
          Pty = vPty
      elseif ((vPtx  <> aMPtx) and (aMPtx <> nPtx)) then
          if (vPty = nPty) then
                Ptx = aMPtx
                Pty = vPty
          elseif (vPty < nPty) then
                if (nPtx = vPtx) then
                      Ptx = nPtx
                      Pty = nPty
                elseif (nPtx <> vPtx) then
                      xnvDiff = nPtx - vPtx
                      ynvDiff = nPty - vPty
                      xMvDiff = aMPtx - vPtx
                      Ptx = aMPtx
                      Pty  = (ynvDiff / xnvDiff) * xMvDiff + vPty
                end
          elseif (vPty > nPty) then
                if (nPtx = vPtx) then
                      Ptx = nPtx
                      Pty = nPty
                elseif (nPtx <> vPtx) then
                      xnvDiff = nPtx - vPtx
                      yvnDiff = vPty - nPty
                      xnMDiff = nPtx - aMPtx
                      Ptx = aMPtx
                      Pty  = (yvnDiff / xnvDiff) * xnMDiff + nPty
                end
          end
      elseif (nPtx  = aMPtx)  then
          Ptx = nPtx
          Pty = nPty
      end
    end          ' Ende von (if ((minDist = 0) or (minDist < 10)) then)
    ListofGrPt.Add(Ptx@Pty)       'Ein Grenz-Punkt für den Abschnitt

    ListofGrIdx = {}
    ListofGrIdx.Add(vGrIdx)
    ListofGrIdx.Add(nGrIdx)
    ListofListofGrIdx.Add(ListofGrIdx)

    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/4*100)
    if (not more) then
      break
    end
  end          ' Ende der Schleife für Abschnitt
end     'Ende für (for each j in 0..1  'Schleife für Themen)


'Neues Polygon oder neue Polyline des Datensatzes wird
'durch Ersetzen des Abschnittes hergestellt.
ListofNT = {}
ListofFLPt00 = ListofListofFLPt.Get(0)  ' Liste der Punkte
AnzFLPt00 = ListofFLPt00.Count          ' zur Korrektur des Abschnittes
IdxFLPt00 = AnzFLPt00 - 1

ListofFLPt11 = ListofListofFLPt.Get(1)  ' Liste der Punkte
AnzFLPt11 = ListofFLPt11.Count          ' zur Übernahme des Abschnittes
IdxFLPt11 = AnzFLPt11 - 1
```

```
aShpCStr00 = ListofaShpClassStr.Get(0)
aShpCStr11 = ListofaShpClassStr.Get(1)

ListofTPt1100 = {}   'Teilliste der Polyline oder des Polygons
ListofTPt2200 = {}
ListofTPt3300 = {}
ListofTPt1111 = {}
ListofTPt2211 = {}
ListofTPt3311 = {}

if ((aShpCStr00 = "Polyline") and (aShpCStr11 = "Polyline")) then
  AnfFLPt00 = ListofGrPt.Get(0)
  EndFLPt00 = ListofGrPt.Get(1)
  AnfFLPt11 = ListofGrPt.Get(2)
  EndFLPt11 = ListofGrPt.Get(3)

  ListofGrIdxAnf00 = ListofListofGrIdx.Get(0)
  ListofGrIdxEnd00 = ListofListofGrIdx.Get(1)
  ListofGrIdxAnf11 = ListofListofGrIdx.Get(2)
  ListofGrIdxEnd11 = ListofListofGrIdx.Get(3)

elseif (((aShpCStr00 = "Polyline") and (aShpCStr11 = "Polygon")) or
     ((aShpCStr00 = "Polygon") and (aShpCStr11 = "Polyline"))) then

  AnfFLPt00 = ListofGrPt.Get(0)
  MitFLPt00 = ListofGrPt.Get(1)
  EndFLPt00 = ListofGrPt.Get(2)
  AnfFLPt11 = ListofGrPt.Get(3)
  MitFLPt11 = ListofGrPt.Get(4)
  EndFLPt11 = ListofGrPt.Get(5)

  ListofGrIdxAnf00 = ListofListofGrIdx.Get(0)
  ListofGrIdxEnd00 = ListofListofGrIdx.Get(2)
  ListofGrIdxAnf11 = ListofListofGrIdx.Get(3)
  ListofGrIdxEnd11 = ListofListofGrIdx.Get(5)

elseif ((aShpCStr00 = "Polygon") and (aShpCStr11 = "Polygon")) then
  AnfFLPt00 = ListofGrPt.Get(0)
  MitFLPt00 = ListofGrPt.Get(1)
  EndFLPt00 = ListofGrPt.Get(2)
  Mit2FLPt00 = ListofGrPt.Get(3)

  AnfFLPt11 = ListofGrPt.Get(4)
  MitFLPt11 = ListofGrPt.Get(5)
  EndFLPt11 = ListofGrPt.Get(6)
  Mit2FLPt11 = ListofGrPt.Get(7)

  ListofGrIdxAnf00 = ListofListofGrIdx.Get(0)
  ListofGrIdxEnd00 = ListofListofGrIdx.Get(2)
  ListofGrIdxAnf11 = ListofListofGrIdx.Get(4)
  ListofGrIdxEnd11 = ListofListofGrIdx.Get(6)
end

vGrIdxAnf00 = ListofGrIdxAnf00.Get(0)
nGrIdxAnf00 = ListofGrIdxAnf00.Get(1)

vGrIdxEnd00 = ListofGrIdxEnd00.Get(0)
nGrIdxEnd00 = ListofGrIdxEnd00.Get(1)

vGrIdxAnf11 = ListofGrIdxAnf11.Get(0)
nGrIdxAnf11 = ListofGrIdxAnf11.Get(1)
```

```
vGrIdxEnd11 = ListofGrIdxEnd11.Get(0)
nGrIdxEnd11 = ListofGrIdxEnd11.Get(1)

if (vGrIdxAnf00 < vGrIdxEnd00) then
  vKlIdx00 = vGrIdxAnf00
  vgrIdx00 = vGrIdxEnd00
  nKlIdx00 = nGrIdxAnf00
  ngrIdx00 = nGrIdxEnd00
  klPt00 = AnfFLPt00
  grPt00 = EndFLPt00
elseif (vGrIdxAnf00 > vGrIdxEnd00) then
  vKlIdx00 = vGrIdxEnd00
  vgrIdx00 = vGrIdxAnf00
  nKlIdx00 = nGrIdxEnd00
  ngrIdx00 = nGrIdxAnf00
  klPt00 = EndFLPt00
  grPt00 = AnfFLPt00
end

if (vGrIdxAnf11 < vGrIdxEnd11) then
  vKlIdx11 = vGrIdxAnf11
  vgrIdx11 = vGrIdxEnd11
  nKlIdx11 = nGrIdxAnf11
  ngrIdx11 = nGrIdxEnd11
  klPt11 = AnfFLPt11
  grPt11 = EndFLPt11
elseif (vGrIdxAnf11 > vGrIdxEnd11) then
  vKlIdx11 = vGrIdxEnd11
  vgrIdx11 = vGrIdxAnf11
  nKlIdx11 = nGrIdxEnd11
  ngrIdx11 = nGrIdxAnf11
  klPt11 = EndFLPt11
  grPt11 = AnfFLPt11
end

for each i in 0..IdxFLPt00
  aPt = ListofFLPt00.Get(i)
  if (i <= vKlIdx00) then
    ListofTPt1100.Add(aPt)
  elseif ((i >= nKlIdx00) and (i <= vgrIdx00)) then
    ListofTPt2200.Add(aPt)
  elseif (i >= ngrIdx00) then
    ListofTPt3300.Add(aPt)
  end
end

for each i in 0..IdxFLPt11
  aPt = ListofFLPt11.Get(i)
  if (i <= vKlIdx11) then
    ListofTPt1111.Add(aPt)
  elseif ((i >= nKlIdx11) and (i <= vgrIdx11)) then
    ListofTPt2211.Add(aPt)
  elseif (i >= ngrIdx11) then
    ListofTPt3311.Add(aPt)
  end
end

ListofListofTPt1100 = {}
ListofListofTPt2200 = {}
ListofListofTPt3300 = {}

ListofListofTPt1100.Add(ListofTPt1100)
```

```
ListofListofTPt2200.Add(ListofTPt2200)
ListofListofTPt3300.Add(ListofTPt3300)

aPL1100 = Polyline.Make(ListofListofTPt1100)
aPL2200 = Polyline.Make(ListofListofTPt2200)
aPL3300 = Polyline.Make(ListofListofTPt3300)

ListofListofTPt1111 = {}
ListofListofTPt2211 = {}
ListofListofTPt3311 = {}

ListofListofTPt1111.Add(ListofTPt1111)
ListofListofTPt2211.Add(ListofTPt2211)
ListofListofTPt3311.Add(ListofTPt3311)

aPL1111 = Polyline.Make(ListofListofTPt1111)
aPL2211 = Polyline.Make(ListofListofTPt2211)
aPL3311 = Polyline.Make(ListofListofTPt3311)

ListofTListe = {ListofTPt1100, ListofTPt2200, ListofTPt3300,
            ListofTPt1111, ListofTPt2211, ListofTPt3311}
ListofKW = {"11","12","13","21","22","23"}
ListofTPL = {aPL1100,aPL2200,aPL3300,aPL1111,aPL2211,aPL3311}


if ((aShpCStr00 = "Polyline") and (aShpCStr11 = "Polyline")) then

  AnzTPt2211 = ListofTPt2211.Count
  IdxTPt2211 = AnzTPt2211 - 1

  'Feststellung der Reihenfolge
  minklIdx = 1000000
  minAbst = 1000000
  for each i in 0..IdxTPt2211
    aPt = ListofTPt2211.Get(i)
    aPtx = aPt.Getx
    aPty = aPt.Gety
    klPt00x = klPt00.Getx
    klPt00y = klPt00.Gety
    Abst = (((aPtx - klPt00x) ^ 2) + ((aPty - klPt00y) ^ 2)).Sqrt.Abs
    if (Abst < minAbst) then
      minAbst = Abst
      minklIdx = i
    end
  end

  'Herstellung einer neuen Polyline (zum Teil ersetzt)
  for each i in 0..IdxFLPt00
    if (i <= vKlIdx00) then
      aPt = ListofFLPt00.Get(i)
      ListofNT.Add(aPt)
    end
  end
  ListofNT.Add(klPt00)

  if (minklIdx = 0) then
    for each i in 0..IdxFLPt11
      if ((i >= nKlIdx11) and (i <= vgrIdx11)) then
        aPt = ListofFLPt11.Get(i)
        ListofNT.Add(aPt)
      end
    end
```

```
  elseif (minklIdx = IdxTPt2211) then
    Ng = -1
    for each i in 0..IdxFLPt11
      if ((i >= nKlIdx11) and (i <= vgrIdx11)) then
        Ng = Ng + 1
        umkIdx = vgrIdx11 - Ng
        aPt = ListofFLPt11.Get(umkIdx)
        ListofNT.Add(aPt)
      end
    end
  end
  ListofNT.Add(grPt00)
  for each i in 0..IdxFLPt00
    if (i >= ngrIdx00) then
      aPt = ListofFLPt00.Get(i)
      ListofNT.Add(aPt)
    end
  end

elseif ((aShpCStr00 = "Polyline") and (aShpCStr11 = "Polygon")) then

  'Auswahl einer Teilliste des Polygons, um Daten zu übernehmen
  aCircle2M = Circle.Make(MitFLPt11, 5)

  Test11 = aCircle2M.Intersects(aPL1111)
  Test22 = aCircle2M.Intersects(aPL2211)
  Test33 = aCircle2M.Intersects(aPL3311)

  'MsgBox.Report("Test11:"++"  "++Test11.AsString+NL+
  '              "Test22:"++"  "++Test22.AsString+NL+
  '              "Test33:"++"  "++Test33.AsString,
  '              "Ausgewählte Teilliste des Polygons")

  'Herstellung einer neuen Polyline (zum Teil ersetzt)
  for each i in 0..IdxFLPt00
    if (i <= vKlIdx00) then
      aPt = ListofFLPt00.Get(i)
      ListofNT.Add(aPt)
    end
  end
  ListofNT.Add(klPt00)

  if ((Test11) or (Test33)) then
    AnzTPt3311 = ListofTPt3311.Count
    IdxTPt3311 = AnzTPt3311 - 1

    'Feststellung der Reihenfolge
    minklIdx = 1000000
    minAbst = 1000000
    for each i in 0..IdxTPt3311
      aPt = ListofTPt3311.Get(i)
      aPtx = aPt.Getx
      aPty = aPt.Gety
      klPt00x = klPt00.Getx
      klPt00y = klPt00.Gety
      Abst = (((aPtx - klPt00x) ^ 2) + ((aPty - klPt00y) ^ 2)).Sqrt.Abs
      if (Abst < minAbst) then
        minAbst = Abst
        minklIdx = i
      end
    end
```

```
  if (minklIdx = 0) then
    for each i in 0..IdxFLPt11
      if (i >= ngrIdx11) then
        aPt = ListofFLPt11.Get(i)
        ListofNT.Add(aPt)
      end
    end
    for each i in 0..IdxFLPt11
      if (i <= vklIdx11) then
        aPt = ListofFLPt11.Get(i)
        ListofNT.Add(aPt)
      end
    end

  elseif (minklIdx = IdxTPt3311) then
    Ng = -1
    for each i in 0..IdxFLPt11
      if (i <= vklIdx11) then
        Ng = Ng + 1
        umkIdx = vklIdx11 - Ng
        aPt = ListofFLPt11.Get(umkIdx)
        ListofNT.Add(aPt)
      end
    end
    Ng = -1
    for each i in 0..IdxFLPt11
      if (i >= ngrIdx11) then
        Ng = Ng + 1
        umkIdx = IdxFLPt11 - Ng
        aPt = ListofFLPt11.Get(umkIdx)
        ListofNT.Add(aPt)
      end
    end
  end

elseif (Test22) then
  AnzTPt2211 = ListofTPt2211.Count
  IdxTPt2211 = AnzTPt2211 - 1

  'Feststellung der Reihenfolge
  minklIdx = 1000000
  minAbst = 1000000
  for each i in 0..IdxTPt2211
    aPt = ListofTPt2211.Get(i)
    aPtx = aPt.Getx
    aPty = aPt.Gety
    klPt00x = klPt00.Getx
    klPt00y = klPt00.Gety
    Abst = (((aPtx - klPt00x) ^ 2) + ((aPty - klPt00y) ^ 2)).Sqrt.Abs
    if (Abst < minAbst) then
      minAbst = Abst
      minklIdx = i
    end
  end

  if (minklIdx = 0) then
    for each i in 0..IdxFLPt11
      if ((i >= nKlIdx11) and (i <= vgrIdx11)) then
        aPt = ListofFLPt11.Get(i)
        ListofNT.Add(aPt)
      end
    end
```

```
    elseif (minklIdx = IdxTPt2211) then
      Ng = -1
      for each i in 0..IdxFLPt11
        if ((i >= nKlIdx11) and (i <= vgrIdx11)) then
          Ng = Ng + 1
          umkIdx = vgrIdx11 - Ng
          aPt = ListofFLPt11.Get(umkIdx)
          ListofNT.Add(aPt)
        end
      end
    end
  end

  ListofNT.Add(grPt00)
  for each i in 0..IdxFLPt00
    if (i >= ngrIdx00) then
      aPt = ListofFLPt00.Get(i)
      ListofNT.Add(aPt)
    end
  end

elseif ((aShpCStr00 = "Polygon") and (aShpCStr11 = "Polyline")) then

  'Auswahl einer Teilliste des Polygons, um Daten zu übernehmen
  aCircle2M = Circle.Make(MitFLPt00, 5)

  Test11 = aCircle2M.Intersects(aPL1100)
  Test22 = aCircle2M.Intersects(aPL2200)
  Test33 = aCircle2M.Intersects(aPL3300)

  'MsgBox.Report("Test11:"++"  "++Test11.AsString+NL+
  '              "Test22:"++"  "++Test22.AsString+NL+
  '              "Test33:"++"  "++Test33.AsString,
  '              "Ausgewählte Teilliste des Polygons")

  'Herstellung einer neuen Polyline (zum Teil ersetzt)

  if ((Test11) or (Test33)) then
    for each i in 0..IdxFLPt00
      if (i <= vKlIdx00) then
        aPt = ListofFLPt00.Get(i)
        ListofNT.Add(aPt)
      end
    end
    ListofNT.Add(klPt00)

    AnzTPt2211 = ListofTPt2211.Count
    IdxTPt2211 = AnzTPt2211 - 1

    'Feststellung der Reihenfolge
    minklIdx = 1000000
    minAbst = 1000000
    for each i in 0..IdxTPt2211
      aPt = ListofTPt2211.Get(i)
      aPtx = aPt.Getx
      aPty = aPt.Gety
      klPt00x = klPt00.Getx
      klPt00y = klPt00.Gety
      Abst = (((aPtx - klPt00x) ^ 2) + ((aPty - klPt00y) ^ 2)).Sqrt.Abs
      if (Abst < minAbst) then
        minAbst = Abst
        minklIdx = i
```

```
    end
  end

  if (minklIdx = 0) then
    for each i in 0..IdxFLPt11
      if ((i >= nklIdx11) and (i <= vgrIdx11)) then
        aPt = ListofFLPt11.Get(i)
        ListofNT.Add(aPt)
      end
    end
  elseif (minklIdx = IdxTPt2211) then
    Ng = -1
    for each i in 0..IdxFLPt11
      if ((i >= nklIdx11) and (i <= vgrIdx11)) then
        Ng = Ng + 1
        umkIdx = vgrIdx11 - Ng
        aPt = ListofFLPt11.Get(umkIdx)
        ListofNT.Add(aPt)
      end
    end
  end
  ListofNT.Add(grPt00)

  for each i in 0..IdxFLPt00
    if (i >= ngrIdx00) then
      aPt = ListofFLPt00.Get(i)
      ListofNT.Add(aPt)
    end
  end
elseif (Test22) then
  ListofNT.Add(klPt00)
  for each i in 0..IdxFLPt00
    if ((i >= nKlIdx00) and (i <= vgrIdx00)) then
      aPt = ListofFLPt00.Get(i)
      ListofNT.Add(aPt)
    end
  end
  ListofNT.Add(grPt00)

  AnzTPt2211 = ListofTPt2211.Count
  IdxTPt2211 = AnzTPt2211 - 1

  'Feststellung der Reihenfolge
  minklIdx = 1000000
  minAbst = 1000000
  for each i in 0..IdxTPt2211
    aPt = ListofTPt2211.Get(i)
    aPtx = aPt.Getx
    aPty = aPt.Gety
    grPt00x = grPt00.Getx
    grPt00y = grPt00.Gety
    Abst = (((aPtx - grPt00x) ^ 2) + ((aPty - grPt00y) ^ 2)).Sqrt.Abs
    if (Abst < minAbst) then
      minAbst = Abst
      minklIdx = i
    end
  end

  if (minklIdx = 0) then
    for each i in 0..IdxFLPt11
      if ((i >= nklIdx11) and (i <= vgrIdx11)) then
        aPt = ListofFLPt11.Get(i)
```

```
            ListofNT.Add(aPt)
          end
        end
      elseif (minklIdx = IdxTPt2211) then
        Ng = -1
        for each i in 0..IdxTPt2211
          Ng = Ng + 1
          umkIdx = IdxTPt2211 - Ng
          aPt = ListofTPt2211.Get(umkIdx)
          ListofNT.Add(aPt)
        end
      end
    end

elseif ((aShpCStr00 = "Polygon") and (aShpCStr11 = "Polygon")) then


    'Feststellung der richtigen Wahl-Punkte an Maus-Klicken
    MP1 = ListofMPt.Get(1)
    MP2 = ListofMPt.Get(3)

    ListofMP = {MP1, MP2}
    ListofMitPt = {MitFLPt00,Mit2FLPt00,MitFLPt11,Mit2FLPt11}
    ListofRWP = {}

    for each j in 0..1
      aMt = ListofMP.Get(j)
      aMtx = aMt.Getx
      aMty = aMt.Gety
      minAbst = 1000000
      minIdx = -1
      for each i in 0..3
        aMitPt = ListofMitPt.Get(i)
        aWPx = aMitPt.Getx
        aWPy = aMitPt.Gety
        aAbst = (((aWPx - aMtx) ^ 2) + ((aWPy - aMty) ^ 2)).Sqrt.Abs
        if (aAbst < minAbst) then
          minAbst = aAbst
          minIdx = i
        end
      end
      aRWP = ListofMitPt.Get(minIdx)
      ListofRWP.Add(aRWP)
    end
    a1MP = ListofRWP.Get(0)
    a2MP = ListofRWP.Get(1)

    'Auswahl einer Teilliste der Polygone
    ListofIdxList = {}

    for each j in 0..1
      aRWP = ListofRWP.Get(j)
      aWPCircle = Circle.Make(aRWP, 5)
      for each i in 0..5
        aPL = ListofTPL.Get(i)
        aTest = aWPCircle.Intersects(aPL)
        if (aTest) then
          IdxList = i
        end
      end
      ListofIdxList.Add(IdxList)
    end
```

```
aIdxW0 = ListofIdxList.Get(0)
aIdxW1 = ListofIdxList.Get(1)
KW1 = ListofKW.Get(aIdxW0)
KW2 = ListofKW.Get(aIdxW1)
Fall = ""

'MsgBox.Report("ausgewählte Teilliste 1:"++"  "++KW1+NL+
'             "ausgewählte Teilliste 2:"++"  "++KW2,
'             "Kontrolle (Kennwort)")

if ((KW1 = "11") or (KW1 = "13")) then
  if (KW2 = "12") then
    Fall = "a"
  elseif ((KW2 = "21") or (KW2 = "23")) then
    Fall = "e"
  elseif (KW2 = "22") then
    Fall = "c"
  end
elseif (KW1 = "12") then
  if ((KW2 = "11") or (KW2 = "13")) then
    Fall = "a"
  elseif ((KW2 = "21") or (KW2 = "23")) then
    Fall = "d"
  elseif (KW2 = "22") then
    Fall = "f"
  end
elseif ((KW1 = "21") or (KW1 = "23")) then
  if ((KW2 = "11") or (KW2 = "13")) then
    Fall = "e"
  elseif (KW2 = "12") then
    Fall = "d"
  elseif (KW2 = "22") then
    Fall = "b"
  end
elseif (KW1 = "22") then
  if ((KW2 = "11") or (KW2 = "13")) then
    Fall = "c"
  elseif (KW2 = "12") then
    Fall = "f"
  elseif ((KW2 = "21") or (KW2 = "23")) then
    Fall = "b"
  end
end

ListofNT = {}

if (Fall = "a") then
  MsgBox.Info("Das erste Polygon wird nicht korrigiert.",
        "Information")
elseif (Fall = "b") then
  for each i in 0..IdxFLPt11
    aPt = ListofFLPt11.Get(i)
    ListofNT.Add(aPt)
  end
elseif (Fall = "c") then
  for each i in 0..IdxFLPt00
    if (i <= vKlIdx00) then
      aPt = ListofFLPt00.Get(i)
      ListofNT.Add(aPt)
    end
  end
```

```
    ListofNT.Add(klPt00)

    for each i in 0..IdxFLPt11
      if ((i >= nKlIdx11) and (i <= vgrIdx11)) then
        aPt = ListofFLPt11.Get(i)
        ListofNT.Add(aPt)
      end
    end

    ListofNT.Add(grPt00)
    for each i in 0..IdxFLPt00
      if (i >= ngrIdx00) then
        aPt = ListofFLPt00.Get(i)
        ListofNT.Add(aPt)
      end
    end

  elseif (Fall = "d") then
    for each i in 0..IdxFLPt11
      if (i <= vKlIdx11) then
        aPt = ListofFLPt11.Get(i)
        ListofNT.Add(aPt)
      end
    end
    ListofNT.Add(klPt00)

    for each i in 0..IdxFLPt00
      if ((i >= nKlIdx00) and (i <= vgrIdx00)) then
        aPt = ListofFLPt00.Get(i)
        ListofNT.Add(aPt)
      end
    end

    ListofNT.Add(grPt00)
    for each i in 0..IdxFLPt11
      if (i >= ngrIdx11) then
        aPt = ListofFLPt11.Get(i)
        ListofNT.Add(aPt)
      end
    end

  elseif (Fall = "e") then
    for each i in 0..IdxFLPt00
      if (i <= vKlIdx00) then
        aPt = ListofFLPt00.Get(i)
        ListofNT.Add(aPt)
      end
    end
    ListofNT.Add(klPt00)
    Ng = -1
    for each i in 0..IdxFLPt11
      if (i <= vklIdx11) then
        Ng = Ng + 1
        umkIdx = vklIdx11 - Ng
        aPt = ListofFLPt11.Get(umkIdx)
        ListofNT.Add(aPt)
      end
    end
    Ng = -1
    for each i in 0..IdxFLPt11
      if (i >= ngrIdx11) then
        Ng = Ng + 1
```

```
          umkIdx = IdxFLPt11 - Ng
          aPt = ListofFLPt11.Get(umkIdx)
          ListofNT.Add(aPt)
        end
      end

      ListofNT.Add(grPt00)
      for each i in 0..IdxFLPt00
        if (i >= ngrIdx00) then
          aPt = ListofFLPt00.Get(i)
          ListofNT.Add(aPt)
        end
      end
    elseif (Fall = "f") then
      ListofNT.Add(grPt00)
      Ng = -1
      for each i in 0..IdxFLPt00
        if ((i >= nKlIdx00) and (i <= vgrIdx00)) then
          Ng = Ng + 1
          umkIdx = vgrIdx00 - Ng
          aPt = ListofFLPt00.Get(umkIdx)
          ListofNT.Add(aPt)
        end
      end
      ListofNT.Add(klPt00)
      for each i in 0..IdxFLPt11
        if ((i >= nKlIdx11) and (i <= vgrIdx11)) then
          aPt = ListofFLPt11.Get(i)
          ListofNT.Add(aPt)
        end
      end
    end
end

av.ShowMsg("Speicherung der neuen 2D-Profilschnitte in einem Thema ...")

AnzNPt = ListofNT.Count
if (AnzNPt <> 0) then
  ListofListofNPoint={}
  ListofListofNPoint.Add(ListofNT)
  aShpClassStr = ListofaShpClassStr.Get(0)

  if (aShpClassStr = "Polyline") then
    theNShp=PolyLine.Make(ListofListofNPoint)
  elseif (aShpClassStr = "Polygon") then
    theNShp=Polygon.Make(ListofListofNPoint)
  end

  theFTab = AkTheme.GetFTab
  theShpFld = theFTab.FindField("Shape")
  RecNr = ListofRecIdx.Get(0)
  theFTab.SetEditable(false)
  theFTab.SetEditable(true)
  theFTab.SetValue(theShpFld, recNr, theNShp)
  theFTab.SetEditable(false)

elseif (AnzNPt = 0) then
  MsgBox.Report("Die Polyline oder das Polygon wurde nicht korrigert!"
        +NL+"Die Reihenfolge der Maus-Klicken waren wahrscheinlich"
          ++"nicht so, wie vorgesehen."+NL+
          "Das Programm wird abgebrochen!", "Information")
end
```

```
av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

'Definition der Farbe der Legende
'Bezeichnung der Datensätze
aListofgB={"---","a","a/Mj","Aussen","d","f","Gy","H",
    "hg/plRR","Hj","Hn","Lf","Lfh/N","Lö","Lö/Hj","Lö/Mj",
    "Löy","Ma","mi-olK","miIV","Mj","N","plRR","Rhein",
    "Sf","Sfh/N","sSo","tAb","tTt","Sonst",
    "Deck","D","De","NT","MT","HT","Präq","Präm",
    "GH","NA","NQ","NT",
    "Deck","QmitD","QohneD","QohneT","TrorAe",
    "Deckschichten","Niederterrassen","Mittelterrassen",
    "Präquartäre Schichten",
    "rMTl", "lMTl", "rHTl", "lNT", "rNT", "lHTr", "rHTr",
    "HTr", "MTl", "MTr", "HTl"}
'Die Nummer der Farbe im Symbolwindow
aListofNr={20,43,24,5,36,8,24,32,3,32,11,17,14,47,30,
         24,6,29,35,11,26,16,53,20,18,16,24,53,51,54,
         53,53,53,14,26,32,3,3,53,5,8,14,53,20,26,8,3,
         53,14,26,3,20,26,32,14,17,32,32,32,26,20,32}
Legendeinfo = 0

for each j in 0..1
  theTheme = ListofThemen.Get(j)
  aorgLTyp = ListofLegendTyp.Get(j)
  if (aorgLTyp = #LEGEND_TYPE_SIMPLE) then
    ListoforgColor = ListofListoforgColor.Get(j)
    theLegend = theTheme.GetLegend
    theLegend.SetLegendType(#LEGEND_TYPE_SIMPLE)
    theLegend.SingleSymbol
    theSymbol=theLegend.GetSymbols.Get(0)
    theSymbol.SetColor(ListoforgColor.Get(0))
  else
    aSchtFld = ListofField.Get(j)
    aSchFldStr = aSchtFld.AsString

    av.ShowMsg("Veränderung der Legenden des Themas"
      ++theTheme.AsString+"...")

    theLegend=theTheme.GetLegend
    theLegend.SetLegendType(#Legend_Type_Unique)
    theLegend.Unique(theTheme, aSchFldStr)
    ListofKlasse=theLegend.GetClassifications
    AnzKlasse=ListofKlasse.Count
    IdxKlasse=AnzKlasse-2
    'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")

    aListofSColor = {}
    for each i in 0..IdxKlasse
      theKlasseLb=ListofKlasse.Get(i).GetLabel
      'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
      aIdxLb = aListofgB.FindByValue(theKlasseLb)
      if (aIdxLb <> -1) then
        aCNr = aListofNr.Get(aIdxLb)
      elseif (aIdxLb = -1) then
        if (i < 60) then
          aCNr = i
        elseif (i > 59) then
          aCNr = i - 59
        end
```

```
        Legendeinfo = 1
      end
      theColor=(thePalette.GetList(#PALETTE_LIST_COLOR).Get(aCNr))
      theRgbList=theColor.GetRgbList
      aColor=Color.Make
      aColor.SetRgbList(theRgbList)
      aListofSColor.Add(aColor)
    end

    aListofSymbol=theLegend.GetSymbols
    AnzSymb=aListofSymbol.Count
    AnzSymbIdx=AnzSymb-2

    for each symb in 0..AnzSymbIdx
      aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
    end
    if (Legendeinfo = 1) then
       MsgBox.Report("Die Farbe der Legende ist zum Teil"
         +NL+"oder gar nicht definiert!"
         +NL+"Die Definition der Farbe"
         +NL+"in diesem Programm ist zu ändern.",
       "Information")
    end
  end
end
theTheme.UpdateLegend




'pglkomfk.ave
'Ein Punkt am Fadenkreuz wird auf einem Profilschnitt eines aktiven
'Themas eingesetzt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc    'Ein aktives View
theTheme=theView.GetActiveThemes.Get(0) 'Ein aktives Thema
myScript=theProject.FindScript("pglkomfk")
myScript.SetNumberFormat( "d.dd") ' script default

ThStr=theTheme.AsString

kt00=MsgBox.YesNo("Ist das aktive Thema"++ThStr
     +NL+"zur Korrektur richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
            "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

ListofThms=theView.GetThemes
ListofPLThms = {}

for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aClNm = aFTab.GetShapeClass.GetClassName
      if (aClNm = "PolyLine") then
        ListofPLThms.Add(aT)
```

```
      end
    end
end
KrTheme = MsgBox.ChoiceAsString(ListofPLThms,
  "um es auf einem Profilschnitt einzusetzen",
  "Auswahl eines Fadenkreuzes")

KrFTab=KrTheme.GetFTab
KrShpFld=KrFTab.FindField("Shape")
KrIdFld=KrFTab.FindField("Id")
AnzRec=0
for each rec in KrFTab
   AnzRec=AnzRec+1
end
AnzRecIdx=AnzRec-1
'MsgBox.Info(AnzRec.AsString, "Anzahl der Datensätze im Thema"
'     ++KrTheme.AsString)
IdNr=1
PtNr=0
ListofKrPt={}
for each rec in 0..AnzRecIdx
  aPL=KrFTab.ReturnValue(KrShpFld, rec)
  ListofKrPL={}
  ListofKrPL.Add({aPL})
  for each Lst in ListofKrPL
    theLs=Lst.Get(0).AsList
    for each L in theLs
      for each ptx in L
        PtNr=(PtNr+1).SetFormat("").SetFormat("d")
        ListofKrPt.Add(ptx)
       'MsgBox.Report("Der"++PtNr.AsString++"."++"Punkt:"++ptx.AsString,
       '             "Die Koordinaten der Punkte an dem Fadenkreuz")
      end
    end
  end
end

PtKr0 = ListofKrPt.Get(0)
PtKrx0 = PtKr0.Getx

Anzx0 = 0
for each i in 0..3
  aPtKr = ListofKrPt.Get(i)
  axkrd = aPtKr.Getx
  if (axkrd = PtKrx0) then
    Anzx0 = Anzx0 + 1
  end
end

if (Anzx0 = 1) then
  PtKr0 = ListofKrPt.Get(0)
  yl = PtKr0.Gety
  PtKr2 = ListofKrPt.Get(2)
  xob = PtKr2.Getx
elseif (Anzx0 > 1) then
  PtKr0 = ListofKrPt.Get(0)
  xob = PtKr0.Getx
  PtKr2 = ListofKrPt.Get(2)
  yl = PtKr2.Gety
end

YFaktStr="50"
```

```
YFtStr=MsgBox.Input("zur Überhöhung der Höhen",
     "Eingabe eines Faktors", YFaktStr)
YFt=YFtStr.AsNumber
theKrPt=Point.Make(xob, yl)
theyH=yl/ YFt
'MsgBox.Report("Der Punkt am Fadenkreuz:"++theKrPt.AsString
'             +NL+"("+theyH.AsString++"[m ü NN])",
'             "Die Koordinaten der Punkte an dem Fadenkreuz")
PtAnfx=xob
PtAnfy=yl
'MsgBox.Info(PtAnfx.AsString, "X-Koordinate des Punktes")
'MsgBox.Info(PtAnfy.AsString++"("+theyH.AsString++"m ü NN)",
'     "Y-Koordinate des Punktes")
'AW1=MsgBox.YesNo("Sind die Daten richtig? ", "Kontrolle der Daten",
' TRUE)
'if (Not AW1) then
'   MsgBox.Error("Die Daten sind nicht richtig!"+NL+
'               "Das Fadenkreuz soll neu eingegeben werden!", "")
'   Exit
'end

FTab1=theTheme.GetFTab
ShpFld1=FTab1.FindField("Shape")
ListofFlds = FTab1.GetFields
FLFld1 = MsgBox.ChoiceAsString(ListofFlds,
       "das den geologischen Namen enthält",
       "Auswahl eines Feldes im Thema"++theTheme.AsString)
IdxofFLFld1 = ListofFlds.FindByValue(FLFld1)

'MsgBox.Info(theTheme.AsString, "Der Name des Profilschnittes")

'Feststellung der Anzahl der 2D-PolyLine in dem Thema
Anz2DPL=0
for each rec in FTab1
   Anz2DPL=Anz2DPL+1
end
Anz2DPLIdx=Anz2DPL-1

ListofFlaeche = {}
for each aF in 0..Anz2DPLIdx
   aFlaeche = FTab1.ReturnValue(FLFld1, aF)
   ListofFlaeche.Add(aFlaeche)
end

aGFL = MsgBox.ListAsString(ListofFlaeche,
     "um das Fadenkreuz einzusetzen",
     "Auswahl einer Fläche")
aIdx = ListofFlaeche.FindByValue(aGFL)
thePlShp = FTab1.ReturnValue(ShpFld1, aIdx) 'Auswahl einer Fläche

minx = 3000000
maxx = 0

av.ShowMsg("Feststellung der Koordinaten des ausgewählten"
       ++"2D-Profilschnittes"++(theTheme.AsString)++"...")

theProfilList1={}
theProfilList1.Add({thePlShp})

ListofFLx = {}     'Liste der Vertices der geologischen Fläche
ListofFLy = {}
ListofFLPt = {}
```

```
minxkrd = 100000000
minykrd = 100000000
maxxkrd = 0
maxykrd = 0

if (theProfilList1 <> 0) then
  for each q in theProfilList1
    theLines=q.Get(0).AsList
    for each m in theLines
      for each ptx in m
        myx=ptx.Getx
        myy=ptx.Gety
        ListofFLx.Add(myx)
        ListofFLy.Add(myy)
        ListofFLPt.Add(myx@myy)
        if (myx < minxkrd) then
          minxkrd = myx
        end
        if (myx > maxxkrd) then
          maxxkrd = myx
        end
        if (myy < minykrd) then
          minykrd = myy
        end
        if (myy > maxykrd) then
          maxykrd = myy
        end
      end
    end
  end
end

PtAnz= ListofFLx.Count  'Anzahl der Vertices der geologischen Fläche
PtAnzIdx= PtAnz-1

'Bestimmung der Stelle des Fadenkreuzes auf der geologischen Fläche

minDist=1000
minIdx=-1

aMPtx = xob
aMPty = yl

if (aMPtx < minxkrd) then
  aMPtx = minxkrd
elseif (aMPtx > maxxkrd) then
  aMPtx = maxxkrd
end
if (aMPty < minykrd) then
  aMPty = minykrd
elseif (aMPty > maxykrd) then
  aMPty = maxykrd
end

for each i in 0..PtAnzIdx
  xkrd= ListofFLx.Get(i)
  ykrd= ListofFLy.Get(i)
  xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
  yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
  xyAbst = ((xAbst  + yAbst).sqrt) .Abs
  if (xyAbst < minDist) then
    minDist = xyAbst
```

```
      minIdx = i  'Index des nähesten Punktes des Maus-Klickens
    end
end

'Bestimmung der Stelle auf der geologischen Fläche
'als x-, y-Koordinaten

if (minIdx = 0) then
    vIdx = 0     'ein Vertex auf der Linie vor dem Fadenkeuz
    nIdx = 1     'ein Vertex auf der Linie nach dem Fadenkreuz
elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
    xkrdv = ListofFLx.Get((minIdx -1))   'x-Koord. des letzten Punktes
    xkrd  = ListofFLx.Get(minIdx)   'x-Koord. des nähesten Punktes
    xkrdn = ListofFLx.Get((minIdx +1)) 'x-Koord. des nächsten Punktes

    ykrdv = ListofFLy.Get((minIdx -1)) 'y-Koord. des letzten Punktes
    ykrd = ListofFLy.Get(minIdx)       'y-Koord. des nähesten Punktes
    ykrdn = ListofFLy.Get((minIdx +1)) 'y-Koord. des nächsten Punktes

    xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
    yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
    xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten Punkt
                                   'vom nähesten Punkt
    xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
    yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
    xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten Punkt
                                   'vom nähesten Punkte
    xML = (xkrd - aMPtx) * (xkrd - aMPtx)
    yML = (ykrd - aMPty) * (ykrd - aMPty)
    xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Fadenkreuz
                                   'vom nähesten Punkt
    xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
    yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
    xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Fadenkreuz
                                      'vom letzten Punkt
    xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
    yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
    xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Fadenkreuz
                                     'vom nächsten Punkt
    vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem
    vLpML = xyAv + xyML        'Fadenkreuz und dem letzten Punkt

    nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen dem
    nLpML = xyAn + xyML        'Fadenkreuz und dem nächsten Punkt

    vM = (vLmML - xyMLv).Abs   'Differenz zwischen der theorischen
    vP = (vLpML - xyMLv).Abs   'und der tatsächlichen Länge im Bezug
                               'auf den letzten Punkt
    nM = (nLmML - xyMLn).Abs   'Differenz zwischen der theorischen
    nP = (nLpML - xyMLn).Abs   'und der tatsächlichen Länge im Bezug
                               'auf den nächsten Punkt
    ListofLg = {vM, vP, nM, nP}   'Suche nach der kleinsten Länge
    MinLg = 10000
    TheLgIdx = -1
    For each aLg in 0..3
        theLg = ListofLg.Get(aLg)
        if (theLg < MinLg) then
            MinLg = theLg
            TheLgIdx = aLg
        end
    end
    if (xyML < 10) then
```

```
      vIdx = minIdx - 1
      nIdx = minIdx + 1
    end
    if (xyML >= 10) then
        if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
            vIdx = minIdx - 1  'ein Vertex auf Linie vor dem Fadenkr
            nIdx= minIdx       'ein Vertex auf Linie nach dem Fadenkr
        elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
            vIdx = minIdx     'ein Vertex auf Linie vor dem Fadenkr
            nIdx = minIdx + 1 'ein Vertex auf Linie nach dem Fadenkr
        else
            vIdx = minIdx     'ein Vertex auf Linie vor dem Fadenkr
            nIdx = minIdx + 1 'ein Vertex auf Linie nach dem Fadenkr
        end
    end
elseif (minIdx = PtAnzIdx)   then
     vIdx = minIdx - 1
     nIdx = minIdx
end

ListofPoint = {}  'Einsetzen des Punktes am Fadenkeuz
                  'auf den Profilschnitt
for each i in 0..PtAnzIdx
  if (i <= vIdx) then
     ax = ListofFLx.Get(i)
     ay = ListofFLy.Get(i)
     ListofPoint.Add(ax@aY)
  end
end
ListofPoint.Add(xob@yl)
for each i in 0..PtAnzIdx
  if (i >= nIdx) then
     ax = ListofFLx.Get(i)
     ay = ListofFLy.Get(i)
     ListofPoint.Add(ax@aY)
  end
end
ListofListofPoint={}
ListofListofPoint.Add(ListofPoint)
thePolyLine=PolyLine.Make(ListofListofPoint)

av.ShowMsg("Speicherung des neuen 2D-Profilschnittes in das Thema"
       ++theTheme.AsString++"...")

recNr = aIdx
FTab1.SetEditable(false)
FTab1.SetEditable(true)
FTab1.SetValue(ShpFld1, recNr, thePolyLine)
FTab1.SetEditable(false)

LgdFld = FLFld1.AsString   'Feld für Schichten-Kennzeichnung
theLegend=theTheme.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(theTheme, LgdFld)

ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette
```

```
aListofColor={}
aListofNr={53, 5, 8, 20, 32, 26, 3}
'Reihenfolge der Klasse der Schichten in der Legende
aListofgB={"GH","NA","NQ","NT"}
'Reihenfolge der Farbe der Klasse
aLofF={"braun","schwarz","rot","blau",
      "Magantha","türkisch","grau"}

for each Nmb in 0..IdxKlasse
  if (Nmb <= 6) then
    aNumb=aListofNr.Get(Nmb)
  elseif ((Nmb > 6) and (Nmb < 60))then
    aNumb=Nmb
  elseif (Nmb > 59) then
    aNumb = Nmb - 59
  end
  theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
  theRgbList=theColor.GetRgbList
  aColor=Color.Make
  aColor.SetRgbList(theRgbList)
  aListofColor.Add(aColor)
end

aListofSColor = {}
for each i in 0..IdxKlasse
  theKlasseLb=ListofKlasse.Get(i).GetLabel
  'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
  aIdxLb = aListofgB.FindByValue(theKlasseLb)
  if (aIdxLb <> -1) then
    theColorNr = aListofColor.Get(aIdxLb)
  elseif (aIdxLb = -1) then
    aR = i Mod 60
    if (aR < 3) then
      aCIdx = 2
    elseif (aR > 2) then
      aCIdx = aR
    end
    theColorNr = aListofColor.Get(aCIdx)
  end
  aListofSColor.Add(theColorNr)
end

aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
AnzSymbIdx=AnzSymb-2

for each symb in 0..AnzSymbIdx
   aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
end
theTheme.UpdateLegend




'pglkotm1.ave
'Eine Polyline oder ein Polygon wird durch einen Punkt korrigiert,
'der durch eine Tastatur-Eingabe, ein Fadenkreuz, einen Punkt in
'einem Punkt-Thema, einen Stützpunkt einer Polyline oder eines Polygons
'oder einen Punkt eines Polygons bestimmt wird.
'Dieses Script wird als ein Werkzeug (Tool) im aktiven View benutzt.
```

```
theProject=av.GetProject
myScript=theProject.FindScript("pglkotm1")
myScript.SetNumberFormat( "d.dd") ' script default

AkView = av.GetActiveDoc

'Einmaliges Klicken der Maus auf das Bildschirm in der Nähe eines
'Punktes eines Punkt-Themas in einem aktiven View,
'um den Punkt in einer Polyline oder in einem Polygon einzusetzen.

aMausPt = AkView.GetDisplay.ReturnUserPoint
aMPtx = aMausPt.Getx
aMPty = aMausPt.Gety

aCircle = Circle.Make(aMausPt, 50)
aMCPg = aCircle.AsPolygon

AkTheme = AkView.GetActiveThemes.Get(0)
AkStr = AkTheme.AsString

kt00=MsgBox.YesNo("Ist das aktive Thema"++AkStr
     +NL+"zur Korrektur richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++AkStr++"ist falsch!"+NL+
           "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

'Auswahl eines Punktes, um den in einer Polyline oder
'in einem Polygon einzusetzen
ListofThemes = AkView.GetThemes
ListofPtThms = {}
ListofPLThms = {}
ListofPgThms = {}

for each aT in ListofThemes
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     aSCl = aFTab.GetShapeClass.GetClassName
     if (aSCl = "Point") then
       ListofPtThms.Add(aT)
     elseif (aSCl = "PolyLine") then
       ListofPLThms.Add(aT)
     elseif (aSCl = "Polygon") then
       ListofPgThms.Add(aT)
     end
   end
end

ListofPtForm = {"ein Punkt einer Tastatur-Eingabe",
           "ein Punkt an einem Fadenkreuz",
           "ein Punkt in einem Punkt-Thema",
           "ein Stützpunkt auf einer Polyline",
           "ein Punkt auf einem Polygon"}

ErstPt = MsgBox.ListAsString(ListofPtForm,
      "um die Koordinaten zu übernehmen",
      "Auswahl eines Punktes am Maus-Klicken")

if (ErstPt = "ein Punkt einer Tastatur-Eingabe") then
  aEPtx = (aMausPt.Getx.SetFormat("").SetFormat("d.dd")).AsString
```

```
    aEPty = (aMausPt.Gety.SetFormat("").SetFormat("d.dd")).AsString
    aHxStr = MsgBox.Input("Eingabe der x-Koord. des Punktes",
          "Tastaur-Eingabe", aEPtx)
    aHyStr = MsgBox.Input("Eingabe der y-Koord. des Punktes"
           +NL+"(z.B. Höhe (Faktor * [m]))","Tastaur-Eingabe", aEPty)
    aHx = aHxStr.AsNumber
    aHy = aHyStr.AsNumber
    aNPt = Point.Make(aHx, aHy)

  elseif (ErstPt = "ein Punkt an einem Fadenkreuz") then

    KrThm = MsgBox.ListAsString(ListofPLThms,
          "um Koordinaten zu übernehmen",
          "Auswahl eines Fadenkreuz-Themas")
    KrFTab = KrThm.GetFTab
    KrShpFld=KrFTab.FindField("Shape")

    AnzRec = 0
    for each rec in KrFTab
       AnzRec = AnzRec+1
    end
    AnzRecIdx = AnzRec-1
    'MsgBox.Info(AnzRec.AsString,
    '   "Anzahl der Datensätze im Thema (Fadenkreuz)"++KrThm.AsString)
    ListofKrPtx = {}
    ListofKrPty = {}

    for each rec in 0..AnzRecIdx
       aPL=KrFTab.ReturnValue(KrShpFld, rec)
       ListofKrPL = {}
       ListofKrPL.Add({aPL})
       for each q in ListofKrPL
          theLs = q.Get(0).AsList
          for each L in theLs
             for each apt in L
                aptx = apt.Getx
                apty = apt.Gety
                ListofKrPtx.Add(aptx)
                ListofKrPty.Add(apty)
             end
          end
       end
    end

    AnzKrPtx = ListofKrPtx.Count
    IdxKrPtx = AnzKrPtx - 1

    for each j in 0..IdxKrPtx
       aKrx1 = ListofKrPtx.Get(j)
       aKry1 = ListofKrPty.Get(j)
       xZ = 0
       yZ = 0
       for each i in 0..IdxKrPtx
          aKrx2 = ListofKrPtx.Get(i)
          aKry2 = ListofKrPty.Get(i)
          if (aKrx1 = aKrx2) then
            xZ = xZ + 1
          end
          if (aKry1 = aKry2) then
            yZ = yZ + 1
          end
       end
```

```
      if (xZ = 2) then
        theX = aKrx1
      end
      if (yZ = 2) then
        theY = aKry1
      end
    end
    aNPt = Point.Make(theX, theY)

    'MsgBox.Report("Der Punkt am Fadenkreuz"++"("+KrThm.AsString+"):"
    '             ++aNPt.AsString,
    '             "Die Koordinaten der Punkte an dem Fadenkreuz")

  elseif (ErstPt = "ein Punkt in einem Punkt-Thema") then

    'Auswahl eines Punkt-Themas (evtl. Ereignis-Themas) im aktiven View

    PtTheme = MsgBox.ChoiceAsString(ListofPtThms,
        "um Koordinaten der Punkte zu übernehmen",
        "Auswahl eines Punkt-Themas im View"++AkView.AsString)
    if (PtTheme = nil) then
      MsgBox.Error("Der Name der Punkt-Datei fehlt!"+NL+
            "Das Programm wird abgebrochen!", "")
      exit
    end

    PtFTab = PtTheme.GetFTab
    ListofPtFld = PtFTab.GetFields
    PtShpFld=ListofPtFld.Get(0) 'erstes Shape-Feld beim Ereignisthema

    AnzPt=0
    for each Pt in PtFTab
      AnzPt=AnzPt+1
    end
    AnzPtIdx=AnzPt-1
    AnzPtStr = AnzPt.SetFormat("d").AsString
    'MsgBox.Info(AnzPtStr, "Die Anzahl der Punkte im Thema"
    '          ++PtTheme.AsString)

    'Die Bezeichnung der Punkte wird in der Tabelle gesucht.

    AnzPtFld = ListofPtFld.Count
    IdxPtFld = AnzPtFld - 1

    'Feststellung des ersten Datensatzes
    FldStr=""
    DtStr=""
    for each i in 0..IdxPtFld
      aFld=ListofPtFld.Get(i)
      aFldStr=aFld.GetName
      if (aFldStr <> "Shape") then
        FldStr = FldStr + aFldStr+"; "
        aValue=PtFTab.ReturnValue(aFld, 0)
        DtStr=DtStr+aValue.AsString+"; "
      end
    end

    MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
          +NL+FldStr+NL+NL+
          "Der erste Datensatz:"+NL+DtStr,
          "Information")
    aPtKWFld = MsgBox.ListAsString(ListofPtFld,
```

```
    "für Klassifizierung der Legende",
    "Auswahl eines Feldes des Themas"++PtTheme.AsString)
  aPtKWFldStr = aPtKWFld.AsString

  PtLegend = PtTheme.GetLegend
  PtLegend.SetLegendType(#Legend_Type_Unique)
  PtLegend.Unique(PtTheme, aPtKWFldStr)
  ListofPtKlasse = PtLegend.GetClassifications
  AnzPtKlasse = ListofPtKlasse.Count
  IdxPtKlasse = AnzPtKlasse-2
  'MsgBox.Info(AnzPtKlasse.AsString, "Anzahl der Klasse (Pt)")

  aListofPtBez = {}
  for each i in 0..IdxPtKlasse
    theKlasseLb=ListofPtKlasse.Get(i).GetLabel
    aListofPtBez.Add(theKlasseLb)
  end

  thePtBz=MsgBox.ListAsString(aListofPtBez,
    "zur Auswahl der Punkte",
    "Auswahl der Bezeichnung der Punkte im"++PtTheme.AsString)
  if (thePtBz = nil) then
    MsgBox.Error("Die Bezeichnung der Punkte oder"+NL+
          "der Punkt mit der Bezeichnung fehlt!"+NL+
          "Das Programm wird abgebrochen!", "")
    exit
  end

  'Der nächeste Punkt vom Maus-Klicken wird in der Tabelle
  'der Punkt-Datei im aktiven View gesucht.

  minAbst=1000000
  for each Pt in 0..AnzPtIdx
     aPtBz=PtFTab.ReturnValue(aPtKWFld, Pt)
     if (aPtBz = thePtBz) then
       thePt = PtFTab.ReturnValue(PtShpFld, Pt)
       thePtx = thePt.Getx
       thePty = thePt.Gety
       Abst=(((thePtx-aMPtx) ^ 2) + ((thePty-aMPty) ^ 2)).Sqrt.Abs
       if (Abst < minAbst) then
         PtIdx = Pt
         minAbst = Abst
       end
     end
  end
  thePt = PtFTab.ReturnValue(PtShpFld, PtIdx)
  theX = thePt.Getx
  theY = thePt.Gety
  'MsgBox.Report("X-Koordinate"++": "++theX.AsString+NL+
  '          "Y-Koordinate"++": "++theY.AsString,
  '          "Der Punkt in der Tabelle der Punkt-Datei")

 aNPt = Point.Make(theX, theY)

elseif ((ErstPt = "ein Stützpunkt auf einer Polyline") or
     (ErstPt = "ein Punkt auf einem Polygon")) then
  'Auswahl eines Polyline-Themas im aktiven View


  if (ErstPt = "ein Stützpunkt auf einer Polyline") then
    PLTheme = MsgBox.ChoiceAsString(ListofPLThms,
      "um Koordinaten eines Punktes zu übernehmen",
```

```
      "Auswahl eines Polyline-Themas im View"++AkView.AsString)
elseif (ErstPt = "ein Punkt auf einem Polygon") then
  PLTheme = MsgBox.ChoiceAsString(ListofPgThms,
    "um Koordinaten eines Punktes zu übernehmen",
    "Auswahl eines Polygon-Themas im View"++AkView.AsString)
  ListofPtPg = {"ein Stützpunkt am Maus-Klicken",
        "ein Stützpunkt mit der kleinsten x-Koord.",
        "ein Stützpunkt mit der größten x-Koord."}

  ZweitAw = MsgBox.ListAsString(ListofPtPg,
    "um die Koordinaten zu übernehmen",
    "Auswahl eines Punktes am Maus-Klicken")
end

aTStr = PLTheme.AsString

if (PLTheme = nil) then
  MsgBox.Error("Der Name der Polyline- oder Polygon-Datei fehlt!"
        +NL+"Das Programm wird abgebrochen!", "")
  exit
end

PLFTab = PLTheme.GetFTab
ListofPLFld = PLFTab.GetFields
PLShpFld=ListofPLFld.Get(0)

PLShpClassStr = PLFTab.GetShapeClass.GetClassName
if (PLShpClassStr = "Polyline") then
  Art = "der Polyline"
  Art2 = "die Polyline"
elseif (PLShpClassStr = "Polygon") then
  Art = "des Polygons"
  Art2 = "das Polygon"
end

AnzPL=0
for each PL in PLFTab
   AnzPL = AnzPL+1
end
AnzPLIdx = AnzPL - 1
AnzPLStr = AnzPL.SetFormat("d").AsString
'MsgBox.Info(AnzPLStr, "Die Anzahl"++Art++"im Thema"++aTStr)

'Die Bezeichnung der Polyline oder des Polygons wird
'in der Tabelle gesucht.

AnzPLFld = ListofPLFld.Count
IdxPLFld = AnzPLFld - 1

'Feststellung des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxPLFld
   aFld=ListofPLFld.Get(i)
   aFldStr=aFld.GetName
   if (aFldStr <> "Shape") then
      FldStr = FldStr + aFldStr+"; "
      aValue=PLFTab.ReturnValue(aFld, 0)
      DtStr=DtStr+aValue.AsString+"; "
   end
end
```

```
MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
        +NL+FldStr+NL+NL+
        "Der erste Datensatz:"+NL+DtStr,
        "Information")
aPLKWFld = MsgBox.ListAsString(ListofPLFld,
  "für Klassifizierung der Legende, um einen Punkt zu übernehmen",
  "Auswahl eines Feldes des Themas"++aTStr)
aPLKWFldStr = aPLKWFld.AsString

PLLegend = PLTheme.GetLegend
PLLegend.SetLegendType(#Legend_Type_Unique)
PLLegend.Unique(PLTheme, aPLKWFldStr)
ListofPLKlasse = PLLegend.GetClassifications
AnzPLKlasse = ListofPLKlasse.Count
IdxPLKlasse = AnzPLKlasse-2
'MsgBox.Info(AnzPLKlasse.AsString, "Anzahl der Klasse (PL)")

aListofPLKl = {}
for each i in 0..IdxPLKlasse
  theKlasseLb=ListofPLKlasse.Get(i).GetLabel
  aListofPLKl.Add(theKlasseLb)
end


thePLKl=MsgBox.ListAsString(aListofPLKl,
   "im"++aTStr++"zur Auswahl"++Art++"um einen Punkt zu übernehmen",
   "Auswahl einer Klasse der Datensätze")
if (thePLKl = nil) then
  MsgBox.Error("Die Klasse"++Art++"oder"
     +NL+Art2++"mit der Klasse fehlt!"
     +NL+"Das Programm wird abgebrochen!", "")
  exit
end

'Auswahl der Datensätze im Thema mit den Maus-Klicken
ListofSBz = {}
ListofIdxS = {}

if (PLTheme.CanSelect) then
 'MsgBox.Info("Das Thema"++aTStr++"ist selectierbar.","CanSelect?")

 PLTheme.SelectByPolygon(aMCPg, #VTAB_SELTYPE_NEW)

 'die ausgewählten Datensätze
 AnzS = 0
 for each rec in PLFTab.GetSelection
  'MsgBox.Info("von"++aTStr++":  "++rec.AsString,
  '    "Index-Nummer der selektierten Datensätze")
  aSBz = PLFTab.ReturnValue(aPLKWFld, rec)
  ListofSBz.Add(aSBz)
  AnzS = AnzS + 1
  ListofIdxS.Add(rec.AsString)
 end
 'MsgBox.Info(AnzS.AsString, "Anzahl der selektierten Datensätze")
 'MsgBox.ListAsString(ListofIdxS, "Index-Nummer der selektierten"
 '                ++"Datensätze"++"von"++aTStr, "Information")
 if (AnzS = 0) then
  for each i in 0..AnzPLIdx
   aBz = PLFTab.ReturnValue(aPLKWFld, i)
   if (aBz = thePLKl) then
     ListofSBz.Add(aBz)
     ListofIdxS.Add(i.AsString)
```

```
      end
    end
  end
else
  for each i in 0..AnzPLIdx
    aBz = PLFTab.ReturnValue(aPLKWFld, i)
    if (aBz = thePLKl) then
       ListofSBz.Add(aBz)
       ListofIdxS.Add(i.AsString)
    end
  end
end

aSBz = MsgBox.ListAsString(ListofSBz,
     "von"++aTStr++","++
     "um"++Art2++"auszuwählen"++"um einen Punkt zu übernehmen",
     "Auswahl eines Datensatzes")
aSBzIdx = ListofSBz.FindByValue(aSBz)
aSRIdx = ListofIdxS.Get(aSBzIdx).AsNumber

'MsgBox.Info("von"++aTStr++":  "++aSRIdx.AsString,
'     "Index des ausgewählten Datensatzes")
theShp = PLFTab.ReturnValue(PLShpFld, aSRIdx)
'MsgBox.report("des Themas"++aTStr+NL+
'             "Index-Nummer:"++aSRIdx.AsString+NL+NL+
'             theShp.AsString, "Selektiertes Shape")

PLTheme.ClearSelection

'Feststellung der Koordinaten der ausgewählten Formen im Thema

theProfilList1={}
theProfilList1.Add({theShp})

'Liste der Stützpunkte
ListofFLPt = {}
minxkrd = 100000000
maxxkrd = 0

if (theProfilList1 <> 0) then
  for each q in theProfilList1
    theLines=q.Get(0).AsList
    for each m in theLines
       for each ptx in m
          myx=ptx.Getx
          myy=ptx.Gety
          ListofFLPt.Add(myx@myy)
          if (myx < minxkrd) then
            minxkrd = myx
            minykrd = myy
          end
          if (myx > maxxkrd) then
            maxxkrd = myx
            maxykrd = myy
          end
       end
    end
  end
end

PtAnz= ListofFLPt.Count  'Anzahl der Stützpunkte der Polyline
```

```
  if (ErstPt = "ein Stützpunkt auf einer Polyline") then
    PtAnzIdx= PtAnz - 1
  elseif (ErstPt = "ein Punkt auf einem Polygon") then
    PtAnzIdx= PtAnz - 2
  end

  'Bestimmung der Stellen an Maus-Klicken

  minDist = 1000000
  minIdx = -1

  aMPtx = aMausPt.Getx
  aMPty = aMausPt.Gety

  for each i in 0..PtAnzIdx
      aPt = ListofFLPt.Get(i)
      xkrd= aPt.Getx
      ykrd= aPt.Gety
      xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
      yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
      xyAbst = ((xAbst  + yAbst).sqrt) .Abs
      if (xyAbst < minDist) then
        minDist = xyAbst
        minIdx = i  'Index des nähesten Punktes des Maus-Klickens
      end
  end

  aNPt = ListofFLPt.Get(minIdx)

  if (ErstPt = "ein Punkt auf einem Polygon") then
   if (ZweitAw = "ein Stützpunkt mit der kleinsten x-Koord.") then
     aNPt = Point.Make(minxkrd, minykrd)
   elseif (ZweitAw = "ein Stützpunkt mit der größten x-Koord.") then
     aNPt = Point.Make(maxxkrd, maxykrd)
   end
  end
end

'Korrektur der Polyline oder des Polygons im aktiven View

AkFTab=AkTheme.GetFTab
AkShapeFld=AkFTab.FindField("Shape")
aShpClassStr = AkFTab.GetShapeClass.GetClassName
'MsgBox.Info(aShpClassStr, "Klasse des Themas")

AnzAk = 0   'Anzahl der Datensätze im aktiven Thema
for each rec in AkFTab
  AnzAk = AnzAk + 1
end
IdxAk = AnzAk - 1

'Auswahl eines Datensatzes zur Auswahl
ListofAkFlds=AkFTab.GetFields
AnzAkFld=ListofAkFlds.Count
IdxAkFld=AnzAkFld-1

FldStr=""
for each j in 0..IdxAk
  DtStr=""
  for each i in 0..IdxAkFld
    aFld=ListofAkFlds.Get(i)
    aFldStr=aFld.GetName
```

```
      if (aFldStr <> "Shape") then
        if (j = 0) then
          FldStr = FldStr+aFldStr+"; "
        end
        aValue=AkFTab.ReturnValue(aFld, j)
        DtStr=DtStr+aValue.AsString+"; "
      end
    end
end

MsgBox.Report("vom Thema"++AkStr+","+NL+
      "um zum Teil zu korrigieren"+NL+NL+
      "Die Namen der Felder ohne Shape-Felder"
       +NL+FldStr+NL+NL+
      "Der erste Datensatz:"+NL+DtStr,
      "Information eines Polygons oder einer Polyline ")

AkKWFld = MsgBox.ListAsString(ListofAkFlds,
  "des Themas"++AkStr+NL+
  "für Bezeichnung der Datensäze und"+NL+
  "für Klassifizierung der Legende zur Korrektur",
  "Auswahl eines Feldes")

AkKWFldStr = AkKWFld.AsString

aLegend = AkTheme.GetLegend
aorgLTyp = aLegend.GetLegendType
'MsgBox.Info(aorgLTyp.AsString, "eigentliche Legende")
aListofSymbol = aLegend.GetSymbols
AnzSymb = aListofSymbol.Count
AnzSymbIdx = AnzSymb-1
ListoforgColor = {}

for each symb in 0..AnzSymbIdx
    aorgColor = aListofSymbol.Get(symb).GetColor
    ListoforgColor.Add(aorgColor)
end

aLegend.SetLegendType(#Legend_Type_Unique)
aLegend.Unique(AkTheme, AkKWFldStr)
ListofKlasse = aLegend.GetClassifications
AnzKlasse = ListofKlasse.Count
IdxKlasse = AnzKlasse-1
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")

ListofKlBez = {}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   ListofKlBez.Add(theKlasseLb)
end

theKlBz=MsgBox.ListAsString(ListofKlBez,
  "im"++AkStr++
  "zur Auswahl der Polyline oder des Polygons zur Korrektur",
  "Auswahl einer Bezeichnung der Datensätze")
if (theKlBz = nil) then
  MsgBox.Error("Die Bezeichnung der Polyline, des Polygons oder"
     +NL+"die Polyline oder das Polygon mit der Bezeichnung fehlt!"
     +NL+"Das Programm wird abgebrochen!", "")
 exit
end
```

```
'Selection der Datensätze im Thema mit den Maus-Klicken
ListofSBz = {}
ListofIdxS = {}

if (AkTheme.CanSelect) then
  'MsgBox.Info("Das Thema"++AkStr++"ist selectierbar.","CanSelect?")

  AkTheme.SelectByPolygon(aMCPg, #VTAB_SELTYPE_NEW)

  'die selektierten Datensätze
  AnzS = 0
  for each rec in AkFTab.GetSelection
    'MsgBox.Info("von"++AkStr++":  "++rec.AsString,
    '    "Index-Nummer der selektierten Datensätze")
    aSBz = AkFTab.ReturnValue(AkKWFld, rec)
    ListofSBz.Add(aSBz)
    AnzS = AnzS + 1
    ListofIdxS.Add(rec.AsString)
  end
  'MsgBox.Info(AnzS.AsString, "Anzahl der selektierten Datensätze")
  'MsgBox.ListAsString(ListofIdxS, "Index-Nummer der selektierten"
  '                 ++"Datensätze"++"von"++AkStr, "Information")
  'Anzahl der selektierten Datensätze
  AnzSDS = ListofSBz.Count
  if (AnzSDS = 0) then
    for each i in 0..IdxAk
      aBz = AkFTab.ReturnValue(AkKWFld, i)
      if (aBz = theKlBz) then
        ListofSBz.Add(aBz)
        ListofIdxS.Add(i.AsString)
      end
    end
  end
else
  for each i in 0..IdxAk
    aBz = AkFTab.ReturnValue(AkKWFld, i)
    if (aBz = theKlBz) then
      ListofSBz.Add(aBz)
      ListofIdxS.Add(i.AsString)
    end
  end
end

aSBz = MsgBox.ListAsString(ListofSBz,
     "von"++AkStr++","++
     "um die Polyline oder das Polygon zum Teil zu korrigieren",
     "Auswahl eines Datensatzes")
aSBzIdx = ListofSBz.FindByValue(aSBz)
aSRIdx = ListofIdxS.Get(aSBzIdx).AsNumber

'MsgBox.Info("von"++AkStr++":  "++aSRIdx.AsString,
'     "Index des ausgewählten Datensatzes")
theShp = AkFTab.ReturnValue(AkShapeFld, aSRIdx)
'MsgBox.report("des Themas"++AkStr+NL+
'             "Index-Nummer:"++aSRIdx.AsString+NL+NL+
'             theShp.AsString, "Selektiertes Shape")

AkTheme.ClearSelection

minx = 3000000
maxx = 0
```

```
av.ShowMsg("Feststellung der Koordinaten des aktivenThemas"
    ++(AkTheme.AsString)++"...")

theProfilList1={}
theProfilList1.Add({theShp})

ListofFLx = {}     'Liste der Vertices der geologischen Fläche
ListofFLy = {}
ListofFLPt = {}
minxkrd = 100000000
minykrd = 100000000
maxxkrd = 0
maxykrd = 0

if (theProfilList1 <> 0) then
  for each q in theProfilList1
    theLines=q.Get(0).AsList
    for each m in theLines
      for each ptx in m
        myx=ptx.Getx
        myy=ptx.Gety
        ListofFLx.Add(myx)
        ListofFLy.Add(myy)
        ListofFLPt.Add(myx@myy)
        if (myx < minxkrd) then
          minxkrd = myx
        end
        if (myx > maxxkrd) then
          maxxkrd = myx
        end
        if (myy < minykrd) then
          minykrd = myy
        end
        if (myy > maxykrd) then
          maxykrd = myy
        end
      end
    end
  end
end

PtAnz= ListofFLx.Count  'Anzahl der Vertices der geologischen Fläche
PtAnzIdx= PtAnz-1

'Bestimmung der Stelle des Punktes auf der geologischen Fläche

minDist=1000
minIdx=-1

aMPtx = aNPt.Getx
aMPty = aNPt.Gety

for each i in 0..PtAnzIdx
  xkrd= ListofFLx.Get(i)
  ykrd= ListofFLy.Get(i)
  xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
  yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
  xyAbst = ((xAbst  + yAbst).sqrt) .Abs
  if (xyAbst < minDist) then
    minDist = xyAbst
    minIdx = i  'Index des nähesten Punktes des Maus-Klickens
  end
```

```
end

'Bestimmung der Stelle auf der geologischen Fläche
'als x-, y-Koordinaten

if (minIdx = 0) then
   vIdx = 0     'ein Vertex auf der Linie vor dem Punkt im Punkt_Thema
   nIdx = 1     'ein Vertex auf der Linie nach dem Punkt im Punkt_Thema
elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
   xkrdv = ListofFLx.Get((minIdx -1))  'x-Koord. vom letzten Vertex
   xkrd  = ListofFLx.Get(minIdx)       'x-Koord. vom nächesten Vertex
   xkrdn = ListofFLx.Get((minIdx +1)) 'x-Koord. vom nächsten Vertex

   ykrdv = ListofFLy.Get((minIdx -1))  'y-Koord. vom letzten Vertex
   ykrd = ListofFLy.Get(minIdx)        'y-Koord. vom nächesten Vertex
   ykrdn = ListofFLy.Get((minIdx +1))  'y-Koord. vom nächsten Vertex

   xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
   yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
   xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten Vertex
                                  'vom nächesten Vertex
   xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
   yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
   xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten Vertex
                                  'vom nähesten Vertex
   xML = (xkrd - aMPtx) * (xkrd - aMPtx)
   yML = (ykrd - aMPty) * (ykrd - aMPty)
   xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Punkt
                                  'vom nächesten Vertex
   xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
   yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
   xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Punkt
                                     'vom letzten Vertex
   xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
   yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
   xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Punkt
                                    'vom nächsten Vertex
   vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem
   vLpML = xyAv + xyML        'Punkt und dem letzten Vertex

   nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen
   nLpML = xyAn + xyML        'dem Punkt und dem nächsten Vertex

   vM = (vLmML - xyMLv).Abs   'Differenz zwischen der theorischen und der
   vP = (vLpML - xyMLv).Abs   'tatsächlichen Länge im Bezug
                              'auf den letzten Vertex
   nM = (nLmML - xyMLn).Abs   'Differenz zwischen der theorischen und der
   nP = (nLpML - xyMLn).Abs   'tatsächlichen Länge im Bezug
                              'auf den nächsten Vertex
   ListofLg = {vM, vP, nM, nP}   'Suche nach der kleinsten Länge
   MinLg = 10000
   TheLgIdx = -1
   For each aLg in 0..3
        theLg = ListofLg.Get(aLg)
        if (theLg < MinLg) then
             MinLg = theLg
             TheLgIdx = aLg
        end
   end
   if (xyML < 10) then
     vIdx = minIdx - 1
     nIdx = minIdx + 1
```

```
    end
    if (xyML >= 10) then
      if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
        vIdx = minIdx - 1  'ein Vertex auf der Linie vor dem Punkt
        nIdx= minIdx       'ein Vertex auf der Linie nach dem Punkt
      elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
        vIdx = minIdx      'ein Vertex auf der Linie vor dem Punkt
        nIdx = minIdx + 1  'ein Vertex auf der Linie nach dem Punkt
      else
        vIdx = minIdx      'ein Vertex auf der Linie vor dem Punkt
        nIdx = minIdx + 1  'ein Vertex auf der Linie nach dem Punkt
      end
    end
elseif (minIdx = PtAnzIdx)   then
     vIdx = minIdx - 1
     nIdx = minIdx
end
'Einsetzen des Punktes auf die Polyline oder das Polygon
ListofPoint = {}

for each i in 0..PtAnzIdx
   if (i <= vIdx) then
      ax = ListofFLx.Get(i)
      ay = ListofFLy.Get(i)
      ListofPoint.Add(ax@aY)
   end
end
ListofPoint.Add(aNPt)
for each i in 0..PtAnzIdx
   if (i >= nIdx) then
      ax = ListofFLx.Get(i)
      ay = ListofFLy.Get(i)
      ListofPoint.Add(ax@aY)
   end
end
ListofListofPoint={}
ListofListofPoint.Add(ListofPoint)
if (aShpClassStr = "Polyline") then
  theNShp=PolyLine.Make(ListofListofPoint)
elseif (aShpClassStr = "Polygon") then
  theNShp=Polygon.Make(ListofListofPoint)
end

AkFTab.SetEditable(false)
AkFTab.SetEditable(true)
AkFTab.SetValue(AkShapeFld, aSRIdx, theNShp)
AkFTab.SetEditable(false)

av.ShowMsg("Veränderung der Legenden des Themas"
    ++AkTheme.AsString+"...")

theLegend = AkTheme.GetLegend
if (aorgLTyp = #LEGEND_TYPE_SIMPLE) then
    theLegend.SetLegendType(#LEGEND_TYPE_SIMPLE)
    theLegend.SingleSymbol
    theSymbol=theLegend.GetSymbols.Get(0)
    theSymbol.SetColor(ListoforgColor.Get(0))
else
    theLegend.SetLegendType(#Legend_Type_Unique)
    theLegend.Unique(AkTheme, AkKWFldStr)
    ListofKlasse=theLegend.GetClassifications
    AnzKlasse=ListofKlasse.Count
```

```
    IdxKlasse=AnzKlasse-2
    'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
    aListofSymbol = theLegend.GetSymbols
    AnzSymb = aListofSymbol.Count
    AnzSymbIdx = AnzSymb-1

  for each symb in 0..AnzSymbIdx
     aColor = ListoforgColor.Get(symb)
     aListofSymbol.Get(symb).SetColor(aColor)
  end
end
AkTheme.UpdateLegend




'pgzauslz.ave
'Aus den PolyLinienZ in einer 3D-Szene wird ein Thema mit den PolygonZ hergestellt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc        'ein aktives View (3D Szene)
PLThm=theView.GetActiveThemes.Get(0) 'ein aktives Thema im aktiven View
Qt1=MsgBox.YesNo("Ist das aktive Thema"++PLThm.AsString++"richtig?",
  "Kontrolle des ausgewählten PolyLine-Themas", true)
if (Not Qt1) then
  MsgBox.Error("Das ausgewählte Thema ist falsch!"+NL+
            "Das Programm wird unterbrochen.", "")
  Exit
end
PLFTab=PLThm.GetFTab
PLShpFld=PLFTab.FindField("Shape")
ListofPLFld=PLFTab.GetFields
'Feststellung der Anzahl der 3D-PolyLine in dem Thema
Anz3DPL=0
for each rec in PLFTab
   Anz3DPL=Anz3DPL+1
end
Idx3DPL=Anz3DPL-1
av.ShowMsg("Feststellung der Koordinaten der 3D-Profilschnitte in Thema"
         ++(PLThm.AsString)++"...")
av.ShowStopButton
ListofListofPLPT={}
for each aPL in 0..Idx3DPL
  Ng=aPL+1
  theShp=PLFTab.ReturnValue(PLShpFld, aPL)
  ListofPLPT={}
  ListofListofPt=theShp.AsList
  for each aListofPt in ListofListofPt
     for each aPt in aListofPt
        ListofPLPT.Add(aPt)
     end
  end
  ListofListofPLPT.Add(ListofPLPT)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/Anz3DPL*100)
  if (not more) then
    break
 end
```

```
end
av.ShowMsg("Umwandlung der ProfilZ im Thema"++PLThm.AsString
      ++"in PolygonZ ...")
av.ShowStopButton
minYStr=MsgBox.Input("die kleinste Höhe"+NL+
      "der gesammten PolygonZ [m]",
      "Bestimmung der Koordinaten des Polygons", "0.00")
minY=minYStr.AsNumber
ListofPg={}
for each i in 0..Idx3DPL
  ListofListofPgPts={}
  ListofPgPts={}
  ListofPLPT={}
  ListofPLPT=ListofListofPLPT.Get(i)
  AnzPLPT=ListofPLPT.Count
  IdxPLPT=AnzPLPT-1
  for each k in 0..IdxPLPT
    thePT=ListofPLPT.Get(k)
    ListofPgPts.Add(thePT)
  end
  j=i+1
  ListofNPLPT={}
  if ((j < Idx3DPL) or (j = Idx3DPL)) then
    ListofNPLPT=ListofListofPLPT.Get(j)
  elseif (j > Idx3DPL) then
    j2=j-1
    ListofNPLPT=ListofListofPLPT.Get(j2)
  end
  AnzNPLPT=ListofNPLPT.Count
  IdxNPLPT=AnzNPLPT-1
  for each k in 0..IdxNPLPT
    UmkIdx=IdxNPLPT-k
    thePT=ListofNPLPT.Get(UmkIdx)
    if (j > Idx3DPL) then
      thePTx=thePT.Getx
      thePTy=thePT.Gety
      ListofPgPts.Add(thePTx@thePTy@minY)
    else
      ListofPgPts.Add(thePT)
    end
  end
  ListofListofPgPts.Add(ListofPgPts)
  thePG=PolygonZ.Make(ListofListofPgPts)
  ListofPg.Add(thePg)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(j/Anz3DPL*100)
  if (not more) then
    break
  end
end
av.ShowMsg("Speicherung der neuen PolygonZ in einem Thema ...")
ListofGeol={"Deckschichten", "Niederterrassen",
        "Mittelterrassen", "Ältere Schichten als Mittelterrassen"}
AnzGeol=ListofGeol.Count
IdxGeol=AnzGeol-1
'Ein Feature-Shape-File für Schichten-Polygon (PolygonZ) wird hergestellt.
WDStr=theProject.GetWorkDir.AsString
fnStr=PLThm.AsString.Left(8)
fnName=FileName.Make(WDStr).MakeTmp(fnStr, "shp")
fnPg=FileDialog.Put(fnName, "*.shp", "Output shape File (PolygonZ)")
if (fnPg=nil) then exit end
fnPg.SetExtension("shp")
```

```
FTabPg=FTab.MakeNew(fnPg, PolygonZ)
ShpFld=FTabPg.FindField("shape")
IDFld=Field.Make("ID", #Field_Short, 2, 0)
SchichtFld=Field.Make("Schichten", #Field_Char, 50, 0)
ListofNFld1={IDFld, SchichtFld}
FTabPg.AddFields(ListofNFld1)
FTabPg.SetEditable(false)
FTabPg.SetEditable(true)
for each aRec in 0..Idx3DPL
   FTabPg.AddRecord
   myPolyg=ListofPg.Get(aRec)
   FTabPg.SetValue(ShpFld, aRec, myPolyg)
   FTabPg.SetValue(IDFld, aRec, aRec)
   myGeol=ListofGeol.Get(aRec)
   FTabPg.SetValue(SchichtFld, aRec, myGeol)
end
FTabPg.SetEditable(false)
thmNew=FTheme.Make(FTabPg)
theView.AddTheme(thmNew)
```

```
'pgzrecht.ave
'Ein rechteckiges 3D-PolygonZ wird  in einer aktiven Szene hergestellt.
'Ein Menü oder eine Schaltfläche in einer aktiven Szene zum Anklicken.

theProject = av.GetProject
theSzene = av.GetActiveDoc

myScript = theProject.FindScript("pgzrecht")
myScript.SetNumberFormat("d.dd")

'Eingabe der Koordinaten des Polygons
aminxStr = MsgBox.Input("für einen kleinsten RW",
        "Eingabe der Koordinaten des Gebietes", "2558600.00")
amaxxStr = MsgBox.Input("für einen größten RW",
        "Eingabe der Koordinaten des Gebietes", "2582450.00")
aminyStr = MsgBox.Input("für einen kleinsten HW",
        "Eingabe der Koordinaten des Gebietes", "5618450.00")
amaxyStr = MsgBox.Input("für einen größten HW",
        "Eingabe der Koordinaten des Gebietes", "5641050.00")
aminzStr = MsgBox.Input("für eine kleinste Hoehe [m]",
        "Eingabe der Koordinaten des Gebietes", "0.00")
amaxzStr = MsgBox.Input("für eine größte Hoehe [m]",
        "Eingabe der Koordinaten des Gebietes", "200.00")

aminx = aminxStr.AsNumber
aminy = aminyStr.AsNumber
aminz = aminzStr.AsNumber

amaxx = amaxxStr.AsNumber
amaxy = amaxyStr.AsNumber
amaxz = amaxzStr.AsNumber

ListofStelle = {"Nord", "Sued", "Ost", "West", "Oben", "Boden"}
ListofkSt = {"N", "S", "O", "W", "T", "B"}
aSt = MsgBox.ListAsString(ListofStelle, "für die Stelle eines Polygonzs",
    "Auswahl einer Stelle")
aStIdx = ListofStelle.FindByValue(aSt)
```

```
akSt = ListofkSt.Get(aStIdx)


if (akSt = "N") then
  ListofPgZPt = {amaxx@amaxy@amaxz,aminx@amaxy@amaxz,
              aminx@amaxy@aminz,amaxx@amaxy@aminz}
elseif (akSt = "S") then
  ListofPgZPt = {aminx@aminy@amaxz,amaxx@aminy@amaxz,
              amaxx@aminy@aminz,aminx@aminy@aminz}
elseif (akSt = "O") then
  ListofPgZPt = {amaxx@aminy@amaxz,amaxx@amaxy@amaxz,
              amaxx@amaxy@aminz,amaxx@aminy@aminz}
elseif (akSt = "W") then
  ListofPgZPt = {aminx@amaxy@amaxz,aminx@aminy@amaxz,
              aminx@aminy@aminz,aminx@amaxy@aminz}
elseif (akSt = "T") then
  ListofPgZPt = {aminx@amaxy@amaxz,amaxx@amaxy@amaxz,
              amaxx@aminy@amaxz,aminx@aminy@amaxz}
elseif (akSt = "B") then
  ListofPgZPt = {aminx@amaxy@aminz,amaxx@amaxy@aminz,
              amaxx@aminy@aminz,aminx@aminy@aminz}
end

ListofListofPgZPt = {}
ListofListofPgZPt.Add(ListofPgZPt)
thePg = PolygonZ.Make(ListofListofPgZPt)
ListofAbk = {"Oben", "Boden", "Deck", "Nord", "Sued",
          "Ost", "West", "MT", "NT", "Präq"}
aAbk = MsgBox.ListAsString(ListofAbk, "für das neue Polygon",
    "Auswahl einer Abkürzung")

'Speicherung des neuen PolygonZ
av.ShowMsg("Speicherung der Fläche als ein Polygon ...")
av.ShowStopButton

ListofAW={"ein vorhandenes Thema", "ein neues Thema"}
AW11=MsgBox.ChoiceAsString(ListofAW, "zur Speicherung von PolygonZ",
  "Auswahl eines PolygonZ-Themas")

if (AW11 = "ein vorhandenes Thema") then
  ListofPgZ={}
  ListofThms = theSzene.GetThemes
  for each aT in ListofThms
    if (aT.Is(FTheme)) then
      aFTab=aT.GetFTab
      if(aFTab.GetShapeClass.IsSubClassof(PolygonZ)) then
        ListofPgZ.Add(aT)
      end
    end
  end
  thePgZThm=MsgBox.ChoiceAsString(ListofPgZ,
        "zur Speicherung von PolygonZ",
        "Auswahl eines PolygonZ-Themas")
  PgFTab=thePgZThm.GetFTab
  PgShpFld=PgFTab.FindField("Shape")
  ListofPgFlds = PgFTab.GetFields
  PgSIDFld=MsgBox.ListAsString(ListofPgFlds, "für ID",
       "Auswahl eines Feldes im Thema"++thePgZThm.AsString)
  PgNmAbkFld=MsgBox.ListAsString(ListofPgFlds, "für Namen_Abk",
       "Auswahl eines Feldes im Thema"++thePgZThm.AsString)
  PgBZFld=MsgBox.ListAsString(ListofPgFlds,
       "für Bezeichnung des Polygons",
```

```
        "Auswahl eines Feldes im Thema"++thePgZThm.AsString)

  AnzPgZR=0
  for each arec in PgFTab
    AnzPgZR=AnzPgZR+1
  end
  recNr=AnzPgZR

elseif (AW11 = "ein neues Thema") then
  WDStr=theProject.GetWorkDir.AsString
  fnStr=FileName.Make(WDStr).MakeTmp("Pgnts11","shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PolygonZ)")
  if (fName=nil) then exit end
  fName.SetExtension("shp")
  PgFTab=FTab.MakeNew(fName, PolygonZ)
  PgShpFld=PgFTab.FindField("shape")
  PgSIDFld=Field.Make("PgZ_ID", #Field_Short, 2, 0)
  PgNmAbkFld=Field.Make("Namen_Abk", #Field_Char, 6, 0)
  PgBZFld=Field.Make("Kennwort", #Field_Char, 4, 0)

  ListofPgFlds={PgSIDFld, PgNmAbkFld, PgBZFld}
  PgFTab.AddFields(ListofPgFlds)

  recNr=0
end

PgFTab.SetEditable(false)
PgFTab.SetEditable(true)

PgFTab.AddRecord
PgFTab.SetValue(PgShpFld, recNr, thePg)
PgFTab.SetValue(PgSIDFld, recNr, recNr)
PgFTab.SetValue(PgNmAbkFld, recNr, aAbk)
PgFTab.SetValue(PgBZFld, recNr, akSt)

PgFTab.SetEditable(false)

if (AW11 = "ein vorhandenes Thema") then
  theTheme=thePgZThm
elseif (AW11 = "ein neues Thema") then
  NewFThm=FTheme.Make(PgFTab)
  theSzene.AddTheme(NewFThm)
  theTheme=NewFThm
end

theTheme.UpdateLegend




'plaus2pl.ave
'Dieses Script nimmt zwei Polyline in einem Karten-View für eine Rinne.
'Aus denen wird eine Mittellinie im ganzen oder bestimmten Bereich gebildet.
'Die Mittellinie für einen anderen Bereich wird von einer anderen Mittellinie
'übernommen.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc    'Ein aktives View
myScript=theProject.FindScript("plaus2pl")
myScript.SetNumberFormat( "d.dd") ' script default
```

```
'Auswahl eines Bereiches zur Berechnung der Mittellinie einer Rinne
ListofBereich={"ganzen Bereich", "einen Teilbereich"}
myBr=MsgBox.ChoiceAsString(ListofBereich,
    "zur Berechnung der Mittellinie"+NL+"einer Rinne",
    "Auswahl eines Bereiches der Rinne")

if (myBr = "einen Teilbereich") then
  'Der Teilbereich zur Berechnung der Mittellinie wird eingegeben.
  AnfHWStr=MsgBox.Input("Der kleinste Hochwert des Bereiches:",
        "Eingabe eines Bereiches zur Berechnung", "5618450.00")
  EndHWStr=MsgBox.Input("Der größte Hochwert des Bereiches:",
        "Eingabe eines Bereiches zur Berechnung", "5641050.00")
  AnfHW=AnfHWStr.AsNumber
  EndHW=EndHWStr.AsNumber
  HWAbstStr00="50"
  HWAbstStr =MsgBox.Input("zwischen Punkten von HW",
         "Eingabe eines Raster-Abstandes (m)", HWAbstStr00)
  HWAbst=HWAbstStr.AsNumber
  HWIdx=(((AnfHW-EndHW).Abs)/ HWAbst).SetFormat("").SetFormat("d")
  AnzHW=HWIdx+1
end

ListofThms= theView.GetThemes
ListofPLThemes = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     if (aFTab.GetShapeClass.IsSubclassOf(PolyLine)) then
       ListofPLThemes.Add(aT)
     end
   end
end
RPLTheme=MsgBox.ChoiceAsString(ListofPLThemes,
  "Rechtes PolyLine-Thema für eine Rinne", "Auswahl eines Themes")
RPLFTab=RPLTheme.GetFTab
RPLShpFld=RPLFTab.FindField("Shape")
theRShape=RPLFTab.ReturnValue(RPLShpFld, 0)
ListofRShapes={}
ListofRShapes.Add({theRShape})
'Erkennung der Koordinaten auf der rechten Polyline
'2D-Polyline wird in Liste der Koordinaten umgewandelt
ListofRPLPt={}
if (ListofRShapes <> 0) then
  for each aShp in ListofRShapes
     aListofListofPts=aShp.Get(0).AsList
     for each aList in aListofListofPts
        for each myPt in aList
           ListofRPLPt.Add(myPt)
        end
     end
  end
end
LtofLtofRPLPt={}
LtofLtofRPLPt.Add(ListofRPLPt)
RPolyL=PolyLine.Make(LtofLtofRPLPt)
MsgBox.Report(RPolyL.AsString, "Kontrolle: Rechte PolyLine (Input)")

LPLTheme=MsgBox.ChoiceAsString(ListofPLThemes,
 "Linkes PolyLine-Thema für eine Rinne", "Auswahl eines Themas")
LPLFTab=LPLTheme.GetFTab
LPLShpFld=LPLFTab.FindField("Shape")
```

```
theLShape=LPLFTab.ReturnValue(LPLShpFld, 0)
ListofLShapes={}
ListofLShapes.Add({theLShape})
'Erkennung der Koordinaten auf der linken Polyline
'2D-Polyline wird in Liste der Koordinaten umgewandelt.
ListofLPLPt={}
if (ListofLShapes <> 0) then
  for each aShp in ListofLShapes
    aListofListofPts=aShp.Get(0).AsList
    for each aList in aListofListofPts
      for each myPt in aList
        ListofLPLPt.Add(myPt)
      end
    end
  end
end
LtofLtofLPLPt={}
LtofLtofLPLPt.Add(ListofLPLPt)
LPolyL=PolyLine.Make(LtofLtofLPLPt)
MsgBox.Report(LPolyL.AsString, "Kontrolle: linke PolyLine (Input)")
'Berrechnung der Mittellinie von den beiden Polyline
av.ShowMsg("Berrechnung der Mittellinie mit"
        ++LPLTheme.AsString++"und"++RPLTheme.AsString++"...")
av.ShowStopButton
AnzRPt=ListofRPLPt.Count
RPtIdx=AnzRPt-1
MsgBox.Info(AnzRPt.AsString,"Anzahl der Punkte der rechten PolyLine")
AnzLPt=ListofLPLPt.Count
LPtIdx=AnzLPt-1
MsgBox.Info(AnzLPt.AsString,"Anzahl der Punkte der linken PolyLine")
minEntf=100000000
StellemEf=0
Ng=0
ListofMPt={}
for each arVt in 0..RPtIdx
  Ng=Ng+1
  RPt=ListofRPLPt.Get(arVt)
  RPtRW=RPt.Getx
  RPtHW=RPt.Gety
  minRPtHW=RPtHW-5000
  maxRPtHW=RPtHW+5000
  for each alVt in 0..LPtIdx
    LPt=ListofLPLPt.Get(alVt)
    LPtHW=LPt.Gety
    if ((LPtHW > minRPtHW) and (LPtHW < maxRPtHW)) then
      LPtRW=LPt.Getx
      Entf2PLPt=((((RPtRW-LPtRW)*(RPtRW-LPtRW))+((RPtHW-LPtHW)
            *(RPtRW-LPtHW))).Abs).Sqrt
      if (Entf2PLPt < minEntf) then
        minEntf=Entf2PLPt
        StellemEf=alVt
      end
    end
  end
  JPtL=ListofLPLPt.Get(StellemEf)
  JPtLRW=JPtL.Getx
  JPtLHW=JPtL.Gety
  RWM=(RPtRW+JPtLRW)/2
  HWM=(RPtHW+JPtLHW)/2
  ListofMPt.Add(RWM@HWM)
  minEntf=100000000
  'Show percentage complete with enabled stop button
```

```
    more=av.SetStatus((Ng/(AnzRPt))*100)
    if (not more) then
      break
    end
  end
  AnzMPt=ListofMPt.Count
  MPtIdx=AnzMPt-1
  MsgBox.Info(AnzMPt.AsString,
       "Anzahl der Punkte der neuen Mittellinie im Bereich")
  LtofLtofMPt={}
  if (myBr = "ganzen Bereich") then
    LtofLtofMPt={}
    LtofLtofMPt.Add(ListofMPt)
  elseif (myBr = "einen Teilbereich") then
    PLTheme=MsgBox.ChoiceAsString(ListofPLThemes,
          "PolyLine-Thema der Mittellinien"
             ++"von einer anderen Rinne",
             "Auswahl eines Themas, um teilweise zu übernehmen")
    PLFTab=PLTheme.GetFTab
    PLShpFld=PLFTab.FindField("Shape")

    AnzRec=0
    For each rec in PLFTab
       AnzRec=AnzRec+1
    end
    MsgBox.Info(AnzRec.AsString,
      "Anzahl der Records (PolyLine) in Thema"++PLTheme.AsString)

    theRZtr=PLFTab.ReturnValue(PLShpFld, 0)
    ListofShapes={}
    ListofShapes.Add({theRZtr})
    myLstofPts={}
    if (ListofShapes <> 0) then
      for each aShape in ListofShapes
         aListofListofPts=aShape.Get(0).AsList
         for each aLst in aListofListofPts
            for each myPt in aLst
               myLstofPts.Add(myPt)
            end
         end
      end
    end

    AnzRZtr=myLstofPts.Count
    MsgBox.Info(AnzRZtr.AsString,
     "Anzahl der Punkte im alten PolyLine-Thema"++PLTheme.AsString)
    IdxRZtr=AnzRZtr-1
    ListofmyPts={}
    for each ePt in 0..IdxRZtr
       einPt=myLstofPts.Get(ePt)
       myHW=einPt.Gety
       if (myHW < AnfHW) then
         ListofmyPts.Add(einPt)
       end
    end
    AnzMPt=ListofMPt.Count
    MPtIdx=AnzMPt-1
    for each ePt in 0..MPtIdx
       einPt=ListofMPt.Get(ePt)
       myHW=einPt.Gety
       if (((myHW > AnfHW) or (myHW = AnfHW))
          and ((myHW < EndHW) or (myHW = EndHW))) then
```

```
        ListofmyPts.Add(einPt)
      end
    end
    for each ePt in 0..IdxRZtr
      einPt=myLstofPts.Get(ePt)
      myHW=einPt.Gety
      if (myHW > EndHW) then
        ListofmyPts.Add(einPt)
      end
    end
     LtofLtofMPt.Add(ListofmyPts)
end
thePolyL=PolyLine.Make(LtofLtofMPt)
MsgBox.Report(thePolyL.AsString, "Die Punkte der Mittellinie")

'Speicherung der Polyline in einem neuen Mittellinie-Thema
Qt33=MsgBox.YesNo("Gibt es schon ein Polyline-Thema"+NL+
    "für die neue Mittellinie im View"++theView.AsString,
    "Speicherung der Mittellinie", true)
if (Qt33) then
  theMLThm=MsgBox.ChoiceAsString(ListofPLThemes,
  "das die Mittellinien der Rinnen enthält", "Auswahl eines Themas")
  MPLFTab = theMLThm.GetFTab
  MPLIDFld = MPLFTab.FindField("ID")
  MPNmFld = MPLFTab.FindField("Beschreibung der Mittellinie der Rinnen")
  'Anzahl der vorhandenen Mittellinien im Thema
  AnzML=0
  for each aPLrec in MPLFTab
    AnzML = AnzML +1
  end
  recNr= AnzML
elseif (Not Qt33) then
  class = PolyLine
  fnNew = av.GetProject.MakeFileName("mtlrin11", "shp")
  fndef = FileDialog.Put(fnNew, "*.shp",
       "Neues Thema für eine Mittellinie")
  if (fndef <> nil) then
    MPLFTab = FTab.MakeNew(fndef, class)
    if (MPLFTab.HasError) then
      if (MPLFTab.HasLockError) then
        MsgBox.Error("Unable to acquire Write Lock for file "
            + fndef.GetBaseName, "")
      else
        MsgBox.Error("Unable to create " + fndef.GetBaseName, "")
      end
      return nil
    end
  end
  if ((fndef =nil) or (MPLFTab = nil)) then
    MsgBox.Error("Die Datei"+ fndef.GetBaseName +
            "wurde nicht hergestellt", "")
    exit
  end
  MPLIDFld = Field.Make("ID", #FIELD_DECIMAL, 3, 0)
  MPNmFld = Field.Make("Beschreibung der Mittellinie der Rinnen",
         #FIELD_Char, 50, 0)
  MPLIDFld.SetVisible( TRUE )
  MPNmFld.SetVisible( TRUE )
  MPLFTab.AddFields({MPLIDFld, MPNmFld })
  RecNr = 0
end
MPLShpFld = MPLFTab.FindField("Shape")
```

```
BeschrStr=MsgBox.Input("der Rinnen"++ LPLTheme.AsString++"und"
                ++RPLTheme.AsString,
    "Beschreibung der neuen Mittellinie",
    "Mittellinie der Rinne"++recNr.AsString)

MPLFTab.SetEditable(False)
MPLFTab.SetEditable(true)
MPLFTab.AddRecord
MPLFTab.SetValue(MPLShpFld, recNr, thePolyL)
MPLFTab.SetValue(MPLIDFld, recNr, recNr)
MPLFTab.SetValue(MPNmFld, recNr, BeschrStr)
MPLFTab.SetEditable(False)

if (Not Qt33) then
  theMLThm =FTheme.Make(MPLFTab)
  theView.AddTheme(theMLThm)
elseif (Qt33) then
  theMLThm.UpdateLegend
end




'plaus4pt.ave
'Ein Polyline-Thema mit einem oder mit mehreren Datensätzen wird
'in einem aktiven View aus Punkten in einem Punkt-Thema
'(evtl. einem Ereignis-Thema) hergestellt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc  'ein aktives View

ListofThms = theView.GetThemes
ListofPtThms = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofPtThms.Add(aT)
      end
   end
end

av.ShowMsg("Eingabe eines Punkt-Themas und"
     ++"Feststellung der Anzahl der Punkte in dem Thema ...")

PtTheme = MsgBox.ChoiceAsString(ListofPtThms,
      "um aus den Punkten ein Polyline-Thema herzustellen.",
      "Auswahl eines Punkt-Themas")

PtFTab = PtTheme.GetFTab
ListofFlds = PtFTab.GetFields
MsgBox.ListAsString(ListofFlds, "die im Punkt-Thema"
      ++PtTheme.GetName++"enthalten sind", "Anzeige der Felder")

PtShpFld = ListofFlds.Get(0)

ListofAnzHFld = {"0", "ein oder mehrere Felder", "nicht festgestellt"}

AW11 = MsgBox.ListAsString(ListofAnzHFld,
```

```
      "für Höhendaten zur Herstellung der y-Koordinaten",
      "Feststellung der Anzahl der Felder")

if (AW11 = "nicht festgestellt") then
  Msgbox.Info("Die Datensätze in der Tabelle müssen festgestellt werden!"
  +NL+"Das Programm wird abgebrochen.", "Information")
  exit

elseif (AW11 = "0") then
   aHFldStr = MsgBox.Input("zur Bezeichnung der Polyline",
            "Eingabe eines Kennwortes", "Grrmto")
   IdxHFld = 0

elseif (AW11 = "ein oder mehrere Felder") then

   AnzHFldStr=MsgBox.Input("die die Höhen der Punkte enthalten",
         "Anzahl der Felder", "4")

   AnzHFld=AnzHFldStr.AsNumber.SetFormat("d")
   IdxHFld=AnzHFld-1

   if (AnzHFldStr = nil) then
     MsgBox.Info("Die Anzahl der Felder für Höhen"+NL+"im Thema"
        ++PtTheme.GetName+"ist noch festzustellen!", "Information")
     exit
   end

   ListofHFlds={}
   HStr=""
   for each i in 0..IdxHFld
     Nr=i+1
     PtHFld = MsgBox.ListAsString(ListofFlds,
          "für"++Nr.AsString+". Höhe [m]"+NL+
          "bis jetzt eingegeben:"++HStr,
          "Auswahl eines Feldes im Thema"++PtTheme.GetName)
     ListofHFlds.Add(PtHFld)
     NHStr=PtHFld.GetAlias
     HStr=HStr++NHStr+";"
   end

   HFaktorStr=MsgBox.Input("zur vertikalen Überhöhung der Profilschnitte",
         "Eingabe eines Faktors", "50")
   HFaktor=HFaktorStr.AsNumber

end

AnzPt=0
for each Pt in PtFTab
    AnzPt=AnzPt+1
end
AnzPtIdx=AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"
         ++PtTheme.AsString)

ListofPL={}   'Herstellung der neuen Profilschnitte als Polyline
for each j in 0..IdxHFld
   if (AW11 = "ein oder mehrere Felder") then
     aHFld=ListofHFlds.Get(j)
   end
   Listof2DPt={}
   ListofListof2DPt={}
   for each i in 0..AnzPtIdx
```

```
      aPt = PtFTab.ReturnValue(PtShpFld, i)
      aX = aPt.Getx
      if (AW11 = "0") then
        aY = aPt.Gety
      elseif (AW11 = "ein oder mehrere Felder") then
        aY = (PtFTab.ReturnValue(aHFld, i))*HFaktor
      end
      Listof2DPt.Add(aX@aY)
    end
    ListofListof2DPt.Add(Listof2DPt)
    thePL=Polyline.Make(ListofListof2DPt)
    ListofPL.Add(thePL)
end

av.ShowMsg("Speicherung der hergestellten 2D-PolyLine ...")
'Ein Feature-Shape-File für einen Profilschnitt (PolyLine) wird hergestellt.
theWDStr = theProject.GetWorkDir.AsString
DName = ("Pl"+(PtTheme.AsString.LCase)).Left(6)
fnStr=FileName.Make(theWDStr).MakeTmp(DName,"shp")
fnGrMt=FileDialog.Put(fnStr, "*.shp", "Output 2D shape File (PolyLine)")
if (fnGrMt=nil) then exit end
fnGrMt.SetExtension("shp")
GrMtFTab=FTab.MakeNew(fnGrMt, PolyLine)
ShapeFld=GrMtFTab.FindField("shape")
IDFld=Field.Make("ID", #Field_Short, 2, 0)
NamenFld=Field.Make("Namen", #Field_Char, 10, 0)
ListofFlds1={IDFld, NamenFld}
GrMtFTab.AddFields(ListofFlds1)

GrMtFTab.SetEditable(false)
GrMtFTab.SetEditable(true)
for each i in 0..IdxHFld
  thePolyL = ListofPL.Get(i)
  if (AW11 = "ein oder mehrere Felder") then
    aHFldStr = ListofHFlds.Get(i).AsString
  end
  GrMtFTab.AddRecord
  GrMtFTab.SetValue(ShapeFld, i, thePolyL)
  GrMtFTab.SetValue(IDFld, i, i)
  GrMtFTab.SetValue(NamenFld, i, aHFldStr)
end
GrMtFTab.SetEditable(false)
thmNew=FTheme.Make(GrMtFTab)
theView.AddTheme(thmNew)




'plkoabgh.ave
'Ein Abschnitt eines Profilschnittes wird durch eine gleiche Höhe
'korrigiert. Der Anfang des Abschnittes wird durch einen und dessen
'Ende durch zweimaliges Maus-Klicken auf dem Profilschnitt bestimmt.
'Durch das letzte zweimalige Maus-Klicken wird ein Abschnitt und
'ein Punkt auf dem Profilschnitt bestimmt, der eine bestimmte Höhe hat.
'Die bestimmte Höhe wird durch einen Stützpunkt der Polyline an dem
'ersten Maus-Klicken, einen Punkt auf der Polyline an dem ersten Maus-Klicken,
'eine Tastatur-Eingabe oder einen Fadenkreuz (z.B. Kreuz1.shp) bestimmt
'Dieses Script wird als ein Werkzeug (Tool) im aktiven View benutzt.

theProject=av.GetProject
```

```
AkView=av.GetActiveDoc     'Ein aktives View-Thema

'Maus-Klicken auf das Bildschirm
aEingPolyL = AkView.GetDisplay.ReturnUserPolyLine

ListofListofEingPt = aEingPolyL.AsList
ListofEingPt = ListofListofEingPt.Get(0)
AnzM = ListofEingPt.Count
AnzMIdx = AnzM - 1
ListofMPt = {}
ListofMCPg = {}
for each mPt in 0..AnzMIdx
  aMPt = ListofEingPt.Get(mPt)
  ListofMPt.Add(aMPt)
  aCircle = Circle.Make(aMPt, 50)
  aMCPg = aCircle.AsPolygon
  ListofMCPg.Add(aMCPg)
end

AkTheme=AkView.GetActiveThemes.Get(0)
aTStr = AkTheme.AsString

qt00=MsgBox.YesNo("Ist das aktive Thema"++aTStr
     +NL+"zur Korrektur richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not qt00) then
  MsgBox.Error("Das aktive Thema"++aTStr++"ist falsch!"+NL+
            "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

AkFTab = AkTheme.GetFTab

aListofFlds = AkFTab.GetFields
AnzFld = aListofFlds.Count
IdxFld = AnzFld - 1
AkShpFld = aListofFlds.Get(0)

AnzDs = 0          'Anzahl der Datensätze
for each rec in AkFTab
  AnzDs = AnzDs + 1
end
IdxDs = AnzDs - 1

'Feststellung des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxFld
 aFld = aListofFlds.Get(i)
 aFldStr=aFld.GetName
 if (aFldStr <> "Shape") then
   FldStr = FldStr + aFldStr+"; "
   aValue = AkFTab.ReturnValue(aFld, 0)
   DtStr=DtStr+aValue.AsString+"; "
 end
end

MsgBox.Report("vom Thema"++aTStr+","+NL+
    "um eine Polyline zum Teil zu korrigieren"+NL+NL+
    "Die Namen der Felder ohne Shape-Felder"
     +NL+FldStr+NL+NL+
    "Der erste Datensatz:"+NL+DtStr,
```

```
    "Information")
KWFld = MsgBox.ListAsString(aListofFlds,
    "des Themas"++aTStr+NL+
   "für Bezeichnung der Datensäze und"+NL+
   "für Klassifizierung der Legende",
   "Auswahl eines Feldes")

KWFldStr = KWFld.AsString

aLegend = AkTheme.GetLegend
aorgLTyp = aLegend.GetLegendType

'MsgBox.Info(aorgLTyp.AsString, "eigentliche Legende")
aListofSymbol = aLegend.GetSymbols
AnzSymb = aListofSymbol.Count
AnzSymbIdx = AnzSymb-1
ListoforgColor = {}

for each symb in 0..AnzSymbIdx
   aorgColor = aListofSymbol.Get(symb).GetColor
   ListoforgColor.Add(aorgColor)
end

aLegend.SetLegendType(#Legend_Type_Unique)
aLegend.Unique(AkTheme, KWFldStr)
ListofKlasse = aLegend.GetClassifications
AnzKlasse = ListofKlasse.Count
IdxKlasse = AnzKlasse-1
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")

ListofKlBez = {}
for each i in 0..IdxKlasse
  theKlasseLb=ListofKlasse.Get(i).GetLabel
  ListofKlBez.Add(theKlasseLb)
end

theKlBz=MsgBox.ListAsString(ListofKlBez,
   "im"++aTStr++"zur Auswahl der Polyline",
   "Auswahl einer Klasse der Datensätze")
if (theKlBz = nil) then
  MsgBox.Error("Die Klasse der Polyline oder"
     +NL+"die Polyline mit der Klasse fehlt!"
     +NL+"Das Programm wird abgebrochen!", "")
  exit
end

'Auswahl der Datensätze im Thema mit den Maus-Klicken
ListofSBz = {}
ListofIdxS = {}

if (AkTheme.CanSelect) then
 'MsgBox.Info("Das Thema"++aTStr++"ist selectierbar.","CanSelect?")
 for each rec in 0..AnzMIdx
    aMCPg = ListofMCPg.Get(rec)
    if (rec = 0) then
       AkTheme.SelectByPolygon(aMCPg, #VTAB_SELTYPE_NEW)
    elseif (rec > 0) then
       AkTheme.SelectByPolygon(aMCPg, #VTAB_SELTYPE_OR)
    end
 end
 'die ausgewählten Datensätze
 AnzS = 0
```

```
  for each rec in AkFTab.GetSelection
    'MsgBox.Info("von"++aTStr++":  "++rec.AsString,
    '    "Index-Nummer der selektierten Datensätze")
    aSBz = AKFTab.ReturnValue(KWFld, rec)
    ListofSBz.Add(aSBz)
    AnzS = AnzS + 1
    ListofIdxS.Add(rec.AsString)
  end
  'MsgBox.Info(AnzS.AsString, "Anzahl der selektierten Datensätze")
  'MsgBox.ListAsString(ListofIdxS, "Index-Nummer der selektierten"
  '                ++"Datensätze"++"von"++aTStr, "Information")
  if (AnzS = 0) then
    for each i in 0..IdxDs
      aBz = AkFTab.ReturnValue(KWFld, i)
      if (aBz = theKlBz) then
        ListofSBz.Add(aBz)
        ListofIdxS.Add(i.AsString)
      end
    end
  end
else
  for each i in 0..IdxDs
    aBz = AkFTab.ReturnValue(KWFld, i)
    if (aBz = theKlBz) then
       ListofSBz.Add(aBz)
       ListofIdxS.Add(i.AsString)
    end
  end
end

aSBz = MsgBox.ListAsString(ListofSBz,
       "von"++aTStr++","++
       "um die Polyline zum Teil zu Korrigieren",
       "Auswahl eines Datensatzes")
aSBzIdx = ListofSBz.FindByValue(aSBz)
aSRIdx = ListofIdxS.Get(aSBzIdx).AsNumber

'MsgBox.Info("von"++aTStr++":  "++aSRIdx.AsString,
'     "Index des ausgewählten Datensatzes")
theShp = AkFTab.ReturnValue(AkShpFld, aSRIdx)
'MsgBox.report("des Themas"++aTStr+NL+
'             "Index-Nummer:"++aSRIdx.AsString+NL+NL+
'             theShp.AsString, "Selektiertes Shape")

AkTheme.ClearSelection

'Feststellung der Koordinaten der ausgewählten Figur im Thema

theProfilList1={}
theProfilList1.Add({theShp})

'Liste der Stützpunkte
ListofFLPt = {}
minxkrd = 100000000
minykrd = 100000000
maxxkrd = 0
maxykrd = 0

if (theProfilList1 <> 0) then
  for each q in theProfilList1
     theLines=q.Get(0).AsList
     for each m in theLines
```

```
        for each ptx in m
          myx=ptx.Getx
          myy=ptx.Gety
          ListofFLPt.Add(myx@myy)
          if (myx < minxkrd) then
            minxkrd = myx
          end
          if (myx > maxxkrd) then
            maxxkrd = myx
          end
          if (myy < minykrd) then
            minykrd = myy
          end
          if (myy > maxykrd) then
            maxykrd = myy
          end
        end
      end
    end
end

PtAnz= ListofFLPt.Count  'Anzahl der Stützpunkte der Polyline
PtAnzIdx= PtAnz-1

'Bestimmung der Stellen an Maus-Klicken

ListofVt = {}
ListofminIdx = {}

for each aM in 0..AnzMIdx
   minDist=1000
   minIdx=-1

   aMPt = ListofMPt.Get(aM)
   aMPtx = aMPt.Getx
   aMPty = aMPt.Gety

   if (aMPtx < minxkrd) then
     aMPtx = minxkrd
   elseif (aMPtx > maxxkrd) then
     aMPtx = maxxkrd
   end
   if (aMPty < minykrd) then
     aMPty = minykrd
   elseif (aMPty > maxykrd) then
     aMPty = maxykrd
   end

   for each i in 0..PtAnzIdx
     aPt = ListofFLPt.Get(i)
     xkrd= aPt.Getx
     ykrd= aPt.Gety
     xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
     yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
     xyAbst = ((xAbst  + yAbst).sqrt) .Abs
     if (xyAbst < minDist) then
       minDist = xyAbst
       minIdx = i  'Index des nähesten Punktes des Maus-Klickens
     end
   end

   aVt = ListofFLPt.Get(minIdx)  'ein Stützpunkt der Polyline
```

```
                    'an einem Maus-Klicken
  ListofVt.Add(aVt)
  ListofminIdx.Add(minIdx)
end

'Bestimmung der Höhe des Abschnittes
ListofEing = {"der Stützpunkt der Polyline",
          "ein Punkt auf der Polyline",
          "ein Punkt einer Tastatur-Eingabe",
          "ein Punkt an einem Fadenkreuz"}

ErstPt = MsgBox.ListAsString(ListofEing,
      "um die Höhe des Abschnittes zu bestimmen.",
      "Auswahl eines Punktes am 1. Maus-Klicken")

if (ErstPt = "ein Punkt einer Tastatur-Eingabe") then
  aMPt = ListofMPt.Get(0)
  aMPtx = (aMPt.Getx.SetFormat("").SetFormat("d.dd")).AsString
  aMPty = (aMPt.Gety.SetFormat("").SetFormat("d.dd")).AsString
  aHxStr = MsgBox.Input("Eingabe der x-Koord. des 1."
        ++"Punktes des Abschnittes",
        "Tastaur-Eingabe", aMPtx)
  aHyStr = MsgBox.Input("Eingabe der Höhe (Faktor * [m])"
        ++"des 1. Punktes des Abschnittes",
        "Tastaur-Eingabe", aMPty)
  aHx = aHxStr.AsNumber
  aHy = aHyStr.AsNumber
  aNPt = Point.Make(aHx, aHy)

elseif (ErstPt = "der Stützpunkt der Polyline") then
  aNPt = ListofVt.Get(0)

elseif (ErstPt = "ein Punkt auf der Polyline") then
  minIdx = ListofminIdx.Get(0)

 if (minIdx = 0) then
   vIdx = 0     'ein Vertex auf der Linie vor dem Maus-Klicken
   nIdx = 1     'ein Vertex auf der Linie nach dem Maus-Klicken
 elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
   aPtv = ListofFLPt.Get((minIdx - 1))
   aPt0 = ListofFLPt.Get(minIdx)
   aPtn = ListofFLPt.Get((minIdx + 1))

   xkrdv = aPtv.Getx   'x-Koord. des letzten Punktes
   xkrd  = aPt0.Getx   'x-Koord. des nähesten Punktes
   xkrdn = aPtn.Getx   'x-Koord. des nächsten Punktes

   ykrdv = aPtv.Gety   'y-Koord. des letzten Punktes
   ykrd  = aPt0.Gety          'y-Koord. des nähesten Punktes
   ykrdn = aPtn.Gety   'y-Koord. des nächsten Punktes

   xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
   yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
   xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten
                          'Punkt vom nähesten Punkt
   xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
   yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
   xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten
                          'Punkt vom nähesten Punkte
   xML = (xkrd - aMPtx) * (xkrd - aMPtx)
   yML = (ykrd - aMPty) * (ykrd - aMPty)
   xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Maus-Klicken
```

```
                        'vom nähesten Punkt
  xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
  yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
  xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Maus-Klicken
                          'vom letzten Punkt
  xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
  yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
  xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Maus-Klicken
                          'vom nächsten Punkt
  vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem
  vLpML = xyAv + xyML        'Maus-Klicken und dem letzten Punkt

  nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen dem
  nLpML = xyAn + xyML        'Maus-Klicken und dem nächsten Punkt

  vM = (vLmML - xyMLv).Abs   'Differenz zwischen der theorischen
  vP = (vLpML - xyMLv).Abs   'und der tatsächlichen Länge im Bezug
                      'auf den letzten Punkt
  nM = (nLmML - xyMLn).Abs   'Differenz zwischen der theorischen
  nP = (nLpML - xyMLn).Abs   'und der tatsächlichen Länge im Bezug
                      'auf den nächsten Punkt
  ListofLg = {vM, vP, nM, nP}   'Suche nach der kleinsten Länge
  MinLg = 10000
  TheLgIdx = -1
  For each aLg in 0..3
     theLg = ListofLg.Get(aLg)
     if (theLg < MinLg) then
          MinLg = theLg
          TheLgIdx = aLg
     end
  end
  if (xyML < 10) then
    vIdx = minIdx - 1
    nIdx = minIdx + 1
  elseif (xyML >= 10) then
    if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
        vIdx = minIdx - 1 'ein Vertex auf der Linie vor dem Maus-Klicken
        nIdx= minIdx     'ein Vertex auf der Linie nach dem Maus-Klicken
    elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
        vIdx = minIdx    'ein Vertex auf der Linie vor dem Maus-Klicken
        nIdx = minIdx + 1 'ein Vertex auf der Linie nach dem Maus-Klicken
    else
        vIdx = minIdx     'ein Vertex auf der Linie vor dem Maus-Klicken
        nIdx = minIdx + 1 'ein Vertex auf der Linie nach dem Maus-Klicken
    end
  end
elseif (minIdx = PtAnzIdx)   then
   vIdx = minIdx - 1
   nIdx = minIdx
end

'Bestimmung der Koordinaten des Maus-Klickens auf dem Profilschnitt
vPt = ListofFLPt.Get(vIdx)
vPtx = vPt.Getx
vPty = vPt.Gety
nPt = ListofFLPt.Get(nIdx)
nPtx = nPt.Getx
nPty = nPt.Gety

if ((minDist = 0) or (minDist < 10)) then 'Der näheste Punkt liegt innerhalb
   mPt = ListofFLPt.Get(minIdx)
   Ptx = mPt.Getx    '10 m vom dem Maus-Klicken auf der Linie
```

```
    Pty = mPt.Gety
    if ( minIdx = 0) then
      vGrIdx = minIdx
      nGrIdx = minIdx + 1
    elseif (( minIdx > 0) and ( minIdx < PtAnzIdx)) then
      vGrIdx = minIdx -1
      nGrIdx = minIdx + 1
    elseif ( minIdx = PtAnzIdx) then
      vGrIdx = minIdx -1
      nGrIdx = minIdx
    end
  elseif (minDist >= 10) then  'Berechnung der Koordinaten des Punktes
    vGrIdx = vIdx      'auf der Linie als Projektion des Maus-Klickens
    nGrIdx = nIdx          'auf die Linie (Profilschnitt)
    if  (vPtx  = aMPtx) then
        Ptx = vPtx
        Pty = vPty
    elseif ((vPtx  <> aMPtx) and (aMPtx <> nPtx)) then
        if (vPty = nPty) then
              Ptx = aMPtx
              Pty = vPty
        elseif (vPty < nPty) then
              if (nPtx = vPtx) then
                    Ptx = nPtx
                    Pty = nPty
              elseif (nPtx <> vPtx) then
                    xnvDiff = nPtx - vPtx
                    ynvDiff = nPty - vPty
                    xMvDiff = aMPtx - vPtx
                    Ptx = aMPtx
                    Pty  = (ynvDiff / xnvDiff) * xMvDiff + vPty
              end
        elseif (vPty > nPty) then
              if (nPtx = vPtx) then
                    Ptx = nPtx
                    Pty = nPty
              elseif (nPtx <> vPtx) then
                    xnvDiff = nPtx - vPtx
                    yvnDiff = vPty - nPty
                    xnMDiff = nPtx - aMPtx
                    Ptx = aMPtx
                    Pty  = (yvnDiff / xnvDiff) * xnMDiff + nPty
              end
        end
    elseif (nPtx  = aMPtx)  then
        Ptx = nPtx
        Pty = nPty
    end
  end          ' Ende von (if ((minDist = 0) or (minDist < 10)) then)
  aNPt = Point.Make(Ptx, Pty)

elseif (ErstPt = "ein Punkt an einem Fadenkreuz") then
   theThemes = AkView.GetThemes
   ListofPLThms = {}

   for each aT in theThemes
    if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(PolyLine)) then
        ListofPLThms.Add(aT)
      end
    end
```

```
end

KrThm = MsgBox.ListAsString(ListofPLThms,
      "zur Bestimmung der Stelle und Höhe des Abschnitt-Anfangs",
      "Auswahl eines Fadenkreuz-Themas")
KrFTab = KrThm.GetFTab
KrShpFld=KrFTab.FindField("Shape")

AnzRec = 0
for each rec in KrFTab
   AnzRec = AnzRec+1
end
AnzRecIdx = AnzRec-1
'MsgBox.Info(AnzRec.AsString,
'   "Anzahl der Datensätze im Thema (Fadenkreuz)"++KrThm.AsString)
ListofKrPtx = {}
ListofKrPty = {}

for each rec in 0..AnzRecIdx
   aPL=KrFTab.ReturnValue(KrShpFld, rec)
   ListofKrPL = {}
   ListofKrPL.Add({aPL})
   for each q in ListofKrPL
      theLs = q.Get(0).AsList
      for each L in theLs
         for each apt in L
            aptx = apt.Getx
            apty = apt.Gety
            ListofKrPtx.Add(aptx)
            ListofKrPty.Add(apty)
         end
      end
   end
end

AnzKrPtx = ListofKrPtx.Count
IdxKrPtx = AnzKrPtx - 1

for each j in 0..IdxKrPtx
   aKrx1 = ListofKrPtx.Get(j)
   aKry1 = ListofKrPty.Get(j)
   xZ = 0
   yZ = 0
   for each i in 0..IdxKrPtx
      aKrx2 = ListofKrPtx.Get(i)
      aKry2 = ListofKrPty.Get(i)
      if (aKrx1 = aKrx2) then
        xZ = xZ + 1
      end
      if (aKry1 = aKry2) then
        yZ = yZ + 1
      end
   end
   if (xZ = 2) then
     theX = aKrx1
   end
   if (yZ = 2) then
     theY = aKry1
   end
end
aNPt = Point.Make(theX, theY)
```

```
  'MsgBox.Report("Der Punkt am Fadenkreuz"++"("+KrThm.AsString+"):"
  '             ++aNPt.AsString,
  '             "Die Koordinaten der Punkte an dem Fadenkreuz")
end


'Bestimmung des Teilabschnittes mit den 2. und 3. Maus-Klicken
ListofTPt = {}
IdxA = ListofminIdx.Get(1)
IdxE = ListofminIdx.Get(2)
for each i in 0..PtAnzIdx
   if ((i >= IdxA) and (i <= IdxE)) then
      aPt = ListofFLPt.Get(i)
      ListofTPt.Add(aPt)
   end
end

'Bestimmung des Punktes am Ende des Abschnittes
AnzTP = ListofTPt.Count
IdxTP = AnzTP - 1

aNPtx = aNPt.Getx
aNPty = aNPt.Gety

for each i in 0..IdxTP
  aPt = ListofTPt.Get(i)
  xKrd = aPt.Getx
  yKrd = aPt.Gety
  if (i < IdxTP) then
     aPt2 = ListofTPt.Get(i+1)
     xKrd2 = aPt2.Getx
     yKrd2 = aPt2.Gety
     if ((aNPty = yKrd) and (aNPty = yKrd2)) then
        theNx2 = xKrd
     elseif (((aNPty >= yKrd) and (aNPty < yKrd2)) or
           ((aNPty > yKrd2) and (aNPty <= yKrd))) then
        if (xKrd = xKrd2) then
           theNx2 = xKrd
        elseif (xKrd < xKrd2) then
           xD = xKrd2 - xKrd
           yD = yKrd2 - yKrd
           theNx2 = ((xD/yD) * (aNPty-yKrd)) + xKrd
        elseif (xKrd > xKrd2) then
           xD = xKrd - xKrd2
           yD = yKrd2 - yKrd
           theNx2 = xKrd - ((xD/yD) * (aNPty-yKrd))
        end
     end
  elseif (i = IdxTP) then
     if (yKrd = aNPty) then
        theNx2 = xKrd
     end
  end
end

Pt2Ab = Point.Make(theNx2, aNPty)
ListofPtAb = {aNPt,Pt2Ab}

'Bestimmung des Abschnittes als Index des Stützpunktes
'mit dem kleinsten Abstand von den Maus-Klicken
ListofGrPt = {}
ListofListofGrIdx = {}
```

```
for each aAPt in 0..1   ' Anfang der Schleife für einen Abschnitt
   minDist=1000
   minIdx=-1

   aMPt = ListofPtAb.Get(aAPt)
   aMPtx = aMPt.Getx
   aMPty = aMPt.Gety

   for each i in 0..PtAnzIdx
     aPt = ListofFLPt.Get(i)
     xkrd= aPt.Getx
     ykrd= aPt.Gety
     xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
     yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
     xyAbst = ((xAbst  + yAbst).sqrt) .Abs
     if (xyAbst < minDist) then
       minDist = xyAbst
       minIdx = i  'Index des nähesten Punktes des Maus-Klickens
     end
   end

   'Bestimmung des Abschnittes des Shapes
   'als x-, y-Koordinaten

   if (minIdx = 0) then
     vIdx = 0     'ein Vertex auf der Linie vor dem Maus-Klicken
     nIdx = 1     'ein Vertex auf der Linie nach dem Maus-Klicken
   elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
     aPtv = ListofFLPt.Get((minIdx - 1))
     aPt0 = ListofFLPt.Get(minIdx)
     aPtn = ListofFLPt.Get((minIdx + 1))

     xkrdv = aPtv.Getx   'x-Koord. des letzten Punktes
     xkrd  = aPt0.Getx   'x-Koord. des nähesten Punktes
     xkrdn = aPtn.Getx   'x-Koord. des nächsten Punktes

     ykrdv = aPtv.Gety   'y-Koord. des letzten Punktes
     ykrd  = aPt0.Gety          'y-Koord. des nähesten Punktes
     ykrdn = aPtn.Gety   'y-Koord. des nächsten Punktes

     xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
     yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
     xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten Punkt
                                    'vom nähesten Punkt
     xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
     yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
     xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten Punkt
                                    'vom nähesten Punkte
     xML = (xkrd - aMPtx) * (xkrd - aMPtx)
     yML = (ykrd - aMPty) * (ykrd - aMPty)
     xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Maus-Klicken
                                    'vom nähesten Punkt
     xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
     yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
     xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Maus-Klicken
                                       'vom letzten Punkt
     xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
     yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
     xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Maus-Klicken
                                       'vom nächsten Punkt
     vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem
```

```
  vLpML = xyAv + xyML        'Maus-Klicken und dem letzten Punkt

  nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen dem
  nLpML = xyAn + xyML        'Maus-Klicken und dem nächsten Punkt

  vM = (vLmML - xyMLv).Abs 'Differenz zwischen der theorischen und
  vP = (vLpML - xyMLv).Abs 'der tatsächlichen Länge im Bezug auf
                    'den letzten Punkt
  nM = (nLmML - xyMLn).Abs  'Differenz zwischen der theorischen und
  nP = (nLpML - xyMLn).Abs  'der tatsächlichen Länge im Bezug auf
                     'den nächsten Punkt
  ListofLg = {vM, vP, nM, nP}   'Suche nach der kleinsten Länge
  MinLg = 10000
  TheLgIdx = -1
  For each aLg in 0..3
     theLg = ListofLg.Get(aLg)
     if (theLg < MinLg) then
          MinLg = theLg
          TheLgIdx = aLg
     end
  end
  if (xyML < 10) then
    vIdx = minIdx - 1
    nIdx = minIdx + 1
  elseif (xyML >= 10) then
   if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
    vIdx = minIdx - 1  'ein Vertex auf der Linie vor dem Maus-Klicken
    nIdx= minIdx   'ein Vertex auf der Linie nach dem Maus-Klicken
   elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
    vIdx = minIdx   'ein Vertex auf der Linie vor dem Maus-Klicken
    nIdx = minIdx + 1  'ein Vertex auf der Linie nach dem Maus-Klicken
   else
    vIdx = minIdx    'ein Vertex auf der Linie vor dem Maus-Klicken
    nIdx = minIdx + 1  'ein Vertex auf der Linie nach dem Maus-Klicken
   end
  end
elseif (minIdx = PtAnzIdx)   then
   vIdx = minIdx - 1
   nIdx = minIdx
end

'Bestimmung der Koordinaten des Maus-Klickens auf dem Profilschnitt
vPt = ListofFLPt.Get(vIdx)
vPtx = vPt.Getx
vPty = vPt.Gety
nPt = ListofFLPt.Get(nIdx)
nPtx = nPt.Getx
nPty = nPt.Gety

if ((minDist = 0) or (minDist < 10)) then 'Der näheste Punkt liegt
   mPt = ListofFLPt.Get(minIdx)          'innerhalb 10 m vom
   Ptx = mPt.Getx                'dem Maus-Klicken auf der Linie
   Pty = mPt.Gety
   if ( minIdx = 0) then
     vGrIdx = minIdx
     nGrIdx = minIdx + 1
   elseif (( minIdx > 0) and ( minIdx < PtAnzIdx)) then
     vGrIdx = minIdx -1
     nGrIdx = minIdx + 1
   elseif ( minIdx = PtAnzIdx) then
     vGrIdx = minIdx -1
     nGrIdx = minIdx
```

```
      end
    elseif (minDist >= 10) then  'Berechnung der Koordinaten des Punktes
      vGrIdx = vIdx      'auf der Linie als Projektion des Maus-Klickens
      nGrIdx = nIdx          'auf die Linie (Profilschnitt)
      if  (vPtx  = aMPtx) then
          Ptx = vPtx
          Pty = vPty
      elseif ((vPtx  <> aMPtx) and (aMPtx <> nPtx)) then
          if (vPty = nPty) then
                Ptx = aMPtx
                Pty = vPty
          elseif (vPty < nPty) then
                if (nPtx = vPtx) then
                     Ptx = nPtx
                     Pty = nPty
                elseif (nPtx <> vPtx) then
                     xnvDiff = nPtx - vPtx
                     ynvDiff = nPty - vPty
                     xMvDiff = aMPtx - vPtx
                     Ptx = aMPtx
                     Pty  = (ynvDiff / xnvDiff) * xMvDiff + vPty
                end
          elseif (vPty > nPty) then
                if (nPtx = vPtx) then
                     Ptx = nPtx
                     Pty = nPty
                elseif (nPtx <> vPtx) then
                     xnvDiff = nPtx - vPtx
                     yvnDiff = vPty - nPty
                     xnMDiff = nPtx - aMPtx
                     Ptx = aMPtx
                     Pty  = (yvnDiff / xnvDiff) * xnMDiff + nPty
                end
          end
      elseif (nPtx  = aMPtx)  then
          Ptx = nPtx
          Pty = nPty
      end
    end          ' Ende von (if ((minDist = 0) or (minDist < 10)) then)
    ListofGrPt.Add(Ptx@Pty)       'Ein Grenz-Punkt für den Abschnitt

    ListofGrIdx = {}
    ListofGrIdx.Add(vGrIdx)
    ListofGrIdx.Add(nGrIdx)
    ListofListofGrIdx.Add(ListofGrIdx)

end         ' Ende der Schleife für Abschnitt

'Ersetzen des Profilschnittes
ListofAbschnV = ListofListofGrIdx.Get(0)
vIdxAbschnV = ListofAbschnV.Get(0)
nIdxAbschnV = ListofAbschnV.Get(1)

ListofAbschnN = ListofListofGrIdx.Get(1)
vIdxAbschnN = ListofAbschnN.Get(0)
nIdxAbschnN = ListofAbschnN.Get(1)

PtAbschnV = ListofGrPt.Get(0)
PtAbschnN = ListofGrPt.Get(1)

ListofNPt = {}
for each i in 0..PtAnzIdx
```

```
  if (i <= vIdxAbschnV) then
    aPt = ListofFLPt.Get(i)
    ListofNPt.Add(aPt)
  end
end

ListofNPt.Add(PtAbschnV)
theYKrd = Pt2Ab.Gety

for each i in 0..PtAnzIdx
  if ((i >= nIdxAbschnV) and (i <= vIdxAbschnN)) then
    aPt = ListofFLPt.Get(i)
    aPtx = aPt.Getx
    ListofNPt.Add(aPtx@theYKrd)
  end
end
ListofNPt.Add(PtAbschnN)

for each i in 0..PtAnzIdx
  if (i >= nIdxAbschnN) then
    aPt = ListofFLPt.Get(i)
    ListofNPt.Add(aPt)
  end
end

ListofListofNPoint={}
ListofListofNPoint.Add(ListofNPt)
theNShp = Polyline.Make(ListofListofNPoint)

AkFTab.SetEditable(false)
AkFTab.SetEditable(true)
AkFTab.SetValue(AkShpFld, aSRIdx, theNShp)
AkFTab.SetEditable(false)

av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

'Definition der Farbe der Legende
'Bezeichnung der Datensätze
aListofdfm = {"---",20,41, "a",43,6,  "a/Mj",24,6,  "Aussen",5,0,
    "d",36,38, "f",8,13, "Gy",24,36,  "H",32,43, "hg/plRR",3,35,
    "Hj",32,43, "Hn",11,34, "Lf",17,8, "Lfh/N",14,6, "Lö",47,8,
    "Lö/Hj",30,27, "Lö/Mj",24,29, "Löy",6,32,  "Ma",29, 28, "mi-olK",35,4,
    "miIV",11, 4, "Mj",26, 28, "N",16, 46, "plRR",53, 7, "Rhein",20,44,
    "Sf",18,5, "Sfh/N",16,5,  "sSo",24,41, "tAb",53,38, "tTt",51,35,
    "Deck",53,38,  "D",53,38, "De",53,38, "NT",14,46, "MT",26,28,
    "HT",32,43, "Präq",3,5, "Präm",3,5,  "GH",53,38, "NA",5,28, "NQ",8,5,
    "TOK",14,23, "NTabF",5,28, "MTabF",8,5, "QB",8,5, "QmitD",20,10,
    "QohneD",26,23, "QohneT",8,21, "TrorAe",3,5,  "UMT",26,28,
    "UMT III",26,28, "UMT IV",26,33, "OMT",41,39, "rMTl",20,28,
    "lMTl",26,28, "rHTl",32,43, "lNT",14,46, "rNT",17,46, "lHTr",32,43,
    "rHTr",32,43, "HTr",32,43, "MTl",26,28, "MTr",20,28, "HTl",32,43,
    "Deck-Mu",53,38, "Deck-LS",47,23, "UMT2",26,28, "Holst-Sd",38,8,
    "Holst-Tf",35,34, "UMT1",41,35, "Holst-Ton",23,15, "Holst-U",44,7,
    "SdScht-HR",45,9, "MMT-R",20,33, "Geländeoberfläche",53,38,
    "Basisfläche der Deckschichten (TOK)",14,46, "NT abgedeckte Fläche",5,28,
    "MT abgedeckte Fläche",8,5, "Quartärbasis",8,5,
    "Oberfläche der als MT ältere Schichten",8,5,
    "Oberfläche der Präquartär-Schichten",8,5,
    "Deckschichten",53,38, "Niederterrassen",14,46, "Mittelterrassen",26,28,
    "Präquartäre Schichten",3,5,
    "Deckschichten (Mutter oder Waldboden)",53,38,
```

```
    "Deckschichten (Lehm oder Sand)",47,23, "Untere Mittelterrasse 2",26,28,
    "Untere Mittelterrasse 1",41,35, "Holstein (Torf, z.T. Tonhaltig)",35,34,
    "Holstein (Sandschichten mit Tonlagen)",38,8,
    "Holstein (Tonschichten)",23,15, "Holstein (Schluffschichten)",44,7,
    "Sandschichten (z.T. Holstein, z.T. Rinnenschotter)",45,9,
    "Die mittlere Mittelterrasse (Rinnenschotter)",20,33,
    "Untere Mittelterrasse",26,28, "Obere Mittelterrasse",41,39,
    "Hauptterrasse",32,43, "als MT ältere Schichten",3,5,
    "Präquartär-Schichten",3,5, "Untere Mittelterrasse III",26,33,
    "Untere Mittelterrasse IV",26,28, "Sonst",0,0}

Legendeinfo = 0

if (aorgLTyp = #LEGEND_TYPE_SIMPLE) then
   theLegend = AkTheme.GetLegend
   theLegend.SetLegendType(#LEGEND_TYPE_SIMPLE)
   theLegend.SingleSymbol
   theSymbol=theLegend.GetSymbols.Get(0)
   theSymbol.SetColor(ListoforgColor.Get(0))
else
   av.ShowMsg("Veränderung der Legenden des Themas"
     ++AkTheme.AsString+"...")

   theLegend = AkTheme.GetLegend
   theLegend.SetLegendType(#Legend_Type_Unique)
   theLegend.Unique(AkTheme, KWFldStr)
   ListofKlasse=theLegend.GetClassifications
   AnzKlasse=ListofKlasse.Count
   IdxKlasse=AnzKlasse-2
   'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")

   aListofSColor = {}
   for each i in 0..IdxKlasse
     theKlasseLb = ListofKlasse.Get(i).GetLabel
     'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
     aIdxLb = aListofdfm.FindByValue(theKlasseLb)
     if (aIdxLb <> -1) then
       aCNr = aListofdfm.Get((aIdxLb + 1))
     elseif (aIdxLb = -1) then
       aR = i Mod 60
       if (aR = 0) then
         aCNr = 1
       elseif (aR > 0) then
         aCNr = aR
       end
       Legendeinfo = 1
     end
     theColor=(thePalette.GetList(#PALETTE_LIST_COLOR).Get(aCNr))
     theRgbList=theColor.GetRgbList
     aColor=Color.Make
     aColor.SetRgbList(theRgbList)
     aListofSColor.Add(aColor)
   end

   aListofSymbol=theLegend.GetSymbols
   AnzSymb=aListofSymbol.Count
   AnzSymbIdx=AnzSymb-2

   for each symb in 0..AnzSymbIdx
     aListofSymbol.Get(symb).SetColor(aListofSColor.Get(symb))
   end
   if (Legendeinfo = 1) then
```

```
      MsgBox.Report("Die Farbe der Legende ist zum Teil"
        +NL+"oder gar nicht definiert!"
        +NL+"Die Definition der Farbe"
        +NL+"in diesem Programm ist zu ändern.",
      "Information")
    end
end

AkTheme.UpdateLegend




'plzaustm.ave
'Ein neues 3D-Thema mit den Entfernungsprofilschnitten als PolyLineZ
'wird in einer aktiven 3D-Szene hergestellt. Punkte mit Höhenwerten
'auf einer Entfernungsprofilschnittlinie in einem Thema in einem
'Karten-View werden als Eingabe-Daten benutzt.
'Ein Menü oder eine Schaltfläche in einem aktiven Karten-View
'zum Anklicken.

theProject=av.GetProject
theSzene = av.GetActiveDoc    'eine aktive 3D-Szene

ListofView={"Kt1_gg", "Kt1_ggd", "Kt1_hg",
           "Kt1_hgd", "Kt1_tga", "Kt1_tgd"}
KtViewStr=MsgBox.ChoiceAsString(ListofView,
          "um ein Punkt-Thema zur Bildung eines"
      ++"Entfernungsprofilschnittes auszuwählen",
          "Auswahl eines Views für Kartendarstellung")
theKtView=theProject.FindDoc(KtViewStr)

ListofKtThms=theKtView.GetThemes

ListofPtThms = {}
for each aT in ListofKtThms
    if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofPtThms.Add(aT)
      end
    end
end

av.ShowMsg("Eingabe eines Punkt-Themas und"
      ++"Feststellung der Anzahl der Punkte in dem Thema ...")

PtTheme=MsgBox.ChoiceAsString(ListofPtThms,
       "um die Punkte in PolyLineZ umzuwandeln.", "Auswahl eines Punkt-Themas")
PtFTab=PtTheme.GetFTab
PtShpFld=PtFTab.FindField("Shape")

ListofFlds=PtFTab.GetFields
MsgBox.ListAsString(ListofFlds, "die im Punkt-Thema"
       ++PtTheme.GetName++"enthalten sind", "Anzeige der Felder")

AnzHFldStr=MsgBox.Input("die die Höhen der Punkte enthalten",
          "Anzahl der Felder", "4")
if (AnzHFldStr = nil) then
   MsgBox.Info("Die Anzahl der Felder für Höhen"+NL+"im Thema"
```

```
      ++PtTheme.GetName+"ist noch festzustellen!", "Information")
  exit
end

AnzHFld=AnzHFldStr.AsNumber.SetFormat("d")
IdxHFld=AnzHFld-1
ListofHFlds={}

for each i in 0..IdxHFld
  Nr=i+1
  PtHFld=MsgBox.ListAsString(ListofFlds, "für"++Nr.AsString+". Höhe",
       "Auswahl eines Feldes im Thema"++PtTheme.GetName)
  ListofHFlds.Add(PtHFld)
end

AnzPt=0
for each Pt in PtFTab
   AnzPt=AnzPt+1
end
AnzPtIdx=AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"++PtTheme.AsString)

ListofPLZ={}
for each j in 0..IdxHFld
  aHFld=ListofHFlds.Get(j)
  Listof3DPt={}
  ListofListof3DPt={}
  for each i in 0..AnzPtIdx
    aPT=PtFTab.ReturnValue(PtShpFld, i)
    aH=PtFTab.ReturnValue(aHFld, i)
    Listof3DPt.Add(aPT@aH)
  end
  ListofListof3DPt.Add(Listof3DPt)
  thePLZ=PolylineZ.Make(ListofListof3DPt)
  ListofPLZ.Add(thePLZ)
end

av.ShowMsg("Speicherung der hergestellten PolyLineZ ...")
'Ein Feature-Shape-File für Grenze (PolyLineZ) wird hergestellt.
theWDStr = theProject.GetWorkDir.AsString
DName=PtTheme.AsString.Left(6)
fnStr=FileName.Make(theWDStr).MakeTmp(DName,"shp")
fnGrMt=FileDialog.Put(fnStr, "*.shp", "Output 3D shape File (PolyLineZ)")
if (fnGrMt=nil) then exit end
fnGrMt.SetExtension("shp")
GrMtFTab=FTab.MakeNew(fnGrMt, PolyLineZ)
ShapeFld=GrMtFTab.FindField("shape")
IDFld=Field.Make("ID", #Field_Short, 2, 0)
NamenFld=Field.Make("Namen", #Field_Char, 10, 0)
ListofFlds1={IDFld, NamenFld}
GrMtFTab.AddFields(ListofFlds1)

GrMtFTab.SetEditable(false)
GrMtFTab.SetEditable(true)

for each i in 0..IdxHFld
  thePolyLZ=ListofPLZ.Get(i)
  aHFldStr=ListofHFlds.Get(i).AsString
  GrMtFTab.AddRecord
  GrMtFTab.SetValue(ShapeFld, i, thePolyLZ)
  GrMtFTab.SetValue(IDFld, i, i)
  GrMtFTab.SetValue(NamenFld, i, aHFldStr)
```

```
end

GrMtFTab.SetEditable(false)
thmNew=FTheme.Make(GrMtFTab)
theSzene.AddTheme(thmNew)



'ptabstlg.ave
'Punkte werden in einem bestimmten Abstand an den beiden Seiten einer
'Polyline oder eines Polygons hergestellt. Man kann nach der
'Durchführung dieses Programms nur die gewünschten Punkte bleiben
'lassen und die anderen Punkte löschen.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("ptabstlg")
myScript.SetNumberFormat( "d.dd") ' script default

PLTheme = theView.GetActiveThemes.Get(0)

kt00 = MsgBox.YesNo("Ist das aktive Thema"++PLTheme.AsString
     +NL+"zur Herstellung der Punkte richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++PLTheme.AsString++"ist falsch!"+NL+
           "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

PLFTab = PLTheme.GetFTab
PLShpFld = PLFTab.FindField("Shape")
PLIDFld = PLFTab.FindField("ID")
aShpClassName = PLFTab.GetShapeClass.GetClassName

AnzPLRec = 0                'Anzahl der Datensätze im Thema
for each i in PLFTab
  AnzPLRec = AnzPLRec +1
end
IdxPLRec = AnzPLRec -1
'MsgBox.Info(AnzPLRec.AsString, "Anzahl der Datensätze im Thema"
'      ++ PLTheme.AsString)

'Auswahl eines Datensatzes
ListofPLFlds = PLFTab.GetFields
AnzPLFld = ListofPLFlds.Count
IdxPLFld = AnzPLFld-1
ListofDtStr = {}
ListofFldStr = {}
FldStr = ""
for each j in 0..IdxPLRec
  DtStr=""
  for each i in 0..IdxPLFld
    aFld = ListofPLFlds.Get(i)
    aFldStr = aFld.GetName
    if (aFldStr <> "Shape") then
      FldStr = FldStr + aFldStr+"; "
      aValue = PLFTab.ReturnValue(aFld, j)
      DtStr = DtStr+aValue.AsString+"; "
```

```
    end
  end
  ListofFldStr.Add(FldStr)
  ListofDtStr.Add(DtStr)
end
aFldStr = ListofFldStr.Get(0)
aDtSStr = MsgBox.ListAsString(ListofDtStr,
     "im Thema"++ PLTheme.AsString+NL+
     "(Feldname: "++aFldStr+")", "Auswahl eines Datensatzes")
aDtSIdx = ListofDtStr.FindByValue(aDtSStr)
PLAusg = PLFTab.ReturnValue(PLShpFld, aDtSIdx)
'MsgBox.Report(PLAusg.AsString, "Das ausgewählte Shape")

'Umwandlung der PolyLine in die Liste der Stützpunkte
av.ShowMsg("Feststellung der Koordinaten der PolyLine in Thema"
        ++(PLTheme.AsString)++"...")

theProfilList = {}
theProfilList.Add({PLAusg })
ListofPt = {}
'2D-Shape wird in Liste der Punkte umgewandelt.
if (theProfilList <> 0) then
  for each q in theProfilList
      theLines = q.Get(0).AsList
      for each m in theLines
         for each apt in m
            ListofPt.Add(apt)
         end
      end
  end
end


'Berechnung der Punkte in einem bestimmten Abstand
'an den beiden Seiten einer Polyline

aAbstStr = MsgBox.Input("zwischen einer Polyline oder"++
                  "dem Polygon und den neuen Punkten",
                  "Eingabe eines Abstandes [m]", "100.00")
aAbst = aAbstStr.AsNumber

AnzPt = ListofPt.Count

if (aShpClassName = "Polyline") then
  IdxPt = AnzPt - 1
  aArt = "der Polyline"
elseif (aShpClassName = "Polygon") then
  IdxPt = AnzPt - 2
  aArt = "des Polylgons"

end

av.ShowMsg("Die neuen Punkte werden in einem Abstand von"++aAbstStr
    ++"[m] an den beiden Seiten"++aArt++"hergestellt ...")
av.ShowStopButton

ListofNPt = {}
Ng = 0

for each j in 0..IdxPt
  Ng = Ng + 1
  aPt = ListofPt.Get(j)
```

```
aPtx = aPt.Getx
aPty = aPt.Gety
if (j < IdxPt) then
  aPt2 = ListofPt.Get(j + 1)
  aPt2x = aPt2.Getx
  aPt2y = aPt2.Gety
  if (aPtx = aPt2x) then
    xN = aPtx + aAbst
    yN = aPty
    ListofNPt.Add(xN@yN)
    xN = aPtx - aAbst
    ListofNPt.Add(xN@yN)
  elseif (aPtx <> aPt2x) then
    xDiff = aPt2x - aPtx
    yDiff = aPt2y - aPty
    Neig2 = yDiff/xDiff
    if (Neig2 = 0) then
      xN = aPtx
      yN = aPty + aAbst
      ListofNPt.Add(xN@yN)
      yN = aPty - aAbst
      ListofNPt.Add(xN@yN)
    elseif (Neig2 <> 0) then
      Neig = (1/Neig2) * (-1)
      Achsab = aPty - (Neig * aPtx)
      aKrdAbs = ((aAbst ^ 2)/((Neig ^ 2) + 1)).Sqrt.Abs
      xN = aPtx + aKrdAbs
      yN = (Neig * xN) + Achsab
      ListofNPt.Add(xN@yN)
      xN = aPtx - aKrdAbs
      yN = (Neig * xN) + Achsab
      ListofNPt.Add(xN@yN)
    end
  end
elseif (j = IdxPt) then
  aPt = ListofPt.Get(j - 1)
  aPtx = aPt.Getx
  aPty = aPt.Gety
  aPt2 = ListofPt.Get(j)
  aPt2x = aPt2.Getx
  aPt2y = aPt2.Gety
  if (aPtx = aPt2x) then
    xN = aPt2x + aAbst
    yN = aPt2y
    ListofNPt.Add(xN@yN)
    xN = aPt2x - aAbst
    ListofNPt.Add(xN@yN)
  elseif (aPtx <> aPt2x) then
    xDiff = aPt2x - aPtx
    yDiff = aPt2y - aPty
    Neig2 = yDiff/xDiff
    if (Neig2 = 0) then
      xN = aPt2x
      yN = aPt2y + aAbst
      ListofNPt.Add(xN@yN)
      yN = aPt2y - aAbst
      ListofNPt.Add(xN@yN)
    elseif (Neig2 <> 0) then
      Neig = (1/Neig2) * (-1)
      Achsab = aPt2y - (Neig * aPt2x)
      aKrdAbs = ((aAbst ^ 2)/((Neig ^ 2) + 1)).Sqrt.Abs
      xN = aPt2x + aKrdAbs
```

```
          yN = (Neig * xN) + Achsab
          ListofNPt.Add(xN@yN)
          xN = aPt2x - aKrdAbs
          yN = (Neig * xN) + Achsab
          ListofNPt.Add(xN@yN)
        end
      end
    end 'Ende von if (j < IdxPt)
    'Show percentage complete with enabled stop button
    more = av.SetStatus((Ng/AnzPt)*100)
    if (not more) then
      break
    end
  end   'Ende von for each j in 0..IdxPt

  'Speichrung der neuen Punkte
  aWD = theProject.GetWorkDir.AsString
  afn00 = "Pt"+aAbstStr+"nt"
  afnStr = FileName.Make(aWD).MakeTmp(afn00, "shp")
  afname = FileDialog.Put(afnStr, "*.shp", "Output shape File (Point)")
  if (afname = nil) then exit end
  afname.SetExtension("shp")
  PtFTab = FTab.MakeNew(afname, Point)

  PtIdFld = Field.Make("ID", #Field_LONG, 6, 0)
  PtRWFld = Field.Make("RW", #Field_Float, 10, 2)
  PtHWFld = Field.Make("HW", #Field_Float, 10, 2)
  PtHmFld = Field.Make("Hoehem", #Field_Float, 7, 2)

  ListofPtFlds = {PtIdFld, PtRWFld, PtHWFld, PtHmFld}
  PtFTab.AddFields(ListofPtFlds)
  PtShpFld = PtFTab.FindField("Shape")

  AnzNPt = ListofNPt.Count
  IdxNPt = AnzNPt - 1

  PtFTab.SetEditable(false)
  PtFTab.SetEditable(true)

  for each i in 0..IdxNPt
    aPt = ListofNPt.Get(i)
    aPtx = aPt.Getx
    aPty = aPt.Gety
    PtFTab.AddRecord
    PtFTab.SetValue(PtShpFld, i, aPt)
    PtFTab.SetValue(PtIdFld, i, i)
    PtFTab.SetValue(PtRWFld, i, aPtx)
    PtFTab.SetValue(PtHWFld, i, aPty)
    PtFTab.SetValue(PtHmFld, i, 0)
  end

  PtFTab.SetEditable(false)

  PtTheme = FTheme.Make(PtFTab)
  theView.AddTheme(PtTheme)
  PtTheme.UpdateLegend
```

```
'ptaufghw.ave
'Punkte werden auf einem Polygon in einem bestimmten Bereich in einem
'bestimmten Abstand vom HW bestimmt und in einem Punkt-Thema gespeichert.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc    'Ein aktives Karten-View
myScript=theProject.FindScript("ptaufghw ")
myScript.SetNumberFormat( "d.dd") ' script default

PgTheme=theView.GetActiveThemes.Get(0)   'Ein aktives Pg-Thema
ThStr = PgTheme.AsString
kt00=MsgBox.YesNo("Ist das aktive Polygon-Thema"++ThStr
     +NL+"richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
            "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

PgFTab=PgTheme.GetFTab
ShpFld=PgFTab.FindField("Shape")
IDFld=PgFTab.FindField("Id")
ListofPgFlds = PgFTab.GetFields

AnzPg=0
for each rec in PgFTab
  AnzPg=AnzPg+1
end
IdxPg=AnzPg-1

'Feststellung der Datensätze im Polygon-Thema
AnzPgFld=ListofPgFlds.Count
IdxPgFld=AnzPgFld-1
ListofDtStr={}

for each j in 0..IdxPg
  DtStr=""
  for each i in 0..IdxPgFld
    aFld=ListofPgFlds.Get(i)
    aFldStr=aFld.GetName
    if (aFldStr <> "Shape") then
      aValue=PgFTab.ReturnValue(aFld, j)
      DtStr=DtStr+aValue.AsString+"; "
    end
  end
  ListofDtStr.Add(DtStr)
end
'Auswahl eines Polygons
aDtS = MsgBox.ListAsString(ListofDtStr,
    "um die Punkte auf dem Pg zu bestimmen",
    "Auswahl eines Polygons im Thema"++PgTheme.AsString)
aDtSIdx = ListofDtStr.FindByValue(aDtS)
theShape=PgFTab.ReturnValue(ShpFld, aDtSIdx)
Nr = aDtSIdx
AnzPgStr = AnzPg.SetFormat("").SetFormat("d").AsString
theKW = MsgBox.Input("für den"
      ++Nr.SetFormat("d").AsString+"."++"Datensatz"+NL+
      "("+aDts+")",
      "Eingabe eines Kennwortes", "MTro1")
```

```
ListofShapes={}
ListofShapes.Add({theShape})

'Erkennung der Koordinaten auf dem Polygon
av.ShowMsg("Berechnung der Koordinaten der Grenze im Thema"
          ++PgTheme.AsString++"...")
av.ShowStopButton
AnfHWStr=MsgBox.Input("Anfang des Hochwertes:"
     +NL+"(der große HW)","Eingabe des Grenzbereiches",
          "5641050.00")
EndHWStr=MsgBox.Input("Ende des Hochwertes:"
     +NL+"(der kleine HW)","Eingabe des Grenzbereiches",
          "5633500.00")
AnfHW=AnfHWStr.AsNumber
EndHW=EndHWStr.AsNumber
HWAbstStr00="50"
HWAbstStr =MsgBox.Input("zwischen Punkten von HW",
        "Eingabe eines Raster-Abstandes (m)", HWAbstStr00)
HWAbst=HWAbstStr.AsNumber
HWIdx=(((AnfHW-EndHW).Abs)/ HWAbst).SetFormat("").SetFormat("d")
ListoflGrPt={}
ListofrGrPt={}
ListofwGrPt={}
Ng=0
for each aHW in 0..HWIdx
   Ng=Ng+1
   theHW=AnfHW-(aHW* HWAbst)
   '2D Polygon wird in Liste der Koordinaten umgewandelt.
   if (ListofShapes <> 0) then
      for each aShp in ListofShapes
         aListofListofPts=aShp.Get(0).AsList
         ListIdx=0
         for each aList in aListofListofPts
            ListIdx=(ListIdx+1).SetFormat("").SetFormat("d")
            ListofPgHx={}
            ListofPgHy={}
            ListofaTPg={}
            ListofListofaTPg={}
            for each myPt in aList
               myx=myPt.Getx
               myy=myPt.Gety
               ListofPgHx.Add(myx)
               ListofPgHy.Add(myy)
               ListofaTPg.Add(myx@myy)
            end
            ListofListofaTPg.Add(ListofaTPg)
            thePg=Polygon.Make(ListofListofaTPg)
            AnzPgHx=ListofPgHx.Count
            PgHIndAnz=AnzPgHx-1
            for each Pt in 0..PgHIndAnz
               if (Pt < PgHIndAnz) then
                  xkrd1=ListofPgHx.Get(Pt)
                  ykrd1=ListofPgHy.Get(Pt)
                  xkrd2=ListofPgHx.Get(Pt+1)
                  ykrd2=ListofPgHy.Get(Pt+1)
                  if (ykrd1 > ykrd2) then
                     yRichtung="S"
                  elseif (ykrd2 > ykrd1) then
                     yRichtung="N"
                  elseif (ykrd1 = ykrd2) then
                     yRichtung="W"
                  end
```

```
  if (ykrd1 > ykrd2) then
    gykrd = ykrd1
    kykrd = ykrd2
  elseif (ykrd2 > ykrd1) then
    gykrd = ykrd2
    kykrd = ykrd1
  end
  if (xkrd1 > xkrd2) then
    gxkrd = xkrd1
    kxkrd = xkrd2
  elseif (xkrd2 > xkrd1) then
    gxkrd = xkrd2
    kxkrd = xkrd1
  end
  if (theHW = ykrd1) then
    theRW=xkrd1
    if (yRichtung = "S") then
      ListofrGrPt.Add(theRW@theHW)
    elseif (yRichtung ="N") then
      ListoflGrPt.Add(theRW@theHW)
    elseif (yRichtung = "W") then
      if (xkrd1 > xkrd2) then
        theRW=((xkrd1-xkrd2)/2)+xkrd2
      elseif (xkrd2 > xkrd1) then
        theRW=((xkrd2-xkrd1)/2)+xkrd1
        ListofwGrPt.Add(theRW@theHW)
      end
    end
  elseif ((theHW > kykrd) and (theHW < gykrd)) then
    if (yRichtung = "S") then
      if (xkrd2 > xkrd1) then
        theRW=((ykrd1-theHW)/(ykrd1-ykrd2))*(xkrd2-xkrd1)+xkrd1
        ListofrGrPt.Add(theRW@theHW)
      elseif (xkrd1 > xkrd2) then
        theRW=xkrd1-(((ykrd1-theHW)/(ykrd1-ykrd2))*(xkrd1-xkrd2))
        ListofrGrPt.Add(theRW@theHW)
      elseif (xkrd1 = xkrd2) then
        theRW=xkrd1
        ListofrGrPt.Add(theRW@theHW)
      end
    elseif (yRichtung = "N") then
      if (xkrd2 > xkrd1) then
        theRW=xkrd2-(((ykrd2-theHW)/(ykrd2-ykrd1))*(xkrd2-xkrd1))
        ListoflGrPt.Add(theRW@theHW)
      elseif (xkrd1 > xkrd2) then
        theRW=((ykrd2-theHW)/(ykrd2-ykrd1))*(xkrd1-xkrd2)+xkrd2
        ListoflGrPt.Add(theRW@theHW)
      elseif (xkrd1 = xkrd2) then
        theRW=xkrd1
        ListofrGrPt.Add(theRW@theHW)
      end
    end
  end
elseif (Pt = PgHIndAnz) then
  ykrd=ListofPgHy.Get(Pt)
  if (theHW = ykrd) then
    xkrd=ListofPgHx.Get(Pt)
    if (yRichtung = "S") then
      ListofrGrPt.Add(xkrd@ykrd)
    elseif (yRichtung = "N") then
      ListoflGrPt.Add(xkrd@ykrd)
    end
```

```
                end
              end
            end
          end
        end
      end
      'Show percentage complete with enabled stop button
      more=av.SetStatus((Ng/(HWIdx+1))*100)
      if (not more) then
        break
      end
    end
    AnzrGr=ListofrGrPt.Count
    AnzrGrIdx=AnzrGr-1
    AnzlGr=ListoflGrPt.Count
    AnzlGrIdx=AnzlGr-1

    AW1=MsgBox.YesNo("Ist ein Punkt-Thema"+NL+"zur Speicherung der Punkte"
       +NL+"im View schon vorhanden?", "Kontrolle", TRUE)
    if (AW1) then
      ListofThms=theView.GetThemes
      ListofPTThemes = {}

      for each aT in ListofThms
        if (aT.Is(FTheme)) then
          aFTab = aT.GetFTab
          if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
            ListofPTThemes.Add(aT)
          end
        end
      end
      TbTheme =MsgBox.ChoiceAsString(ListofPTThemes,
            "um die Punkte der Grenze zu speichern",
            "Auswahl eines PT-Themas im View:"++theView.AsString)
      TbFTab=TbTheme.GetFTab
      TbShpFld=TbFTab.FindField("Shape")
      TbIDFld=TbFTab.FindField("ID")
      ListofTbFlds = TbFTab.GetFields
      TbRtFld=MsgBox.ListAsString(ListofTbFlds,
           "für Richtungen der Hochwerte der G-K-Koord.",
           "Auswahl eines Feldes im Thema"++TbTheme.AsString)
      TbRWFld=MsgBox.ListAsString(ListofTbFlds, "für RW",
           "Auswahl eines Feldes im Thema"++TbTheme.AsString)
      TbHWFld=MsgBox.ListAsString(ListofTbFlds, "für HW",
           "Auswahl eines Feldes im Thema"++TbTheme.AsString)
      TbKWFld=MsgBox.ListAsString(ListofTbFlds, "für Kennwort",
           "Auswahl eines Feldes im Thema"++TbTheme.AsString)
      AnzDt = 0
      for each arec in TbFTab
         AnzDt = AnzDt +1
      end
      IdxDt = AnzDt - 1
      recNr = IdxDt
    elseif (Not AW1) then
      WDStr=theProject.GetWorkDir.AsString
      fStrD = ThStr.Left(6)
      fnStr=FileName.Make(WDStr).MakeTmp(fStrD,"shp")
      fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
      if (fName=nil) then exit end
      fName.SetExtension("shp")
      TbFTab =FTab.MakeNew(fName, Point)
      TbShpFld =TbFTab.FindField("shape")
```

```
  TbIDFld=Field.Make("ID", #Field_Short, 4, 0)
  TbRtFld=Field.Make("HWRichtung", #FIELD_CHAR, 2, 0)
  TbRWFld=Field.Make("RW", #Field_Float, 10, 2)
  TbHWFld=Field.Make("HW", #Field_Float, 10, 2)
  TbKWFld=Field.Make("Kennwort", #Field_Char, 6, 0)
  ListofTbFld={TbIDFld, TbRtFld, TbRWFld, TbHWFld, TbKWFld}
  TbFTab.AddFields(ListofTbFld)
  recNr=-1
  TbTheme =FTheme.Make(TbFTab)
  theView.AddTheme(TbTheme)
end

TbFTab.SetEditable(False)
TbFTab.SetEditable(true)
For each thePt in 0..AnzrGrIdx
   theGrPt=ListofrGrPt.Get(thePt)
   xkrd=theGrPt.Getx
   ykrd=theGrPt.Gety
   recNr=recNr+1
   TbFTab.AddRecord
   TbFTab.SetValue(TbShpFld, recNr, theGrPt)
   TbFTab.SetValue(TbIDFld, recNr, recNr)
   TbFTab.SetValue(TbRtFld, recNr, "S")
   TbFTab.SetValue(TbRWFld, recNr, xkrd)
   TbFTab.SetValue(TbHWFld, recNr, ykrd)
   TbFTab.SetValue(TbKWFld, recNr, theKW)
end

For each thePt in 0..AnzlGrIdx
   theGrPt=ListoflGrPt.Get(thePt)
   xkrd=theGrPt.Getx
   ykrd=theGrPt.Gety
   recNr=recNr+1
   TbFTab.AddRecord
   TbFTab.SetValue(TbShpFld, recNr, theGrPt)
   TbFTab.SetValue(TbIDFld, recNr, recNr)
   TbFTab.SetValue(TbRtFld, recNr, "N")
   TbFTab.SetValue(TbRWFld, recNr, xkrd)
   TbFTab.SetValue(TbHWFld, recNr, ykrd)
   TbFTab.SetValue(TbKWFld, recNr, theKW)
end

AnzwGr=ListofwGrPt.Count
AnzwGrIdx=AnzwGr-1
if (AnzwGr > 0) then
  For each thePt in 0..AnzwGrIdx
     theGrPt=ListofwGrPt.Get(thePt)
     xkrd=theGrPt.Getx
     ykrd=theGrPt.Gety
     recNr=recNr+1
     TbFTab.AddRecord
     TbFTab.SetValue(TbShpFld, recNr, theGrPt)
     TbFTab.SetValue(TbIDFld, recNr, recNr)
     TbFTab.SetValue(TbRtFld, recNr, "W")
     TbFTab.SetValue(TbRWFld, recNr, xkrd)
     TbFTab.SetValue(TbHWFld, recNr, ykrd)
     TbFTab.SetValue(TbKWFld, recNr, theKW)
  end
end
TbFTab.SetEditable(False)
TbTheme.UpdateLegend
```

```
'ptauflhw.ave
'Stützpunkte einer Polyline für eine Rinne werden als Punkte bestimmt.
'Zwischnen den Stützpunkten werden Punkte in einem HW-Abstand von 50 m berechnet.
'Die bestimmten und berechneten Punkte werden in einem Punkt-Thema gespeichert.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("ptauflhw")
myScript.SetNumberFormat( "d.dd") ' script default

ListofThemes=theView.GetThemes
ListofPLFThm = {}
ListofPtFThm = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aClassNm= aFTab.GetShapeClass
      if (aClassNm.IsSubclassOf(PolyLine)) then
        ListofPLFThm.Add(aT)
      elseif (aClassNm.IsSubclassOf(Point)) then
        ListofPtFThm.Add(aT)
      end
   end
end

PLTheme=MsgBox.ChoiceAsString(ListofPLFThm, "PolyLine-Thema für eine Rinne"
                        ++"im View"++theView.AsString,
                        "Auswahl des Themas (Input)")

PLFTab=PLTheme.GetFTab
ShpFld=PLFTab.FindField("Shape")
IDFld=PLFTab.FindField("Id")

'Erkennung der Koordinaten auf der PolyLine
av.ShowMsg("Feststellung der Koordinaten der PolyLine in"++PLTheme.AsString++"...")

theShape=PLFTab.ReturnValue(ShpFld, 0)
ListofShapes={}
ListofShapes.Add({theShape})

ListofPlHx={}
ListofPlHy={}
maxHW=5618000.00
minHW=5642000.00

'2D-PolyLine wird in Liste der Koordinaten umgewandelt
if (ListofShapes <> 0) then
  for each aShp in ListofShapes
     aListofListofPts=aShp.Get(0).AsList
     ListIdx=0
     for each aList in aListofListofPts
        ListIdx=(ListIdx+1).SetFormat("").SetFormat("d")
        for each myPt in aList
           myx=myPt.Getx
           myy=myPt.Gety
           ListofPlHx.Add(myx)
           ListofPlHy.Add(myy)
```

```
            if (myy > maxHW) then
              maxHW=myy
            end
            if (myy < minHW) then
              minHW=myy
            end
          end
        end
      end
    end
end
AnzPlHx=ListofPlHx.Count
PlHIndAnz=AnzPlHx-1
MsgBox.Info(AnzPlHx.AsString, "Anzahl des Vertices")

av.ShowMsg("Berechnung der Koordinaten auf der PolyLine in
Thema"++PLTheme.AsString++"...")
av.ShowStopButton
ListofPt={}
ListofRt={}
Ng=0

for each Pt in 0..PlHIndAnz
   Ng=Ng+1
   ykrd1=ListofPlHy.Get(Pt)
   xkrd1=ListofPlHx.Get(Pt)
   if (Pt < PlHIndAnz) then
     xkrd2=ListofPlHx.Get(Pt+1)
     ykrd2=ListofPlHy.Get(Pt+1)
     if (ykrd1 > ykrd2) then
       yRichtung="S"
       minHW=ykrd2
       maxHW=ykrd1
       kykrd = ykrd2
       gykrd = ykrd1
     elseif (ykrd2 > ykrd1) then
       yRichtung="N"
       minHW=ykrd1
       maxHW=ykrd2
       kykrd = ykrd1
       gykrd = ykrd2
     elseif (ykrd1 = ykrd2) then
       if (xkrd1 > xkrd2) then
         yRichtung="W"
       elseif (xkrd1 < xkrd2) then
         yRichtung="E"
       end
       minHW=ykrd1
       maxHW=ykrd2
     end
     if (xkrd1 > xkrd2) then
       gxkrd = xkrd1
       kxkrd = xkrd2
     elseif (xkrd2 > xkrd1) then
       gxkrd = xkrd2
       kxkrd = xkrd1
     end
     theHW=ykrd1
     theRW=xkrd1
     ListofPt.Add(theRW@theHW)
     ListofRt.Add(yRichtung)
     'Bestimmung der Punkte zwischen zwei Stützpunkten
     if (((yRichtung ="N") or (yRichtung ="E")) or (yRichtung ="W")) then
```

```
  minHWfl=((minHW/100).Floor)*100
  zdiff=minHW-minHWfl
  if (zdiff = 0) then
    zsz=0
  elseif ((zdiff > 0) and ((zdiff < 50) or (zdiff = 50))) then
    zsz=50
  elseif (zdiff > 50) then
    zsz=100
  end
  theHW=minHWfl+zsz
  While (((theHW > minHW) or (theHW = minHW)) and (theHW < maxHW))
     if (theHW = ykrd1) then
       theRW=xkrd1
       'ListofPt.Add(theRW@theHW)
       'ListofRt.Add(yRichtung)
     elseif ((theHW > kykrd) and (theHW < gykrd)) then
       if (yRichtung = "S") then
         if (xkrd2 > xkrd1) then
           theRW=((ykrd1-theHW)/(ykrd1-ykrd2))*(xkrd2-xkrd1)+xkrd1
           ListofPt.Add(theRW@theHW)
           ListofRt.Add(yRichtung)
         elseif (xkrd1 > xkrd2) then
           theRW=xkrd1-(((ykrd1-theHW)/(ykrd1-ykrd2))*(xkrd1-xkrd2))
           ListofPt.Add(theRW@theHW)
           ListofRt.Add(yRichtung)
         elseif (xkrd1 = xkrd2) then
           theRW=xkrd1
           ListofPt.Add(theRW@theHW)
           ListofRt.Add(yRichtung)
         end
       elseif (yRichtung = "N") then
         if (xkrd2 > xkrd1) then
           theRW=xkrd2-(((ykrd2-theHW)/(ykrd2-ykrd1))*(xkrd2-xkrd1))
           ListofPt.Add(theRW@theHW)
           ListofRt.Add(yRichtung)
         elseif (xkrd1 > xkrd2) then
           theRW=((ykrd2-theHW)/(ykrd2-ykrd1))*(xkrd1-xkrd2)+xkrd2
           ListofPt.Add(theRW@theHW)
           ListofRt.Add(yRichtung)
         elseif (xkrd1 = xkrd2) then
           theRW=xkrd1
           ListofPt.Add(theRW@theHW)
           ListofRt.Add(yRichtung)
         end
       end
     end
     theHW=theHW+50
  end                    'End of While
elseif (yRichtung ="S") then
  maxHWcl=((maxHW/100).Ceiling)*100
  zdiff=maxHWcl-maxHW
  if (zdiff = 0) then
    zsz=0
  elseif ((zdiff > 0) and ((zdiff < 50) or (zdiff = 50))) then
    zsz=50
  elseif (zdiff > 50) then
    zsz=100
  end
  theHW=maxHWcl-zsz
  While (((theHW > minHW)) and ((theHW < maxHW) or (theHW = maxHW)))
     if (theHW = ykrd1) then
       theRW=xkrd1
```

```
            'ListofPt.Add(theRW@theHW)
            'ListofRt.Add(yRichtung)
          elseif ((theHW > kykrd) and (theHW < gykrd)) then
            if (yRichtung = "S") then
              if (xkrd2 > xkrd1) then
                theRW=((ykrd1-theHW)/(ykrd1-ykrd2))*(xkrd2-xkrd1)+xkrd1
                ListofPt.Add(theRW@theHW)
                ListofRt.Add(yRichtung)
              elseif (xkrd1 > xkrd2) then
                theRW=xkrd1-(((ykrd1-theHW)/(ykrd1-ykrd2))*(xkrd1-xkrd2))
                ListofPt.Add(theRW@theHW)
                ListofRt.Add(yRichtung)
              elseif (xkrd1 = xkrd2) then
                theRW=xkrd1
                ListofPt.Add(theRW@theHW)
                ListofRt.Add(yRichtung)
              end
            elseif (yRichtung = "N") then
              if (xkrd2 > xkrd1) then
                theRW=xkrd2-(((ykrd2-theHW)/(ykrd2-ykrd1))*(xkrd2-xkrd1))
                ListofPt.Add(theRW@theHW)
                ListofRt.Add(yRichtung)
              elseif (xkrd1 > xkrd2) then
                theRW=((ykrd2-theHW)/(ykrd2-ykrd1))*(xkrd1-xkrd2)+xkrd2
                ListofPt.Add(theRW@theHW)
                ListofRt.Add(yRichtung)
              elseif (xkrd1 = xkrd2) then
                theRW=xkrd1
                ListofPt.Add(theRW@theHW)
                ListofRt.Add(yRichtung)
              end
            end
          end
          theHW=theHW-50
        end     'End of While
      end       'End of if ((yRichtung ="N") or (yRichtung ="E") or (yRichtung ="W")) then

    elseif (Pt = PlHIndAnz) then
      xkrd0=ListofPlHx.Get(Pt-1)
      ykrd0=ListofPlHy.Get(Pt-1)
      theHW=5641050
      if (theHW = ykrd1) then
        if (ykrd0 > ykrd1) then
          yRichtung="S"
        elseif (ykrd1 > ykrd0) then
          yRichtung="N"
        elseif (ykrd0 = ykrd1) then
          if (xkrd1 < xkrd0) then
            yRichtung="W"
          elseif (xkrd0 < xkrd1) then
            yRichtung="E"
          end
        end
        if (yRichtung = "S") then
          ListofPt.Add(xkrd1@ykrd1)
          ListofRt.Add(yRichtung)
        elseif (yRichtung = "N") then
          ListofPt.Add(xkrd1@ykrd1)
          ListofRt.Add(yRichtung)
        elseif (yRichtung = "E") then
          ListofPt.Add(xkrd1@ykrd1)
          ListofRt.Add(yRichtung)
```

```
        elseif (yRichtung = "W") then
          ListofPt.Add(xkrd1@ykrd1)
          ListofRt.Add(yRichtung)
        end
      end
    end              'end for (if (Pt < PlHIndAnz) then)
    'Show percentage complete with enabled stop button
    more=av.SetStatus((Ng/(AnzPlHx))*100)
    if (not more) then
      break
    end
  end                'end for (for each Pt in 0..PlHIndAnz)

'Auswahl eines Themas zur Speicherung der berechneten Punkte
ListofAW={"ein neues Thema", "ein vorhandenes Thema"}
AW11=MsgBox.ChoiceAsString(ListofAW, "zur Speicherung der Punkte der Rinne"
       ++PLTheme.AsString, "Auswahl eines Punkt-Themas")

if (AW11 = "ein neues Thema") then
  'Ein Feature-Shape-File für Punkte der Rinne wird hergestellt.
  WDStr=av.GetProject.GetWorkDir.AsString
  FnStr0="Pt"+ PLTheme.AsString
  fnStr=FileName.Make(WDStr).MakeTmp(FnStr0,"shp")
  'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(FnStr0,"shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
  if (fName =nil) then exit end
  fName.SetExtension("shp")
  PtFTab =FTab.MakeNew(fName, Point)
  PtShpFld = PtFTab.FindField("shape")
  PtIDFld =Field.Make("ID", #Field_Short, 5, 0)          'ID für ganze Datensätze
  PtHWRFld= Field.Make("HWRichtung", #Field_Char, 4, 0)
  PtRWFld =Field.Make("RW", #Field_Float, 10, 2)
  PtHWFld= Field.Make("HW", #Field_Float, 10, 2)
  PtRWprjFld= Field.Make("RWprj", #Field_Float, 10, 2)
  PtHWprjFld =Field.Make("HWprj", #Field_Float, 10, 2)
  PtHmFld= Field.Make("Hoehem", #Field_Float, 6, 2)
  PtH50Fld= Field.Make("Hoehe50", #Field_Float, 8, 2)

  ListofPtFlds={PtIDFld, PtHWRFld, PtRWFld, PtHWFld,
                PtRWprjFld ,PtHWprjFld, PtHmFld, PtH50Fld}
  PtFTab.AddFields(ListofPtFlds)
  recNr=-1                'Nummer für ganze Datensätze
elseif (AW11 = "ein vorhandenes Thema") then
  PtTheme=MsgBox.ChoiceAsString(ListofPtFThm,
                "Punkt-Thema für Rinne"++PLTheme.AsString,
                "Auswahl eines Themas")
  PtFTab=PtTheme.GetFTab
  PtShpFld=PtFTab.FindField("Shape")
  PtIDFld=PtFTab.FindField("ID")          'ID für ganze Datensätze
  PtHWRFld=PtFTab.FindField("HWRichtung")
  PtRWFld=PtFTab.FindField("RW")
  PtHWFld=PtFTab.FindField("HW")
  PtRWprjFld=PtFTab.FindField("RWprj")
  PtHWprjFld=PtFTab.FindField("HWprj")
  PtHmFld=PtFTab.FindField("Hoehem")
  PtH50Fld=PtFTab.FindField("Hoehe50")

  'Feststellung der Anzahl der Datensätze in dem Thema für Punkte
  AnzPtRec=0
  for each rec in PtFTab
    AnzPtRec = AnzPtRec +1
  end
```

```
  IdxPtRec= AnzPtRec -1
  recNr=IdxPtRec   'Nummer für ganze Datensätze
end

av.ShowMsg("Speicherung der Punkte in einem Punkt-Thema ...")
'Speicherung der Punkte in Tabelle
AnzPt=ListofPt.Count
PtIdx=AnzPt-1

AnzRt=ListofRt.Count
RtIdx=AnzRt-1
Diff=AnzPt-AnzRt

MsgBox.Report("Anzahl der Punkte:"++AnzPt.AsString+NL+
          "Anzahl der Richtungen"++AnzRt.AsString+NL+
          "Differenz der beiden Anzahlen"++Diff.AsString, "Kontrolle der Daten")

PtFTab.SetEditable(False)
PtFTab.SetEditable(true)

For each aPt in 0.. PtIdx
   thePt= ListofPt.Get(aPt)
   theHWR= ListofRt.Get(aPt)
   theRW= thePt.Getx
   theHW= thePt.Gety
   theRWprj=0.00
   theHWprj=0.00
   theHm=0.00
   theH50=0.00
   recNr=recNr+1
   PtFTab.AddRecord
   PtFTab.SetValue(PtShpFld, recNr, thePt)
   PtFTab.SetValue(PtIDFld, recNr, recNr)
   PtFTab.SetValue(PtHWRFld, recNr, theHWR)
   PtFTab.SetValue(PtRWFld, recNr, theRW)
   PtFTab.SetValue(PtHWFld, recNr, theHW)
   PtFTab.SetValue(PtRWprjFld, recNr, theRWprj)
   PtFTab.SetValue(PtHWprjFld, recNr, theHWprj)
   PtFTab.SetValue(PtHmFld, recNr, theHm)
   PtFTab.SetValue(PtH50Fld, recNr, theH50)
end

PtFTab.SetEditable(False)

if (AW11 = "ein neues Thema") then
  PtNewTh=FTheme.Make(PtFTab)
  theView.AddTheme(PtNewTh)
  theTheme= PtNewTh
elseif (AW11 = "ein vorhandenes Thema") then
  theTheme= PtTheme
end
theTheme.UpdateLegend




'ptaufplg.ave
'Punkte auf einer Polyline oder auf einem Polygon werden hergestellt.
'Die Punkte bestehen aus den Stützpunkten der Formen oder den Punkten
' in einem bestimmten Abstand auf den Formen.
```

```
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc   'Ein aktives View

'Auswahl eines Polyline-Themas, um Punkte herzustellen.
theThm = theView.GetActiveThemes.Get(0)
ThStr = theThm.AsString

kt00=MsgBox.YesNo("Ist das aktive Thema"++ThStr
     +NL+"zur Herstellung der Punkte richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
           "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

PLgFTab = theThm.GetFTab
aShapeClassName = PLgFTab.GetShapeClass.GetClassName

ListofFlds = PLgFTab.GetFields
PLgShpFld = ListofFlds.Get(0)

'Anzahl der Felder im Thema
AnzFlds = ListofFlds.Count
IdxFlds = AnzFlds - 1

'Feststellung des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxFlds
   aFld=ListofFlds.Get(i)
   aFldStr=aFld.GetName
   if (aFldStr <> "Shape") then
      FldStr = FldStr + aFldStr+"; "
      aValue = PLgFTab.ReturnValue(aFld, 0)
      DtStr=DtStr+aValue.AsString+"; "
   end
end

MsgBox.Report("Die Namen der Felder"+NL+FldStr+NL+NL+
           "Der erste Datensatz:"+NL+DtStr,
           "Information")
aKWFld = MsgBox.ListAsString(ListofFlds,
  "für die Kennwörter der Datensätze",
  "Auswahl eines Feldes des Themas"++ThStr)
aKWFldStr = aKWFld.AsString

theLegend=theThm.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(theThm, aKWFldStr)
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")

aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
AnzSymbIdx=AnzSymb-2
ListoforgColor = {}
```

```
for each symb in 0..AnzSymbIdx
    aorgColor = aListofSymbol.Get(symb).GetColor
    ListoforgColor.Add(aorgColor)
end

ListofKlKW = {}
for each i in 0..IdxKlasse
  theKlasseLb = ListofKlasse.Get(i).GetLabel
  ListofKlKW.Add(theKlasseLb)
end

theKl = MsgBox.ListAsString(ListofKlKW,
  "zur Auswahl eines Datensatzes ("++aShapeClassName++")",
  "Auswahl einer Klasse der Datensätze im"++ThStr)
if (theKl = nil) then
  MsgBox.Error("Eine Klasse der Datensätze oder"+NL+
               "ein Datensatz mit der Klasse fehlt!"+NL+
               "Das Programm wird abgebrochen!", "")
  exit
end

'Anzahl der Datensätze im Thema
AnzPLg=0
for each rec in PLgFTab
   AnzPLg = AnzPLg + 1
end
IdxPLg=AnzPLg - 1

'mit der Klasse ausgewählte Datensätze
ListofDs = {}
ListofIdxgDs = {}
FldStr=""
ListofFldStr = {}
for each j in 0..IdxPLg
  DtStr = ""
  aKW = PLgFTab.ReturnValue(aKWFld, j)
  if (aKW = theKl) then
    for each i in 0..IdxFlds
       aFld=ListofFlds.Get(i)
       aFldStr=aFld.GetName
       if (aFldStr <> "Shape") then
          FldStr = FldStr + aFldStr+"; "
          aValue = PLgFTab.ReturnValue(aFld, j)
          DtStr=DtStr+aValue.AsString+"; "
       end
    end
    ListofFldStr.Add(FldStr)
    ListofDs.Add(DtStr)
    ListofIdxgDs.Add(j)
  end
end
MsgBox.ListAsString(ListofFldStr, "Feldnamen", "Kont")
'Auswahl eines Datensatzes, um Punkte herzustellen
aFldStr = ListofFldStr.Get(0)
aDs = MsgBox.ListAsString(ListofDs,
    "zur Herstellung der Punkte"+NL+
    "(Feldname: "++aFldStr+")",
    "Auswahl eines Datensatzes im"++ThStr)
if (aDs = nil) then
  MsgBox.Error("Ein Datensatz mit dem Kennwort fehlt!"+NL+
               "Das Programm wird abgebrochen!", "")
  exit
```

```
end
aIdxDs = ListofDs.FindByValue(aDs) 'ein Index in der Liste
aIdxgDt = ListofIdxgDs.Get(aIdxDs) 'ein Index in den gesamten Datensätzen
thePLgShp = PLgFTab.ReturnValue(PLgShpFld, aIdxgDt)

av.ShowMsg("Feststellung der Koordinaten der 2D-Form des Themas"
     ++ThStr++"...")

theProfilList1={}
theProfilList1.Add({thePLgShp})
ListofVt = {}

'2D-PolyLine wird in eine Liste der Koordinaten umgewandelt.
if (theProfilList1 <> 0) then
  for each q in theProfilList1
    theLines=q.Get(0).AsList
    for each m in theLines
      for each ptx in m
        myx=ptx.Getx
        myy=ptx.Gety
        ListofVt.Add(myx@myy)
      end
    end
  end
end

ListofArtik = {"der","dem"}
if (aShapeClassName = "Polyline") then
  aArtik = ListofArtik.Get(0)
elseif (aShapeClassName = "Polygon") then
  aArtik = ListofArtik.Get(1)
end

ListofArt = {"Punkte nur von Stützpunkten",
           "Punkte in einem bestimmten Abstand"}
aArt = MsgBox.ListAsString(ListofArt,
    "die auf"++aArtik++aShapeClassName++"entstehen",
    "Auswahl einer Art der Punkte")

if (aArt = "Punkte nur von Stützpunkten") then
  ListofPT = ListofVt.DeepClone
elseif (aArt = "Punkte in einem bestimmten Abstand") then
  'Berechnung der Koordinaten der Punkte,
 'welche in einem bestimmten Abstand liegen.
 av.ShowMsg("Berechnung der Koordinaten der Punkte auf"++aArtik
         ++aShapeClassName+":"++ThStr++"...")
 av.ShowStopButton

 'Eingabe eines Abstandes zwischen den Punkten
 EntfStr=MsgBox.Input("auf"++aArtik++aShapeClassName++"(m)",
     "Eingabe des Abstandes der Punkte", "50")
 Ef=EntfStr.AsNumber

 AnzVt = ListofVt.Count
 if (aShapeClassName = "Polyline") then
   IdxVt = AnzVt - 1
 elseif (aShapeClassName = "Polygon") then
   IdxVt = AnzVt - 2
 end

 Ng = 0
 ListofPT = {}
```

```
for each j in 0..IdxVt
  Ng = Ng + 1
  AnfPt = ListofVt.Get(j)
  Anfx = AnfPt.Getx
  Anfy = AnfPt.Gety
  X0=Anfx
  Y0=Anfy
  ListofPT.Add(X0@Y0)

  if (j < IdxVt) then
    EndPt = ListofVt.Get(j+1)
    Endx = EndPt.Getx
    Endy = EndPt.Gety
    'Berechnung der Gleichung der geraden Linie zwischen Stützpunkten
    DX=Endx-Anfx
    DY=Endy-Anfy
    if (DX <> 0) then
      a=DY/DX
      b=Anfy-(a*Anfx)
    end
    'Länge der Linie
    GL=(((DX*DX)+(DY*DY)).Sqrt).Abs

    if (GL > Ef) then

    AnzFl=(GL/Ef)
    AnzInt=(GL/Ef).Floor
    aRest=AnzFl-AnzInt
    AnzGPT=AnzInt

    if (aRest = 0) then
       IdxGPT=AnzGPT-2
    elseif (aRest <> 0) then
       IdxGPT=AnzGPT-1
    end

    for each i in 0..IdxGPT
      if (DX = 0) then
        X1P=X0
        if (Endy > Anfy) then
          Y1P=Y0+Ef
        elseif (Endy < Anfy) then
          Y1P=Y0-Ef
        end
        aPt = Point.Make(X0,Y1P)
        aTest = thePLgShp.Intersects(aPt)
        if (aTest) then
          ListofPT.Add(aPt)
        end
        Y0=Y1P
      elseif (DX <> 0) then
        if (DY = 0) then
          if (Endx > Anfx) then
            X1P = X0+Ef
          elseif (Endx < Anfx) then
            X1P = X0-Ef
          end
          Y1P = Y0
          aPt = Point.Make(X1P,Y0)
          aTest = thePLgShp.Intersects(aPt)
          if (aTest) then
```

```
            ListofPT.Add(aPt)
          end
          X0=X1P
        elseif (DY <> 0) then
          P=((2*a*b)-(2*X0)-(2*a*Y0))/(1+(a*a))
          Q=((X0*X0)+(Y0*Y0)-(2*Y0*b)+(b*b)-(Ef*Ef))/(1+(a*a))
          WZ=((((P*P)/4)-Q).Sqrt).Abs

          if (Endx > Anfx) then
            X1P = (-1*(P/2))+WZ
          elseif (Endx < Anfx) then
            X1P = (-1*(P/2))-WZ
          end
          Y1P=a*X1P+b
          aPt = Point.Make(X1P,Y1P)
          aTest = thePLgShp.Intersects(aPt)
          if (aTest) then
            ListofPT.Add(aPt)
          end
          X0=X1P
          Y0=Y1P
        end   'Ende von (if (DY = 0) then)
      end  'Ende von (if (DX = 0) then)
    end  'Ende von (for each i in 0..IdxGPT)
    end  'Ende von (if (GL > Ef) then)
  end  'Ende von (if (j < IdxVt) then)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzVt*100)
  if (not more) then
    break
  end
 end  'Ende von (j-Schleife)
end

'Berechnung der Entfernung der Punkte vom Anfang
AnzNPT=ListofPT.Count
IdxNPT=AnzNPT-1

ListofEntf = {}
for each i in 0..IdxNPT
  aPt = ListofPT.Get(i)
  myx=aPt.Getx
  myy=aPt.Gety
  if (i = 0) then
    Entfsum = 0
  elseif (i > 0) then
    entf = (((X0 - myx) ^ 2) + ((Y0 - myy) ^ 2)).Sqrt.Abs
    Entfsum = Entfsum + entf
  end
  ListofEntf.Add(Entfsum)
  X0 = myx
  Y0 = myy
end

'Auswahl oder Herausstellung eines Themas, um die Punkte zu speichern.
Frg1=MsgBox.YesNo("Gibt es schon ein Punkt-Thema im aktiven View?",
  "Ein Thema zur Speicherung der berechneten Daten", true)
if (Frg1) then
  ListofThms = theView.GetThemes
  ListofPtThms = {}

  for each aT in ListofThms
```

```
 if (aT.Is(FTheme)) then
    aFTab = aT.GetFTab
    if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
      ListofPtThms.Add(aT)
    end
 end
end

thePtThm=MsgBox.ChoiceAsString(ListofPtThms,
      "zur Speicherung der berechneten Punkte",
      "Auswahl eines Themas im View"++theView.AsString)
PtFTab=thePtThm.GetFTab
PtShpFld=PtFTab.FindField("Shape")
PtIDFld=PtFTab.FindField("ID")
ListofPtFlds=PtFTab.GetFields

'Anzahl der Felder im Thema
AnzPtFlds = ListofPtFlds.Count
IdxPtFlds = AnzPtFlds - 1

'Feststellung des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxPtFlds
  aFld = ListofPtFlds.Get(i)
  aFldStr = aFld.GetName
  if (aFldStr <> "Shape") then
     FldStr = FldStr + aFldStr+"; "
     aValue = PtFTab.ReturnValue(aFld, 0)
     DtStr=DtStr+aValue.AsString+"; "
  end
end

MsgBox.Report("Die Namen der Felder"+NL+FldStr+NL+NL+
         "Der erste Datensatz:"+NL+DtStr,
         "Information")
PtxFld = MsgBox.ListAsString(ListofPtFlds,
  "für x-Koord.",
  "Auswahl eines Feldes des Themas"++thePtThm.AsString)
PtyFld = MsgBox.ListAsString(ListofPtFlds,
  "für y-Koord.",
  "Auswahl eines Feldes des Themas"++thePtThm.AsString)
PtEntf = MsgBox.ListAsString(ListofPtFlds,
  "für Entfernung vom Anfang",
  "Auswahl eines Feldes des Themas"++thePtThm.AsString)
aKWPtFld = MsgBox.ListAsString(ListofPtFlds,
  "um die Kennwörter der Datensätze zu speichern",
  "Auswahl eines Feldes des Themas"++thePtThm.AsString)


'Anzahl der Datensätze im Thema
AnzPt=0
for each rec in PtFTab
   AnzPt=AnzPt+1
end
IdxPt=AnzPt-1
recNr = IdxPt

elseif (Not Frg1) then
  'Ein Feature-Shape-File zur Speicherung der berechneten Punkte
  WDStr=theProject.GetWorkDir.AsString
  fn00 = "Pt"+ThStr.Left(4)
```

```
  fnStr= FileName.Make(WDStr).MakeTmp(fn00,"shp")
  'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp("Pfpt00kt","shp")
  fName=FileDialog.Put(fnStr, "*.shp",
      "Output shape File (Punkte eines Profils)")
  if (fName=nil) then exit end
  fName.SetExtension("shp")
  PtFTab=FTab.MakeNew(fName, Point)
  PtShpFld=PtFTab.FindField("shape")
  PtIDFld=Field.Make("ID", #Field_Short, 6, 0)
  PtxFld=Field.Make("x-Krd", #Field_Float, 10, 2)
  PtyFld=Field.Make("y-Krd", #Field_Float, 10, 2)
  PtEntf=Field.Make("Entfernung", #Field_Float, 8, 2)
  aKWPtFld=Field.Make("Kennwort", #Field_Char, 6, 0)

  ListofSPtFlds={PtIDFld, PtxFld, PtyFld, PtEntf, aKWPtFld}
  PtFTab.AddFields(ListofSPtFlds)
  thePtThm=FTheme.Make(PtFTab)
  theView.AddTheme(thePtThm)
  recNr = -1
end

av.ShowMsg("Die Punkte werden gespeichert...")

PtFTab.SetEditable(false)
PtFTab.SetEditable(true)
for each aPt in 0..IdxNPT
   recNr = recNr + 1
   thePt = ListofPt.Get(aPt)
   xKrd = thePt.Getx
   yKrd = thePt.Gety
   Entf = ListofEntf.Get(aPt)
   PtFTab.AddRecord
   PtFTab.SetValue(PtShpFld, recNr, thePt)
   PtFTab.SetValue(PtIDFld, recNr, aPt)
   PtFTab.SetValue(PtxFld, recNr, xKrd)
   PtFTab.SetValue(PtyFld, recNr, yKrd)
   PtFTab.SetValue(PtEntf, recNr, Entf)
   PtFTab.SetValue(aKWPtFld, recNr, theKl)
end
PtFTab.SetEditable(false)

for each symb in 0..AnzSymbIdx
  aListofSymbol.Get(symb).SetColor(ListoforgColor.Get(symb))
end
theThm.UpdateLegend
thePtThm.UpdateLegend




'ptauftpl.ave
'Neue Punkte werden auf Teilprofilschnittlinien hergestellt, die aus
'Teilabschnitten einer ganzen, langen Profilschnittlinie bestehen,
'um sie später mit dem ganzen Profilschnitt zusammenzuzeichnen.
'Die Entfernung der neuen Punkte vom Anfang des ganzen Profilschnittes
'werden von einem Abschnitt der ganzen Profilschnittlinie übernommen.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
```

```
theView=av.GetActiveDoc 'Aktives Karten-View

'Eingabe der Datei, die geologisch unterteilten 2D-Profilschnittlinien
'enthält, zur Bestimmung der Punkte

ListofThemes =theView.GetThemes
ListofPLThms = {}
ListofPtThms = {}
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aCNm = aFTab.GetShapeClass.GetClassName
      if (aCNm = "Point") then
         ListofPtThms.Add(aT)
      elseif (aCNm = "Polyline") then
         ListofPLThms.Add(aT)
      end
   end
end

PLTheme=MsgBox.ChoiceAsString(ListofPLThms,
      "das die geologisch unterteilten"
      +NL+"2D-Teil-Profilschnittlinien enthält",
      "Eingabe eines Themas im View"++theView.AsString)

PLFTab=PLTheme.GetFTab
ListofPLFTabFlds=PLFTab.GetFields
PLShpFld=PLFTab.FindField("Shape")
PLIDFld=MsgBox.ChoiceAsString(ListofPLFTabFlds, "für ID",
       "Auswahl eines Feldes in Thema:"++PLTheme.AsString)
PLGeoFld=MsgBox.ChoiceAsString(ListofPLFTabFlds, "für Geologie",
       "Auswahl eines Feldes in Thema:"++PLTheme.AsString)
Listof2DFld={PLShpFld, PLIDFld, PLGeoFld}

'Feststellung der Anzahl der 2D-PolyLine in dem Thema PLTheme
Anz2DPL=0
for each rec in PLFTab
   Anz2DPL=Anz2DPL+1
end
Idx2DPL=Anz2DPL-1
MsgBox.Info(Anz2DPL.AsString, "Anzahl der 2D-PolyLine in dem Thema"
          ++PLTheme.AsString)

'Eingabe der Datei, die Punkte mit Hoehenwerten auf einer ganzen
'Profilschnittlinie enthält, zur Bestimmung der Punkte
PTTheme=MsgBox.ChoiceAsString(ListofPtThms,
       "das alle Punkte auf einer"+NL+
       "ganzen Profilschnittlinie enthält", "Eingabe eines Themas")
PTFTab=PTTheme.GetFTab
ListofFlds=PTFTab.GetFields

PTShpFld=MsgBox.ChoiceAsString(ListofFlds, "Eingabe des Feldes für Shape",
       "Auswahl eines Feldes in Thema:"++PTTheme.AsString)
PTEfFld=MsgBox.ChoiceAsString(ListofFlds,
       "Eingabe des Feldes für Entfernungen vom Anfang",
       "Auswahl eines Feldes in Thema:"++PTTheme.AsString)
ListofPtFld={PTShpFld, PTEfFld}

'Feststellung der Anzahl der Punkte in dem Thema PTTheme
AnzPT=0
for each rec in PTFTab
   AnzPT=AnzPT+1
```

```
end
IdxPT=AnzPT-1
MsgBox.Info(AnzPT.AsString, "Anzahl der Punkte in dem Thema"
        ++PTTheme.AsString)

av.ShowMsg("Bestimmung der Punkte auf den geologisch"
         ++"unterteilten Profilen ...")
av.ShowStopButton

ListofListofNPT={}
ListofListofNEntf={}
ListofNGeol={}

for each gzd in 0..Idx2DPL
  Ng=gzd+1
  the2DPolyL=PLFTab.ReturnValue(PLShpFld, gzd)
  aGeolog=PLFTab.ReturnValue(PLGeoFld, gzd)
  ListofNGeol.Add(aGeolog)
  ListofLPT={}

  'Die Punkte für die PolyLine werden bestimmt.
  theProfilList={}
  theProfilList.Add({the2DPolyL})
  if (theProfilList <> 0) then
    for each theShpL in theProfilList
      ListofListofPt=theShpL.Get(0).AsList
      for each ListofPt in ListofListofPt
          for each aPt in ListofPt
              ListofLPT.Add(aPt)
          end
      end
    end
  end

  AnzLPT=ListofLPT.Count
  IdxLPT=AnzLPT-1
   AnfPT=ListofLPT.Get(0)
  EndPT=ListofLPT.Get(IdxLPT)
  AnfX=AnfPT.Getx
  AnfY=AnfPT.Gety
  EndX=EndPT.Getx
  EndY=EndPT.Gety
  minAbstA=100000
  minAbstE=100000

  'Suche nach dem Abschnitt im Punkt-Thema der ganzen
  'Profilschnittlinie mit dem kleinsten Abstand,
  'um den Abschnitt für Teilprofilschnitt zu finden
  for each i in 0..IdxPT
     thePT=PTFTab.ReturnValue(PTShpFld, i)
     theX=thePT.Getx
     theY=thePT.Gety
     DifXA2=((theX-AnfX)*(theX-AnfX))
     DifYA2=((theY-AnfY)*(theY-AnfY))
     AbstA=(DifXA2+DifYA2).Sqrt.Abs
     if (AbstA < minAbstA) then
       minAbstA=AbstA
       IdxPTA=i
     end
     DifXE2=((theX-EndX)*(theX-EndX))
     DifYE2=((theY-EndY)*(theY-EndY))
     AbstE=(DifXE2+DifYE2).Sqrt.Abs
```

```
    if (AbstE < minAbstE) then
      minAbstE=AbstE
      IdxPTE=i
    end
  end

  'Berechnung der Entfernung der Punkt vom Anfang
  'für den Anfang einer Teilprofilschnittlinie
  if (minAbstA <> 0) then
   VIdxPTA=IdxPTA-1
   NIdxPTA=IdxPTA+1

   thePTVA=PTFTab.ReturnValue(PTShpFld, VIdxPTA)
   thePTNA=PTFTab.ReturnValue(PTShpFld, NIdxPTA)
   thePTIA=PTFTab.ReturnValue(PTShpFld, IdxPTA)

   thePTVAX=thePTVA.Getx
   thePTVAY=thePTVA.Gety
   thePTNAX=thePTNA.Getx
   thePTNAY=thePTNA.Gety
   thePTIAX=thePTIA.Getx
   thePTIAY=thePTIA.Gety

   DifXVA2=((thePTVAX-AnfX)*(thePTVAX-AnfX))
   DifYVA2=((thePTVAY-AnfY)*(thePTVAY-AnfY))
   AbstVA=(DifXVA2+DifYVA2).Sqrt.Abs

   DifXNA2=((thePTNAX-AnfX)*(thePTNAX-AnfX))
   DifYNA2=((thePTNAY-AnfY)*(thePTNAY-AnfY))
   AbstNA=(DifXNA2+DifYNA2).Sqrt.Abs

   DifXVIA=((thePTVAX-thePTIAX)*(thePTVAX-thePTIAX))
   DifYVIA=((thePTVAY-thePTIAY)*(thePTVAY-thePTIAY))
   AbstVIA=(DifXVIA+DifYVIA).Sqrt.Abs

   DifXNIA=((thePTNAX-thePTIAX)*(thePTNAX-thePTIAX))
   DifYNIA=((thePTNAY-thePTIAY)*(thePTNAY-thePTIAY))
   AbstNIA=(DifXNIA+DifYNIA).Sqrt.Abs

   if ((AbstVA < AbstVIA) and (AbstNA > AbstNIA)) then
     VIdxPTA=IdxPTA-1
     NIdxPTA=IdxPTA
     P1IdxPTA=IdxPTA+1
     SWA="V"
   elseif ((AbstVA > AbstVIA) and (AbstNA < AbstNIA)) then
     VIdxPTA=IdxPTA
     NIdxPTA=IdxPTA+1
     P1IdxPTA=IdxPTA+1
     SWA="N"
   end

   thePTVA=PTFTab.ReturnValue(PTShpFld, VIdxPTA)
   thePTNA=PTFTab.ReturnValue(PTShpFld, NIdxPTA)

   thePTVAX=thePTVA.Getx
   thePTVAY=thePTVA.Gety
   thePTNAX=thePTNA.Getx
   thePTNAY=thePTNA.Gety

   DifXVA2=((thePTVAX-AnfX)*(thePTVAX-AnfX))
   DifYVA2=((thePTVAY-AnfY)*(thePTVAY-AnfY))
   AbstVA=(DifXVA2+DifYVA2).Sqrt.Abs
```

```
  EntfVA=PTFTab.ReturnValue(PTEfFld, VIdxPTA)
  EntfPTA=EntfVA+AbstVA

elseif (minAbstA = 0) then
  EntfPTA=PTFTab.ReturnValue(PTEfFld, IdxPTA)
  P1IdxPTA=IdxPTA+1
  SWA="N"
end

'Berechnung der Entfernung der Punkte vom Anfang
'für das Ende einer Teilprofilschnittlinie

if (minAbstE <> 0) then
  VIdxPTE=IdxPTE-1
  NIdxPTE=IdxPTE+1

  thePTE=PTFTab.ReturnValue(PTShpFld, IdxPTE)
  thePTVE=PTFTab.ReturnValue(PTShpFld, VIdxPTE)
  thePTNE=PTFTab.ReturnValue(PTShpFld, NIdxPTE)

  thePTEX=thePTE.Getx
  thePTEY=thePTE.Gety
  thePTVEX=thePTVE.Getx
  thePTVEY=thePTVE.Gety
  thePTNEX=thePTNE.Getx
  thePTNEY=thePTNE.Gety

  DifXVE2=((thePTVEX-EndX)*(thePTVEX-EndX))
  DifYVE2=((thePTVEY-EndY)*(thePTVEY-EndY))
  AbstVE=(DifXVE2+DifYVE2).Sqrt.Abs

  DifXNE2=((thePTNEX-EndX)*(thePTNEX-EndX))
  DifYNE2=((thePTNEY-EndY)*(thePTNEY-EndY))
  AbstNE=(DifXNE2+DifYNE2).Sqrt.Abs

  DifXVIE=((thePTVEX-thePTEX)*(thePTVEX-thePTEX))
  DifYVIE=((thePTVEY-thePTEY)*(thePTVEY-thePTEY))
  AbstVIE=(DifXVIE+DifYVIE).Sqrt.Abs

  DifXNIE=((thePTNEX-thePTEX)*(thePTNEX-thePTEX))
  DifYNIE=((thePTNEY-thePTEY)*(thePTNEY-thePTEY))
  AbstNIE=(DifXNIE+DifYNIE).Sqrt.Abs

  if ((AbstVE < AbstVIE) and (AbstNE > AbstNIE)) then
    VIdxPTE=IdxPTE-1
    NIdxPTE=IdxPTE
    M1IdxPTE=IdxPTE-1
    SWE="V"
  elseif ((AbstVE > AbstVIE) and (AbstNE < AbstNIE)) then
    VIdxPTE=IdxPTE
    NIdxPTE=IdxPTE+1
    M1IdxPTE=IdxPTE-1
    SWE="N"
  end

  thePTVE=PTFTab.ReturnValue(PTShpFld, VIdxPTE)
  thePTNE=PTFTab.ReturnValue(PTShpFld, NIdxPTE)

  thePTVEX=thePTVE.Getx
  thePTVEY=thePTVE.Gety
  thePTNEX=thePTNE.Getx
  thePTNEY=thePTNE.Gety
```

```
    DifXVE2=((thePTVEX-EndX)*(thePTVEX-EndX))
    DifYVE2=((thePTVEY-EndY)*(thePTVEY-EndY))
    AbstVE=(DifXVE2+DifYVE2).Sqrt.Abs
    EntfVE=PTFTab.ReturnValue(PTEfFld, VIdxPTE)
    EntfPTE=EntfVE+AbstVE

  elseif (minAbstE = 0) then
    EntfPTE=PTFTab.ReturnValue(PTEfFld, IdxPTE)
    M1IdxPTE=IdxPTE-1
    SWE="V"
  end

  ListofNPT={}
  ListofNEntf={}
  ListofNPT.Add(AnfPT)
  ListofNEntf.Add(EntfPTA)

  if (SWA = "V") then
     theNNPT=PTFTab.ReturnValue(PTShpFld, NIdxPTA)
     ListofNPT.Add(theNNPT)

     theNNEntf=PTFTab.ReturnValue(PTEfFld, NIdxPTA)
     ListofNEntf.Add(theNNEntf)
  end

  for each i in 0..IdxPT
      if (((i = P1IdxPTA) or (i > P1IdxPTA)) and
          ((i = M1IdxPTE) or (i < M1IdxPTE))) then
         theNPT=PTFTab.ReturnValue(PTShpFld, i)
         ListofNPT.Add(theNPT)

         theNNEntf=PTFTab.ReturnValue(PTEfFld, i)
         ListofNEntf.Add(theNNEntf)
      end
  end

  if (SWE = "N") then
     theNVPT=PTFTab.ReturnValue(PTShpFld, IdxPTE)
     ListofNPT.Add(theNVPT)

     theNNEntf=PTFTab.ReturnValue(PTEfFld, IdxPTE)
     ListofNEntf.Add(theNNEntf)
  end

  ListofNPT.Add(EndPT)
  ListofNEntf.Add(EntfPTE)

  ListofListofNPT.Add(ListofNPT)
  ListofListofNEntf.Add(ListofNEntf)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/Anz2DPL*100)
  if (not more) then
     break
  end
end

'Ein Feature-Shape-File für einen Teil-Profilschnitt (Point) wird hergestellt
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp("Prflpths","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output 2D shape File (Point)")
if (fName=nil) then exit end
```

```
fName.SetExtension("shp")
NPTFTab=FTab.MakeNew(fName, Point)
NPTShpFld=NPTFTab.FindField("shape")
NPTIDFld=Field.Make("ID", #Field_Short, 6, 0)
NPTPrfIDFld=Field.Make("PrfID", #Field_Short, 5, 0)
NPTEntfFld=Field.Make("Entfernung", #Field_Float, 9, 2)
NPTGeoFld=Field.Make("Geologie", #Field_Char, 10, 0)
ListofNPTFlds={NPTIDFld, NPTPrfIDFld, NPTEntfFld, NPTGeoFld}
NPTFTab.AddFields(ListofNPTFlds)
ListofNPTFld2={NPTShpFld, NPTIDFld, NPTPrfIDFld, NPTEntfFld,
            NPTGeoFld}
av.ShowMsg("Speicherung der Punkte auf den geologisch"
        ++"unterteilten Profilen ...")
av.ShowStopButton

NPTFTab.SetEditable(false)
NPTFTab.SetEditable(true)
recNr=-1
for each aPL in 0..Idx2DPL
  Ng=aPL+1
  theNPrfID=aPL
  ListofNPT=ListofListofNPT.Get(aPL)
  ListofNEntf=ListofListofNEntf.Get(aPL)
  aNGeol=ListofNGeol.Get(aPL)
  AnzNPT=ListofNPT.Count
  IdxNPT=AnzNPT-1
  for each aPt in 0..IdxNPT
     theNPT=ListofNPT.Get(aPt)
     theNEnf=ListofNEntf.Get(aPt)
     theNGeol=aNGeol
     recNr=recNr+1
     theNID=recNr
     NPTFTab.AddRecord
     NPTFTab.SetValue(NPTShpFld, recNr, theNPT)
     NPTFTab.SetValue(NPTIDFld, recNr, theNID)
     NPTFTab.SetValue(NPTPrfIDFld, recNr, theNPrfID)
     NPTFTab.SetValue(NPTEntfFld, recNr, theNEnf)
     NPTFTab.SetValue(NPTGeoFld, recNr, aNGeol)
  end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/Anz2DPL*100)
  if (not more) then
    break
  end
end
NPTFTab.SetEditable(false)
NthmNew=FTheme.Make(NPTFTab)
theView.AddTheme(NthmNew)



'ptausm1t.ave
'Ein Punkt, der durch eine Tastatur-Eingabe, ein Fadenkreuz, einen Punkt
'in einem Punkt-Thema, einem Stützpunkt einer Polyline oder eines Polygons
'oder einem Punkt auf einer Polyline oder einem Polygon bestimmt wird,
'wird in einem Punkt-Thema gespeichert.
'Dieses Script wird als ein Werkzeug (Tool) im aktiven View benutzt.

theProject=av.GetProject
myScript=theProject.FindScript("ptausm1t")
```

```
myScript.SetNumberFormat( "d.dd") ' script default

AkView = av.GetActiveDoc

'Einmaliges Klicken der Maus auf das Bildschirm

aMausPt = AkView.GetDisplay.ReturnUserPoint
aMPtx = aMausPt.Getx
aMPty = aMausPt.Gety

aCircle = Circle.Make(aMausPt, 50)
aMCPg = aCircle.AsPolygon

'Auswahl eines Punktes
ListofThemes = AkView.GetThemes
ListofPtThms = {}
ListofPLThms = {}
ListofPgThms = {}

for each aT in ListofThemes
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aSCl = aFTab.GetShapeClass.GetClassName
      if (aSCl = "Point") then
        ListofPtThms.Add(aT)
      elseif (aSCl = "PolyLine") then
        ListofPLThms.Add(aT)
      elseif (aSCl = "Polygon") then
        ListofPgThms.Add(aT)
      end
   end
end

ListofPtForm = {"ein Punkt am Maus-Klicken",
              "ein Punkt einer Tastatur-Eingabe",
              "ein Punkt an einem Fadenkreuz",
              "ein Punkt in einem Punkt-Thema",
              "ein Punkt auf einer Polyline",
              "ein Punkt auf einem Polygon"}

ErstPt = MsgBox.ListAsString(ListofPtForm,
      "um die Koordinaten zu übernehmen",
      "Auswahl eines Punktes am Maus-Klicken")

if (ErstPt = "ein Punkt am Maus-Klicken") then
  aNPt = Point.Make(aMPtx, aMPty)

elseif (ErstPt = "ein Punkt einer Tastatur-Eingabe") then
  aEPtx = (aMausPt.Getx.SetFormat("").SetFormat("d.dd")).AsString
  aEPty = (aMausPt.Gety.SetFormat("").SetFormat("d.dd")).AsString
  aHxStr = MsgBox.Input("Eingabe der x-Koord. des Punktes",
        "Tastaur-Eingabe", aEPtx)
  aHyStr = MsgBox.Input("Eingabe der y-Koord. des Punktes"
         +NL+"(z.B. HW oder Höhe (Faktor * [m]))",
         "Tastaur-Eingabe", aEPty)
  aHx = aHxStr.AsNumber
  aHy = aHyStr.AsNumber
  aNPt = Point.Make(aHx, aHy)

elseif (ErstPt = "ein Punkt an einem Fadenkreuz") then

  KrThm = MsgBox.ListAsString(ListofPLThms,
```

```
      "um Koordinaten zu übernehmen",
      "Auswahl eines Fadenkreuz-Themas")
KrFTab = KrThm.GetFTab
KrShpFld=KrFTab.FindField("Shape")

AnzRec = 0
for each rec in KrFTab
   AnzRec = AnzRec+1
end
AnzRecIdx = AnzRec-1
'MsgBox.Info(AnzRec.AsString,
'   "Anzahl der Datensätze im Thema (Fadenkreuz)"++KrThm.AsString)
ListofKrPtx = {}
ListofKrPty = {}

for each rec in 0..AnzRecIdx
   aPL=KrFTab.ReturnValue(KrShpFld, rec)
   ListofKrPL = {}
   ListofKrPL.Add({aPL})
   for each q in ListofKrPL
      theLs = q.Get(0).AsList
      for each L in theLs
         for each apt in L
            aptx = apt.Getx
            apty = apt.Gety
            ListofKrPtx.Add(aptx)
            ListofKrPty.Add(apty)
         end
      end
   end
end

AnzKrPtx = ListofKrPtx.Count
IdxKrPtx = AnzKrPtx - 1

for each j in 0..IdxKrPtx
   aKrx1 = ListofKrPtx.Get(j)
   aKry1 = ListofKrPty.Get(j)
   xZ = 0
   yZ = 0
   for each i in 0..IdxKrPtx
      aKrx2 = ListofKrPtx.Get(i)
      aKry2 = ListofKrPty.Get(i)
      if (aKrx1 = aKrx2) then
        xZ = xZ + 1
      end
      if (aKry1 = aKry2) then
        yZ = yZ + 1
      end
   end
   if (xZ = 2) then
     theX = aKrx1
   end
   if (yZ = 2) then
     theY = aKry1
   end
end
aNPt = Point.Make(theX, theY)

'MsgBox.Report("Der Punkt am Fadenkreuz"++"("+KrThm.AsString+"):"
'            ++aNPt.AsString,
'            "Die Koordinaten der Punkte an dem Fadenkreuz")
```

```
elseif (ErstPt = "ein Punkt in einem Punkt-Thema") then

  'Auswahl eines Punkt-Themas (evtl. Ereignis-Themas) im aktiven View

  PtTheme = MsgBox.ChoiceAsString(ListofPtThms,
       "um Koordinaten der Punkte zu übernehmen",
       "Auswahl eines Punkt-Themas im View"++AkView.AsString)
  if (PtTheme = nil) then
    MsgBox.Error("Der Name der Punkt-Datei fehlt!"+NL+
           "Das Programm wird abgebrochen!", "")
    exit
  end

  PtFTab = PtTheme.GetFTab
  ListofPtFld = PtFTab.GetFields
  PtShpFld=ListofPtFld.Get(0) 'erstes Shape-Feld beim Ereignisthema

  AnzPt=0
  for each Pt in PtFTab
     AnzPt=AnzPt+1
  end
  AnzPtIdx=AnzPt-1
  AnzPtStr = AnzPt.SetFormat("d").AsString
  'MsgBox.Info(AnzPtStr, "Die Anzahl der Punkte im Thema"
  '          ++PtTheme.AsString)

  'Die Bezeichnung der Punkte wird in der Tabelle gesucht.

  AnzPtFld = ListofPtFld.Count
  IdxPtFld = AnzPtFld - 1

  'Feststellung des ersten Datensatzes
  FldStr=""
  DtStr=""
  for each i in 0..IdxPtFld
     aFld=ListofPtFld.Get(i)
     aFldStr=aFld.GetName
     if (aFldStr <> "Shape") then
        FldStr = FldStr + aFldStr+"; "
        aValue=PtFTab.ReturnValue(aFld, 0)
        DtStr=DtStr+aValue.AsString+"; "
     end
  end

  MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
           +NL+FldStr+NL+NL+
           "Der erste Datensatz:"+NL+DtStr,
           "Information")
  aPtKWFld = MsgBox.ListAsString(ListofPtFld,
    "für Klassifizierung der Legende",
    "Auswahl eines Feldes des Themas"++PtTheme.AsString)
  aPtKWFldStr = aPtKWFld.AsString

  PtLegend = PtTheme.GetLegend
  PtLegend.SetLegendType(#Legend_Type_Unique)
  PtLegend.Unique(PtTheme, aPtKWFldStr)
  ListofPtKlasse = PtLegend.GetClassifications
  AnzPtKlasse = ListofPtKlasse.Count
  IdxPtKlasse = AnzPtKlasse-2
  'MsgBox.Info(AnzPtKlasse.AsString, "Anzahl der Klasse (Pt)")
```

```
  aListofPtBez = {}
  for each i in 0..IdxPtKlasse
    theKlasseLb=ListofPtKlasse.Get(i).GetLabel
    aListofPtBez.Add(theKlasseLb)
  end

  thePtBz=MsgBox.ListAsString(aListofPtBez,
    "zur Auswahl der Punkte",
    "Auswahl der Bezeichnung der Punkte im"++PtTheme.AsString)
  if (thePtBz = nil) then
    MsgBox.Error("Die Bezeichnung der Punkte oder"+NL+
           "der Punkt mit der Bezeichnung fehlt!"+NL+
           "Das Programm wird abgebrochen!", "")
    exit
  end

  'Der nächeste Punkt vom Maus-Klicken wird in der Tabelle
  'der Punkt-Datei im aktiven View gesucht.

  minAbst=1000000
  for each Pt in 0..AnzPtIdx
     aPtBz=PtFTab.ReturnValue(aPtKWFld, Pt)
     if (aPtBz = thePtBz) then
       thePt = PtFTab.ReturnValue(PtShpFld, Pt)
       thePtx = thePt.Getx
       thePty = thePt.Gety
       Abst=(((thePtx-aMPtx) ^ 2) + ((thePty-aMPty) ^ 2)).Sqrt.Abs
       if (Abst < minAbst) then
         PtIdx = Pt
         minAbst = Abst
       end
     end
  end
  thePt = PtFTab.ReturnValue(PtShpFld, PtIdx)
  theX = thePt.Getx
  theY = thePt.Gety
  'MsgBox.Report("X-Koordinate"++": "++theX.AsString+NL+
  '             "Y-Koordinate"++": "++theY.AsString,
  '             "Der Punkt in der Tabelle der Punkt-Datei")

 aNPt = Point.Make(theX, theY)

elseif ((ErstPt = "ein Punkt auf einer Polyline") or
     (ErstPt = "ein Punkt auf einem Polygon")) then

  'Auswahl eines Polyline- oder eines Polygon-Themas im aktiven View


  if (ErstPt = "ein Punkt auf einer Polyline") then
    PLTheme = MsgBox.ChoiceAsString(ListofPLThms,
      "um Koordinaten eines Punktes zu übernehmen",
      "Auswahl eines Polyline-Themas im View"++AkView.AsString)

    ListofPtPl = {"ein Stützpunkt am Maus-Klicken",
                  "ein Punkt zwischen den Stützpunkten"}


  elseif (ErstPt = "ein Punkt auf einem Polygon") then
    PLTheme = MsgBox.ChoiceAsString(ListofPgThms,
      "um Koordinaten eines Punktes zu übernehmen",
      "Auswahl eines Polygon-Themas im View"++AkView.AsString)
```

```
  ListofPtPl = {"ein Stützpunkt am Maus-Klicken",
                "ein Punkt zwischen den Stützpunkten",
                "ein Stützpunkt mit der kleinsten x-Koord.",
                "ein Stützpunkt mit der größten x-Koord."}
end

ZweitAw = MsgBox.ListAsString(ListofPtPl,
     "um die Koordinaten zu übernehmen",
     "Auswahl eines Punktes am Maus-Klicken")

aTStr = PLTheme.AsString

if (PLTheme = nil) then
  MsgBox.Error("Der Name der Polyline- oder Polygon-Datei fehlt!"
        +NL+"Das Programm wird abgebrochen!", "")
  exit
end

PLFTab = PLTheme.GetFTab
ListofPLFld = PLFTab.GetFields
PLShpFld=ListofPLFld.Get(0)

PLShpClassStr = PLFTab.GetShapeClass.GetClassName
if (PLShpClassStr = "Polyline") then
  Art = "der Polyline"
  Art2 = "die Polyline"
elseif (PLShpClassStr = "Polygon") then
  Art = "des Polygons"
  Art2 = "das Polygon"
end

AnzPL=0
for each PL in PLFTab
   AnzPL = AnzPL+1
end
AnzPLIdx = AnzPL - 1
AnzPLStr = AnzPL.SetFormat("d").AsString
'MsgBox.Info(AnzPLStr, "Die Anzahl"++Art++"im Thema"++aTStr)

'Die Bezeichnung der Polyline oder des Polygons wird
'in der Tabelle gesucht.

AnzPLFld = ListofPLFld.Count
IdxPLFld = AnzPLFld - 1

'Feststellung des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxPLFld
   aFld=ListofPLFld.Get(i)
   aFldStr=aFld.GetName
   if (aFldStr <> "Shape") then
      FldStr = FldStr + aFldStr+"; "
      aValue=PLFTab.ReturnValue(aFld, 0)
      DtStr=DtStr+aValue.AsString+"; "
   end
end

MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
        +NL+FldStr+NL+NL+
        "Der erste Datensatz:"+NL+DtStr,
        "Information")
```

```
aPLKWFld = MsgBox.ListAsString(ListofPLFld,
  "für Klassifizierung der Legende, um einen Punkt zu übernehmen",
  "Auswahl eines Feldes des Themas"++aTStr)
aPLKWFldStr = aPLKWFld.AsString

PLLegend = PLTheme.GetLegend
PLLegend.SetLegendType(#Legend_Type_Unique)
PLLegend.Unique(PLTheme, aPLKWFldStr)
ListofPLKlasse = PLLegend.GetClassifications
AnzPLKlasse = ListofPLKlasse.Count
IdxPLKlasse = AnzPLKlasse-2
'MsgBox.Info(AnzPLKlasse.AsString, "Anzahl der Klasse (PL)")

aListofPLKl = {}
for each i in 0..IdxPLKlasse
  theKlasseLb=ListofPLKlasse.Get(i).GetLabel
  aListofPLKl.Add(theKlasseLb)
end


thePLKl=MsgBox.ListAsString(aListofPLKl,
   "im"++aTStr++"zur Auswahl"++Art++"um einen Punkt zu übernehmen",
   "Auswahl einer Klasse der Datensätze")
if (thePLKl = nil) then
  MsgBox.Error("Die Klasse"++Art++"oder"
     +NL+Art2++"mit der Klasse fehlt!"
     +NL+"Das Programm wird abgebrochen!", "")
  exit
end

'Auswahl der Datensätze im Thema mit den Maus-Klicken
ListofSBz = {}
ListofIdxS = {}

if (PLTheme.CanSelect) then
 'MsgBox.Info("Das Thema"++aTStr++"ist selectierbar.","CanSelect?")

 PLTheme.SelectByPolygon(aMCPg, #VTAB_SELTYPE_NEW)

 'die ausgewählten Datensätze
 AnzS = 0
 for each rec in PLFTab.GetSelection
   'MsgBox.Info("von"++aTStr++":  "++rec.AsString,
   '    "Index-Nummer der selektierten Datensätze")
   aSBz = PLFTab.ReturnValue(aPLKWFld, rec)
   ListofSBz.Add(aSBz)
   AnzS = AnzS + 1
   ListofIdxS.Add(rec.AsString)
 end
 'MsgBox.Info(AnzS.AsString, "Anzahl der selektierten Datensätze")
 'MsgBox.ListAsString(ListofIdxS, "Index-Nummer der selektierten"
 '                 ++"Datensätze"++"von"++aTStr, "Information")
 if (AnzS = 0) then
   for each i in 0..AnzPLIdx
     aBz = PLFTab.ReturnValue(aPLKWFld, i)
     if (aBz = thePLKl) then
       ListofSBz.Add(aBz)
       ListofIdxS.Add(i.AsString)
     end
   end
 end
else
```

```
  for each i in 0..AnzPLIdx
    aBz = PLFTab.ReturnValue(aPLKWFld, i)
    if (aBz = thePLKl) then
       ListofSBz.Add(aBz)
       ListofIdxS.Add(i.AsString)
    end
  end
end

aSBz = MsgBox.ListAsString(ListofSBz,
     "von"++aTStr++","++
     "um"++Art2++"auszuwählen"++"um einen Punkt zu übernehmen",
     "Auswahl eines Datensatzes")
aSBzIdx = ListofSBz.FindByValue(aSBz)
aSRIdx = ListofIdxS.Get(aSBzIdx).AsNumber

'MsgBox.Info("von"++aTStr++":  "++aSRIdx.AsString,
'     "Index des ausgewählten Datensatzes")
theShp = PLFTab.ReturnValue(PLShpFld, aSRIdx)
'MsgBox.report("des Themas"++aTStr+NL+
'             "Index-Nummer:"++aSRIdx.AsString+NL+NL+
'             theShp.AsString, "Selektiertes Shape")

PLTheme.ClearSelection

'Feststellung der Koordinaten der ausgewählten Formen im Thema

theProfilList1={}
theProfilList1.Add({theShp})

'Liste der Stützpunkte
ListofFLPt = {}
minxkrd = 100000000
maxxkrd = 0

if (theProfilList1 <> 0) then
  for each q in theProfilList1
    theLines=q.Get(0).AsList
    for each m in theLines
       for each ptx in m
          myx=ptx.Getx
          myy=ptx.Gety
          ListofFLPt.Add(myx@myy)
          if (myx < minxkrd) then
            minxkrd = myx
            minykrd = myy
          end
          if (myx > maxxkrd) then
            maxxkrd = myx
            maxykrd = myy
          end
       end
    end
  end
end

PtAnz= ListofFLPt.Count  'Anzahl der Stützpunkte der Polyline

if (ErstPt = "ein Punkt auf einer Polyline") then
  PtAnzIdx= PtAnz - 1
elseif (ErstPt = "ein Punkt auf einem Polygon") then
  PtAnzIdx= PtAnz - 2
```

```
end

'Bestimmung der Stellen an Maus-Klicken

minDist = 1000000
minIdx = -1

aMPtx = aMausPt.Getx
aMPty = aMausPt.Gety

for each i in 0..PtAnzIdx
    aPt = ListofFLPt.Get(i)
    xkrd= aPt.Getx
    ykrd= aPt.Gety
    xAbst = ((xkrd- aMPtx)* (xkrd- aMPtx))
    yAbst = ((ykrd- aMPty)* (ykrd- aMPty))
    xyAbst = ((xAbst  + yAbst).sqrt) .Abs
    if (xyAbst < minDist) then
      minDist = xyAbst
      minIdx = i  'Index des nähesten Punktes des Maus-Klickens
    end
end

aNPt = ListofFLPt.Get(minIdx)

if (ErstPt = "ein Punkt auf einem Polygon") then
 if (ZweitAw = "ein Stützpunkt mit der kleinsten x-Koord.") then
   aNPt = Point.Make(minxkrd, minykrd)
 elseif (ZweitAw = "ein Stützpunkt mit der größten x-Koord.") then
   aNPt = Point.Make(maxxkrd, maxykrd)
 end
end

if (ZweitAw = "ein Punkt zwischen den Stützpunkten") then

  if (minIdx = 0) then
    vIdx = 0     'ein Vertex auf der Linie vor dem Maus-Klicken
    nIdx = 1     'ein Vertex auf der Linie nach dem Maus-Klicken
  elseif ((minIdx > 0)  and (minIdx < PtAnzIdx)) then
    aPtv = ListofFLPt.Get((minIdx - 1))
    aPt0 = ListofFLPt.Get(minIdx)
    aPtn = ListofFLPt.Get((minIdx + 1))

    xkrdv = aPtv.Getx   'x-Koord. des letzten Punktes
    xkrd  = aPt0.Getx   'x-Koord. des nähesten Punktes
    xkrdn = aPtn.Getx   'x-Koord. des nächsten Punktes

    ykrdv = aPtv.Gety   'y-Koord. des letzten Punktes
    ykrd  = aPt0.Gety          'y-Koord. des nähesten Punktes
    ykrdn = aPtn.Gety   'y-Koord. des nächsten Punktes

    xAv = (xkrdv - xkrd) * (xkrdv - xkrd)
    yAv = (ykrdv - ykrd) * (ykrdv - ykrd)
    xyAv = ((xAv + yAv).sqrt).Abs  'Entfernung zum letzten Punkt vom nähesten Punkt

    xAn = (xkrdn - xkrd) * (xkrdn - xkrd)
    yAn = (ykrdn - ykrd) * (ykrdn - ykrd)
    xyAn = ((xAn + yAn).sqrt).Abs  'Entfernung zum nächsten Punkt vom nähesten Punkte

    xML = (xkrd - aMPtx) * (xkrd - aMPtx)
    yML = (ykrd - aMPty) * (ykrd - aMPty)
    xyML = ((xML + yML).sqrt).Abs  'Entfernung zum Maus-Klicken vom nähesten Punkt
```

```
  xMLv = (xkrdv - aMPtx) * (xkrdv - aMPtx)
  yMLv = (ykrdv - aMPty) * (ykrdv - aMPty)
  xyMLv = ((xMLv + yMLv).sqrt).Abs  'Entfernung zum Maus-Klicken vom letzten Punkt

  xMLn = (xkrdn - aMPtx) * (xkrdn - aMPtx)
  yMLn = (ykrdn - aMPty) * (ykrdn - aMPty)
  xyMLn = ((xMLn + yMLn).sqrt).Abs 'Entfernung zum Maus-Klicken vom nächsten Punkt

  vLmML = (xyAv - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken und dem
  vLpML = xyAv + xyML        'letzten Punkt

  nLmML = (xyAn - xyML).Abs  'theoretische Länge zwischen dem Maus-Klicken und dem
  nLpML = xyAn + xyML        'nächsten Punkt

  vM = (vLmML - xyMLv).Abs   'Differenz zwischen der theorischen und der
  vP = (vLpML - xyMLv).Abs   'tatsächlichen Länge im Bezug auf den letzten Punkt

  nM = (nLmML - xyMLn).Abs   'Differenz zwischen der theorischen und der
  nP = (nLpML - xyMLn).Abs   'tatsächlichen Länge im Bezug auf den nächsten Punkt

  ListofLg = {vM, vP, nM, nP}   'Suche nach der kleinsten Länge
  MinLg = 10000
  TheLgIdx = -1
  For each aLg in 0..3
     theLg = ListofLg.Get(aLg)
     if (theLg < MinLg) then
          MinLg = theLg
          TheLgIdx = aLg
     end
  end
  if (xyML < 10) then
    vIdx = minIdx - 1
    nIdx = minIdx + 1
  elseif (xyML >= 10) then
    if ((TheLgIdx = 0) or (TheLgIdx = 3)) then
        vIdx = minIdx - 1     'ein Vertex auf der Linie vor dem Maus-Klicken
        nIdx= minIdx          'ein Vertex auf der Linie nach dem Maus-Klicken
    elseif ((TheLgIdx = 1) or (TheLgIdx = 2)) then
        vIdx = minIdx         'ein Vertex auf der Linie vor dem Maus-Klicken
        nIdx = minIdx + 1     'ein Vertex auf der Linie nach dem Maus-Klicken
    else
        vIdx = minIdx         'ein Vertex auf der Linie vor dem Maus-Klicken
        nIdx = minIdx + 1     'ein Vertex auf der Linie nach dem Maus-Klicken
    end
  end
elseif (minIdx = PtAnzIdx)   then
   vIdx = minIdx - 1
   nIdx = minIdx
end

'Bestimmung der Koordinaten des Maus-Klickens auf dem Profilschnitt
vPt = ListofFLPt.Get(vIdx)
vPtx = vPt.Getx
vPty = vPt.Gety
nPt = ListofFLPt.Get(nIdx)
nPtx = nPt.Getx
nPty = nPt.Gety

if ((minDist = 0) or (minDist < 10)) then 'Der näheste Punkt liegt innerhalb
   mPt = ListofFLPt.Get(minIdx)
   Ptx = mPt.Getx     '10 m vom dem Maus-Klicken auf der Linie
```

```
      Pty = mPt.Gety
      if ( minIdx = 0) then
        vGrIdx = minIdx
        nGrIdx = minIdx + 1
      elseif (( minIdx > 0) and ( minIdx < PtAnzIdx)) then
        vGrIdx = minIdx -1
        nGrIdx = minIdx + 1
      elseif ( minIdx = PtAnzIdx) then
        vGrIdx = minIdx -1
        nGrIdx = minIdx
      end
    elseif (minDist >= 10) then  'Berechnung der Koordinaten des Punktes
      vGrIdx = vIdx      'auf der Linie als Projektion des Maus-Klickens
      nGrIdx = nIdx          'auf die Linie (Profilschnitt)
      if  (vPtx  = aMPtx) then
          Ptx = vPtx
          Pty = vPty
      elseif ((vPtx  <> aMPtx) and (aMPtx <> nPtx)) then
          if (vPty = nPty) then
                Ptx = aMPtx
                Pty = vPty
          elseif (vPty < nPty) then
                if (nPtx = vPtx) then
                      Ptx = nPtx
                      Pty = nPty
                elseif (nPtx <> vPtx) then
                      xnvDiff = nPtx - vPtx
                      ynvDiff = nPty - vPty
                      xMvDiff = aMPtx - vPtx
                      Ptx = aMPtx
                      Pty  = (ynvDiff / xnvDiff) * xMvDiff + vPty
                end
          elseif (vPty > nPty) then
                if (nPtx = vPtx) then
                      Ptx = nPtx
                      Pty = nPty
                elseif (nPtx <> vPtx) then
                      xnvDiff = nPtx - vPtx
                      yvnDiff = vPty - nPty
                      xnMDiff = nPtx - aMPtx
                      Ptx = aMPtx
                      Pty  = (yvnDiff / xnvDiff) * xnMDiff + nPty
                end
          end
      elseif (nPtx  = aMPtx)  then
          Ptx = nPtx
          Pty = nPty
      end
    end          ' Ende von (if ((minDist = 0) or (minDist < 10)) then)

    aNPt = Point.Make(Ptx, Pty)

  end
end

'Speicherung des neuen Punktes
ListofSp11 = {"Herstellung als ein neues Thema",
              "Speicherung in einem vorhandenen Thema"}

AW11 = MsgBox.ListAsString(ListofSp11,
      "zur Speicherung des neuen Punktes",
      "Auswahl eines Themas")
```

```
if (AW11 = "Herstellung als ein neues Thema") then
  WDStr=av.GetProject.GetWorkDir.AsString
  fn00 = "Ptgrnt"
  fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
  'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
  if (fName =nil) then exit end
  fName.SetExtension("shp")
  NFTab=FTab.MakeNew(fName, Point)

  NShpFld = NFTab.FindField("shape")
  NIDFld=Field.Make("ID", #Field_Short, 3, 0)
  NKWFld=Field.Make("Kennwort", #Field_Char, 10, 0)
  ListofNFlds={NIDFld, NKWFld}
  NFTab.AddFields(ListofNFlds)
  NPtTheme = FTheme.Make(NFTab)
  AkView.AddTheme(NPtTheme)
  recNr = -1

elseif (AW11 = "Speicherung in einem vorhandenen Thema") then
  NPtTheme=MsgBox.Choice(ListofPtThms, "Name eines Punkt-Themas:"
        +NL+"um einen Punkt zu speichern"+NL+"("+ErstPt+")",
        "Auswahl eines Themas im View"++AkView.AsString)
  'MsgBox.Info(NPtTheme.AsString, "Der Name des Punkt-Themas")

   NFTab = NPtTheme.GetFTab

   'Anzahl der Felder im Thema
   ListofNFlds = NFTab.GetFields
   AnzNFlds = ListofNFlds.Count
   IdxNFlds = AnzNFlds - 1

   NShpFld = ListofNFlds.Get(0)
   NIDFld = NFTab.FindField("ID")

   'Anzahl der Datensätze im Thema
   AnzNDs=0
   for each rec in NFTab
     AnzNDs = AnzNDs + 1
   end
   IdxNDs = AnzNDs - 1

   NKWFld = MsgBox.ListAsString(ListofNFlds,
        "für Kennwort des Punktes",
        "Auswahl eines Feldes des Themas"++NPtTheme.AsString)
   recNr = IdxNDs
end

av.ShowMsg("Herstellung des neuen Themas für neue Punkte ...")
av.ShowStopButton

theKW = MsgBox.Input("für eine Bezeichnung des neuen Punktes",
      "Eingabe eines Kennwortes", "Grptnt")

NFTab.SetEditable(false)
NFTab.SetEditable(true)

recNr = recNr + 1
NFTab.AddRecord
NFTab.SetValue(NShpFld, recNr, aNPt)
NFTab.SetValue(NIDFld, recNr, recNr)
```

```
NFTab.SetValue(NKWFld, recNr, theKW)

NFTab.SetEditable(false)

av.ShowMsg("Veränderung der Legenden des Themas"++NPtTheme.AsString+"...")

theLegend = NPtTheme.GetLegend
theLegend.SetLegendType(#LEGEND_TYPE_SIMPLE)
theLegend.SingleSymbol
theSymbol=theLegend.GetSymbols.Get(0)
theSymbol.SetColor((Color.GetBlue))

NPtTheme.UpdateLegend




'ptausmpt.ave
'Punkte, die sich in einer Punkt-Datei in der Nähe der Maus-Klicken befinden,
'werden in einem aktiven View gesucht und in einem Punkt-Thema gespeichert.
'Dieses Script wird als ein Werkzeug (Tool) im aktiven View benutzt.

theProject = av.GetProject
theView = av.GetActiveDoc   'Ein aktives View

'mehrmalige Klicken auf das Bildschirm im aktiven View zur Bestimmung der Stelle
aEingPolyL = theView.GetDisplay.ReturnUserPolyLine

ListofListofEingPt = aEingPolyL.AsList
ListofEingPt = ListofListofEingPt.Get(0)
AnzM = ListofEingPt.Count
AnzMIdx = AnzM - 1
ListofMPt = {}

for each mPt in 0..AnzMIdx
  aMPt = ListofEingPt.Get(mPt)
  ListofMPt.Add(aMPt)
end

ListofThemes=theView.GetThemes
ListofPtThms = {}

av.ShowMsg("Auswahl eines Themas ...")
for each aT in ListofThemes
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     aShpClassStr = aFTab.GetShapeClass.GetClassName
     if (aShpClassStr = "Point") then
       ListofPtThms.Add(aT)
     end
   end
end

'Auswahl eines Punktthemas im View,
'um Punkte im Thema in der nächesten Entfernung
'von Maus-Klicken zu holen.

thePtTheme=MsgBox.Choice(ListofPtThms, "Name eines Punkt-Themas:"
        +NL+"um Punkte an Maus-Klicken zu übernehmen",
        "Auswahl eines Themas im View"++theView.AsString)
'MsgBox.Info(thePtTheme.AsString, "Der Name des Punkt-Themas")
```

```
PtFTab=thePtTheme.GetFTab
ListofPtFlds = PtFTab.GetFields
PtShpFld = ListofPtFlds.Get(0) 'Feldauswahl für evtl. Ereignisthema

'Feststellung der Anzahl der Punkte in dem Thema
AnzThPt=0
for each rec in PtFTab
   AnzThPt=AnzThPt+1
end
ThPtIdx=AnzThPt-1

av.ShowMsg("Bestimmung der Punkte in der Nähe der Mausklicken in"
      ++thePtTheme.AsString++"...")
av.ShowStopButton
ListofNPT={}
MinAbst=22600.00
Ng=0
For each aMPt in 0..AnzMIdx
   Ng=Ng+1
   aPt = ListofMPt.Get(aMPt)
   axMPt=aPt.Getx
   ayMPt=aPt.Gety
   For each aThPt in 0..ThPtIdx
      thePtShape=PtFTab.ReturnValue(PtShpFld, aThPt)
      axThPt=thePtShape.Getx
      ayThPt=thePtShape.Gety
      xAbst=(axMPt-axThPt).Abs
      if (xAbst < 1000) then
         yAbst=(ayMPt-ayThPt)
         PtAbst=(((xAbst*xAbst)+(yAbst*yAbst)).Sqrt).Abs
         if (PtAbst < MinAbst) then
            MinAbst=PtAbst
            xMinThPt=axThPt
            yMinThPt=ayThPt
         end
      end
   end
   ListofNPT.Add(xMinThPt@yMinThPt)
   MinAbst=22600.00
   'Show percentage complete with enabled stop button
   more=av.SetStatus(Ng/AnzM*100)
   if (not more) then
      break
   end
end

'Speicherung der neuen Punkte
ListofSp11 = {"Herstellung als ein neues Thema",
            "Speicherung in einem vorhandenen Thema"}

AW11 = MsgBox.ListAsString(ListofSp11,
     "zur Speicherung der neuen Punkte",
     "Auswahl eines Themas")

if (AW11 = "Herstellung als ein neues Thema") then
  WDStr=av.GetProject.GetWorkDir.AsString
  fn00 = "Pt"+thePtTheme.AsString.Left(4)
  fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
  'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
  if (fName =nil) then exit end
```

```
  fName.SetExtension("shp")
  NFTab=FTab.MakeNew(fName, Point)

  NShpFld = NFTab.FindField("shape")
  NIDFld=Field.Make("ID", #Field_Short, 3, 0)
  NKWFld=Field.Make("Kennwort", #Field_Char, 10, 0)
  ListofNFlds={NIDFld, NKWFld}
  NFTab.AddFields(ListofNFlds)
  NPtTheme = FTheme.Make(NFTab)
  theView.AddTheme(NPtTheme)
  recNr = -1

elseif (AW11 = "Speicherung in einem vorhandenen Thema") then
  NPtTheme=MsgBox.Choice(ListofPtThms, "Name eines Punkt-Themas:"
        +NL+"um Punkte an Maus-Klicken zu speichern",
        "Auswahl eines Themas im View"++theView.AsString)
  'MsgBox.Info(NPtTheme.AsString, "Der Name des Punkt-Themas")

   NFTab = NPtTheme.GetFTab

   'Anzahl der Felder im Thema
   ListofNFlds = NFTab.GetFields
   AnzNFlds = ListofNFlds.Count
   IdxNFlds = AnzNFlds - 1

   NShpFld = ListofNFlds.Get(0)
   NIDFld = NFTab.FindField("ID")

   'Anzahl der Datensätze im Thema
   AnzNDs=0
   for each rec in NFTab
     AnzNDs = AnzNDs + 1
   end
   IdxNDs = AnzNDs - 1

   NKWFld = MsgBox.ListAsString(ListofNFlds,
        "für Kennwort der Punkte",
        "Auswahl eines Feldes des Themas"++NPtTheme.AsString)
   recNr = IdxNDs
end

av.ShowMsg("Herstellung des neuen Themas für neue Punkte ...")
av.ShowStopButton

AnzNPt = ListofNPT.Count
IdxNPt = AnzNPt - 1
MsgBox.Info(AnzNPt.AsString, "Anzahl der neuen Punkte")

theKW = MsgBox.Input("für eine Bezeichnung der neuen Punkte",
      "Eingabe eines Kennwortes", "Grptnt")


NFTab.SetEditable(false)
NFTab.SetEditable(true)

for each i in 0..IdxNPt
   recNr = recNr + 1
   aNPt = ListofNPT.Get(i)
   NFTab.AddRecord
   NFTab.SetValue(NShpFld, recNr, aNPt)
   NFTab.SetValue(NIDFld, recNr, recNr)
   NFTab.SetValue(NKWFld, recNr, theKW)
```

```
end

NFTab.SetEditable(false)

av.ShowMsg("Veränderung der Legenden des Themas"++NPtTheme.AsString+"...")

theLegend = NPtTheme.GetLegend
theLegend.SetLegendType(#LEGEND_TYPE_SIMPLE)
theLegend.SingleSymbol
theSymbol=theLegend.GetSymbols.Get(0)
theSymbol.SetColor((Color.GetBlue))

NPtTheme.UpdateLegend




'ptentfkt.ave
'Aus Punkten in einem Karten-View werden Entfernungen berechnet.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc

thePtThm = theView.GetActiveThemes.Get(0)
ThStr = thePtThm.AsString

kt00=MsgBox.YesNo("Ist das aktive Thema"++ThStr
     +NL+"zur Berechnung der Entfernung der Punkte vom Anfang richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
            "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

PtFTab = thePtThm.GetFTab          'Tabelle für das Punkt-Thema
ListofPtFlds = PtFTab.GetFields    'Liste der Überschrift der Tabelle

PtShpFld = ListofPtFlds.Get(0)
PtIdFld = PtFTab.FindField("ID")

AnzPt = 0
for each rec in PtFTab
   AnzPt = AnzPt + 1
end
IdxPt = AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"++ThStr)

ListofEntf = {}
vx = 0
vy = 0
for each i in 0..IdxPt
   aPt = PtFTab.ReturnValue(PtShpFld, i)
   ax = aPt.Getx
   ay = aPt.Gety
   if (i = 0) then
      EntfSum = 0
   elseif (i > 0) then
      Entf = (((vx - ax) ^ 2)+((vy - ay) ^2)).Sqrt.Abs
      EntfSum = EntfSum + Entf
```

```
    end
    vx = ax
    vy = ay
    ListofEntf.Add(EntfSum)
end

av.ShowMsg("Speicherung der Punkte im Thema"++ThStr++"...")

AnzPtFld = ListofPtFlds.Count
IdxPtFld = AnzPtFld - 1

'Feststellung der Felder und des ersten Datensatzes
FldStr=""
DtStr=""

for each i in 0..IdxPtFld
  aPtFld = ListofPtFlds.Get(i)
  aPtFldStr = aPtFld.GetName
  if (aPtFldStr <> "Shape") then
    FldStr = FldStr + aPtFldStr+"; "
    aValue = PtFTab.ReturnValue(aPtFld, 0)
    DtStr = DtStr + aValue.AsString+"; "
  end
end

MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
            +NL+FldStr+NL+NL+
            "Der erste Datensatz:"+NL+DtStr,
            "Information")

Fr11 = MsgBox.YesNo("Gibt es ein Feld für"++
      "Entfernungen [m] der Punkte vom Anfang im Punkt-Thema:"
      ++ThStr+"?", "Feststellung des Feldes", false)

if (Fr11) then
  EntfFld = MsgBox.ListAsString(ListofPtFlds,
          "für Entfernung [m] der Punkte vom Anfang",
          "Auswahl eines Feldes im Thema"++ThStr)
elseif (Not Fr11) then
  PtFTab.SetEditable(false)
  PtFTab.SetEditable(true)
  EntfFld = Field.Make("Entfern_m", #Field_Float, 9, 2)
  PtFTab.AddFields({EntfFld})
  PtFTab.SetEditable(false)
end

PtFTab.SetEditable(false)
PtFTab.SetEditable(true)

for each i in 0..IdxPt
  aEntf = ListofEntf.Get(i)
  PtFTab.SetValue(EntfFld, i, aEntf)
end

PtFTab.SetEditable(false)
thePtThm.UpdateLegend
```

```
'pthbeskt.ave
'Die Höhen der Punkte werden durch eine interpolierten Fläche
'(Grid oder Tin) in einem aktiven Karten-View bestimmt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject = av.GetProject
theView = av.GetActiveDoc   'Ein aktives Karten-View

'Eingabe einer Punkt-Datei
PtTheme = theView.GetActiveThemes.Get(0)
PtStr = PtTheme.AsString

qt00=MsgBox.YesNo("Ist das aktive Thema"++PtStr
     +NL+"zur Bestimmung der Höhen der Punkte richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not qt00) then
  MsgBox.Error("Das aktive Thema"++PtStr++"ist falsch!"+NL+
           "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

PtFTab=PtTheme.GetFTab
ListofFelds=PtFTab.GetFields
PtShpFld=ListofFelds.Get(0)

PtHFld=MsgBox.ChoiceAsString(ListofFelds, "um Höhe [m] zu speichern",
      "Auswahl eines Feldes des Themas"++PtTheme.AsString)
PtHFktFld=MsgBox.ChoiceAsString(ListofFelds,
     "um erhöhte Höhen zu speichern"+NL+"(z. B. 50 fache Überhöhung)",
     "Auswahl eines Feldes des Themas"++PtTheme.AsString)

AnzPt=0     'Feststellung der Anzahl der Punkte in dem Thema
for each rec in PtFTab
   AnzPt=AnzPt+1
end
AnzPtIdx=AnzPt-1
'MsgBox.Info(AnzPt.AsString, "Anzahl der Punkte in dem Thema")

YFaktStr="50"
YFtStr=MsgBox.Input("zur Überhöhung der Höhen",
     "Eingabe eines Faktors", YFaktStr)
YFt=YFtStr.AsNumber

'Eingabe der interpolierten Datei mit Höhenwerten (Grid oder Tin)
done=False
While (not done)
  surfaceList = {}
  for each t2 in theView.GetThemes
    if (t2.Is(GTheme) or t2.Is(STheme)) then
      surfaceList.Add(t2)
    end
  end
  if (surfaceList.Count = 0) then
    aSurfFN = SourceManager.GetDataSet({Grid,Tin},"Select Surface :"
          ++ PtTheme.GetName)
    if (aSurfFN = NIL) then
      continue
    end
    aSrcName = Grid.MakeSrcName(aSurfFN.AsString)
    if (aSrcName <> NIL) then
      theSurface = Grid.Make(aSrcName)
      surfTheme = GTheme.Make(theSurface)
```

```
    else
      aSrcName = SrcName.Make(aSurfFN.AsString)
      theSurface = Tin.Make(aSrcName)
      surfTheme = STheme.Make(theSurface)
    end
    theView.AddTheme(surfTheme)
  else
    surfTheme = MsgBox.ListAsString(surfaceList,
            "Choose theme to use as surface:",
            "Select Surface :" ++ PtTheme.GetName)
    if (surfTheme = NIL) then
      continue
    end
    if (surfTheme.Is(GTheme)) then
      theSurface = surfTheme.GetGrid
    elseif (surfTheme.Is(STheme)) then
      theSurface = surfTheme.GetSurface
    else
      continue
    end
  end
  done=True
end

av.ShowMsg("Zuweisung der Z Werten in Tabelle ...")
av.ShowStopButton
aPrj=Prj.MakeNull
PtFTab.SetEditable(false)
PtFTab.SetEditable(true)
for each gzd in 0..AnzPtIdx
  Ng=gzd+1
  'Ablesen der Punkte vom Input-Thema
  thePt=PtFTab.ReturnValue(PtShpFld, gzd)

  if (surfTheme.Is(GTheme)) then
    theZValue=theSurface.PointValue(thePt, aPrj).SetFormat("d.dd")
  elseif (surfTheme.Is(STheme)) then
    theZValue=theSurface.Elevation(thePt).SetFormat("d.dd")
  end

  theZFkt=(theZValue* YFt).SetFormat("d.dd")
  PtFTab.SetValue(PtHFld, gzd, theZValue)
  PtFTab.SetValue(PtHFktFld, gzd, theZFkt)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzPt*100)
  if (not more) then
    break
  end
end
PtFTab.SetEditable(false)
PtTheme.UpdateLegend
```

```
'ptkorrig.ave
'Die Punkte in einem Punktthema werden nach einer Korrektur
'(z.B. einer Glättung) mit den Punkten vor der Korrektur in einem
'anderen Punkt-Thema verglichen. Die korrigerten Punkten werden
'in einem neuen Punkt-Thema gespeichert.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.
```

```
theProject=av.GetProject
myScript=theProject.FindScript("ptkorrig")
myScript.SetNumberFormat( "d.dd") ' script default

theView=av.GetActiveDoc    'ein aktives View

'Auswahl eines Punktthemas vor der Korrektur
ListofThms = theView.GetThemes
ListofPtThms = {}

for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofPtThms.Add(aT)
      end
   end
end

ListofZeit = {"vor der Korrektur", "nach der Korrektur"}
ListofListofPt = {}
ListofThemen = {}

for each j in 0..1
  aZ = ListofZeit.Get(j)
  aPtThm = MsgBox.ChoiceAsString(ListofPtThms,
        aZ+NL+
       "Der Name eines Punkt-Themas in View"++theView.AsString,
       "Eingabe einer Punkte-Datei")

  if (aPtThm = nil) then
    MsgBox.Error("Der Name der Punkte-Datei fehlt!"+NL+
              "Das Programm wird abgebrochen!", "")
    exit
  end
  ListofThemen.Add(aPtThm)

  PtFTab = aPtThm.GetFTab
  ListofPtFlds = PtFTab.GetFields
  PtShpFld = ListofPtFlds.Get(0)

  AnzPt=0
  for each Pt in PtFTab
    AnzPt=AnzPt+1
  end
  AnzPtIdx=AnzPt-1
  MsgBox.Report(aPtThm.AsString++aZ+NL+AnzPt.AsString,
            "Die Anzahl der Punkte im Thema")

  ListofPt = {}

  for each aR in 0..AnzPtIdx
     aPt = PtFTab.ReturnValue(PtShpFld, aR)
     ListofPt.Add(aPt)
  end
  ListofListofPt.Add(ListofPt)

end 'das Ende der j-Schleife

'Die Punkte in den beiden Themen werden verglichen.
'Aus den Unterschieden wird ein neues Thema aus den unterschiedlichen
```

```
'Punkten hergestellt.

av.ShowMsg("Vergleich der Punkte in den beiden Themen und Herstellung"
        ++"eines neuen Thema"++"...")
av.ShowStopButton


aListofvPt = ListofListofPt.Get(0)
aListofnPt = ListofListofPt.Get(1)

AnzvPt = aListofvPt.Count
IdxvPt = AnzvPt - 1

AnznPt = aListofnPt.Count
IdxnPt = AnznPt - 1

Ng=0
SW = "ungl"
ListofIdxungPt = {}

for each j in 0..IdxnPt
  Ng = Ng + 1
  anPt = aListofnPt.Get(j)
  nx = anPt.Getx
  ny = anPt.Gety
  for each i in 0..IdxvPt
    avPt = aListofvPt.Get(i)
    vx = avPt.Getx
    if (vx = nx) then
      vy = avPt.Gety
      if (vy = ny) then
        SW = "gleich"
      end
    end
  end
  if (SW = "ungl") then
     ListofIdxungPt.Add(j)
  elseif (SW = "gleich") then
     SW = "ungl"
  end
  'Show percentage complete with enabled stop button
  more=av.SetStatus((Ng/(AnzvPt))*100)
  if (not more) then
    break
  end
end

av.ShowMsg("Ein neues Punktthema wird hergestellt."++"...")
'Wenn ein Thema bei der Bearbeitung ist,
'wird die Bearbeitung vor der Herstellung eines neuen Themas beendet.
editThm = theView.GetEditableTheme
if (editThm <> nil) then
  doSave = MsgBox.YesNoCancel("Save edits to "+editThm.GetName+"?",
  "Stop Editing",true)
  if (doSave = nil) then
    return nil
  end
  if (editThm.StopEditing(doSave).Not) then
     MsgBox.Info("Unable to Save Edits to "
               + editThm.GetName +
               ", please use the Save Edits As option", "")
     return nil
```

```
  else
     theView.SetEditableTheme(NIL)
  end
end

'Ein neues Punkt-Thema wird in einem aktiven View hergestellt.
aPtThm = ListofThemen.Get(1)
aWDStr=theProject.GetWorkDir.AsString
fnStr00 = aPtThm.AsString.Left(6)
fnStr=FileName.Make(aWDStr).MakeTmp(fnStr00,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName=nil) then exit end
fName.SetExtension("shp")

NPtFTab = aPtThm.ExportToFTab (fName)
ListofNPtFlds = NPtFTab.GetFields

NPtShpFld = NPtFTab.FindField("shape")

av.ShowMsg("Herstellung eines neuen Punkt-Themas ...")
AnzRec = 0
for each rec in NPtFTab
  AnzRec = AnzRec + 1
end
IdxRec = AnzRec - 1
MsgBox.Report("des Themas"++aPtThm.AsString+NL+AnzRec.AsString,
            "Anzahl der Datensätze")

AnzungPt = ListofIdxungPt.Count
IdxungPt = AnzungPt - 1

MsgBox.Info(AnzungPt.AsString, "Anzahl der neuen Datensätze")

NPtFTab.SetEditable(false)
NPtFTab.SetEditable(true)

for each j in 0..IdxRec
   aIdx = IdxRec - j
   aIdxPt = ListofIdxungPt.FindByValue(aIdx)
   if (aIdxPt = -1) then
      NPtFTab.RemoveRecord(aIdx)
   end
end

NPtFTab.SetEditable(false)
NewPtThm = FTheme.Make(NPtFTab)
theView.AddTheme(NewPtThm)

theLegend = NewPtThm.GetLegend
theLegend.SetLegendType(#LEGEND_TYPE_SIMPLE)
theLegend.SingleSymbol
theLegend.GetSymbols.Get(0).SetColor(Color.GetBlue)
NewPtThm.UpdateLegend




'ptlg2d3d.ave
'2D-Punkte, -Polyline oder -Polygone in einem Karten-View werden
'in 3D-Shapes umgewandelt und in einem aktiven 3D-Szene geladen.
'Dieses Skript wird als ein Menü oder als eine Schaltfläche
```

```
'in einem aktiven View zum Anklicken benutzt.

theProject=av.GetProject
myScript=theProject.FindScript("ptlg2d3d")
myScript.SetNumberFormat( "d.dd") ' script default
AkSzene = av.GetActiveDoc   'eine aktive Szene

ListofViews = {"Kt1_gg","Kt1_ggd","Kt1_hg","Kt1_hgd","Kt1_tga","Kt1_tgd",
              "Kt2_gg","Kt2_ggd","Kt2_hg","Kt2_hgd","Kt2_tga","Kt2_tgd"}

aViewAW = MsgBox.ListAsString(ListofViews,
          "in dem sich ein 2D-Thema befindet,"+NL+
          "das in 2D-Thema umgewandelt wird.",
          "Auswahl eines Karten-Views")
theViewStr = MsgBox.Input("ein View für ein 2D-Thema",
            "Veränderungsmöglichkeit", aViewAW)
theView = theProject.FindDoc(theViewStr)

ListofThemes = theView.GetThemes
theThm = MsgBox.ChoiceAsString(ListofThemes, "zur Umwanlung in 3D",
         "Auswahl eines 2D-Themas im View"++theViewStr)
ThStr = theThm.AsString

kt00=MsgBox.YesNo("Ist das aktive Thema"++ThStr
      +NL+"zur Umwndlung 2D- in 3D-Shape richtig?",
      "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
               "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

ThFTab = theThm.GetFTab          'Tabelle für das Thema
ListofThFlds = ThFTab.GetFields    'Liste der Überschrift der Tabelle
AnzThFld = ListofThFlds.Count  'Anzahl der Felder
IdxThFld = AnzThFld - 1
ThShpFld = ListofThFlds.Get(0) 'erstes Shape-Feld beim evtl.Ereignisthema

AnzDs=0
for each rec in ThFTab
   AnzDs = AnzDs + 1
end
IdxDs = AnzDs - 1
AnzDsStr = AnzDs.SetFormat("d").AsString
'MsgBox.Info("im Thema"++ThStr+NL+AnzDsStr, "Die Anzahl der Datensätze")

Listof3DShp = {}
aClassNm = ThFTab.GetShapeClass.GetClassName

if (aClassNm = "Point") then
  ListofzKrd = {"Werte in einem Feld für z-Koord.", "ein gleicher z-Wert"}
  zkrdAw = MsgBox.ListAsString(ListofzKrd,
           "zur Bestimmung der z-Koordinate",
           "Auswahl der Werte im Thema"++ThStr)

  if (zkrdAw = "Werte in einem Feld für z-Koord.") then
    'Die Bezeichnung der Felder wird in der Tabelle gesucht.

    'Feststellung der Felder und des ersten Datensatzes
    FldStr=""
    DtStr=""
    for each i in 0..IdxThFld
```

```
        aFld = ListofThFlds.Get(i)
        aFldStr = aFld.GetName
        if (aFldStr <> "Shape") then
          FldStr = FldStr + aFldStr+"; "
          aValue = ThFTab.ReturnValue(aFld, 0)
          DtStr = DtStr + aValue.AsString+"; "
        end
      end

      MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
          +NL+FldStr+NL+NL+
          "Der erste Datensatz:"+NL+DtStr,
          "Information")

     azWFld = MsgBox.ListAsString(ListofThFlds,
         "zur Bestimmung der z-Koordinate",
         "Auswahl eines Feldes des Themas"++ThStr)

    elseif (zkrdAw = "ein gleicher z-Wert") then
     aZWStr = MsgBox.Input("für z-Koordinate",
               "Eingabe eines Wertes", "0.00")
     aZW = aZWStr.AsNumber
    end

    for each i in 0..IdxDs
      aPt = ThFTab.ReturnValue(ThShpFld, i)
      aPtx = aPt.Getx
      aPty = aPt.Gety
      if (zkrdAw = "Werte in einem Feld für z-Koord.") then
        aPtz = ThFTab.ReturnValue(azWFld, i)
      elseif (zkrdAw = "ein gleicher z-Wert") then
        aPtz = aZW
      end
      Listof3DShp.Add(aPtx@aPty@aPtz)
    end
  elseif ((aClassNm = "Polyline") or (aClassNm = "Polygon")) then
    aZWStr = MsgBox.Input("für z-Koordinate",
          "Eingabe eines Wertes", "0.00")
    aZW = aZWStr.AsNumber

    for each i in 0..IdxDs
       Listof3DPt = {}
       Listof2DShps = {}
       aShp = ThFTab.ReturnValue(ThShpFld, i)
       Listof2DShps.Add({aShp})
       if (Listof2DShps <> 0) then
         for each q in Listof2DShps
          theLines=q.Get(0).AsList
          for each m in theLines
             for each ptx in m
                myx = ptx.Getx
                myy = ptx.Gety
                myz = aZW
                Listof3DPt.Add(myx@myy@myz)
             end
          end
         end
       end
       ListofListof3DPt = {}
       ListofListof3DPt.Add(Listof3DPt)
       if (aClassNm = "Polyline") then
         a3DShp = PolylineZ.Make(ListofListof3DPt)
```

```
      elseif (aClassNm = "Polygon") then
        a3DShp = PolygonZ.Make(ListofListof3DPt)
      end
      Listof3DShp.Add(a3DShp)
    end
end

'Ein Feature-3D-Shape-File für das neue Thema wird hergestellt.
aWDStr = theProject.GetWorkDir.AsString
afnD = ThStr.Left(4)+"3d"
fnStr = FileName.Make(aWDStr).MakeTmp(afnD,"shp")
fnName = FileDialog.Put(fnStr, "*.shp", "Output shape File")
if (fnName = nil) then exit end
fnName.SetExtension("shp")

if (aClassNm = "Point") then
  NThFTab = FTab.MakeNew(fnName, PointZ)
elseif (aClassNm = "Polyline") then
  NThFTab = FTab.MakeNew(fnName, PolylineZ)
elseif (aClassNm = "Polygon") then
  NThFTab = FTab.MakeNew(fnName, PolygonZ)
end
NThShpFld = NThFTab.FindField("shape")
Listof2DFld = {}
ListofN3DFld = {}

for each i in 0..IdxThFld
  aThFld = ListofThFlds.Get(i)
  aThFldStr = aThFld.GetName
  if (aThFldStr <> "Shape") then
    aNFld = aThFld.Clone
    Listof2DFld.Add(aThFld)
    ListofN3DFld.Add(aNFld)
  end
end

NThFTab.AddFields(ListofN3DFld)
AnzN3DFld = ListofN3DFld.Count
IdxN3DFld = AnzN3DFld - 1

NThFTab.SetEditable(false)
NThFTab.SetEditable(true)

for each j in 0..IdxDs
   a3DShp = Listof3DShp.Get(j)
   NThFTab.AddRecord
   NThFTab.SetValue(NThShpFld, j, a3DShp)
   for each i in 0..IdxN3DFld
     aFld = Listof2DFld.Get(i)
     aNFld = ListofN3DFld.Get(i)
     aValue = ThFTab.ReturnValue(aFld, j)
     NThFTab.SetValue(aNFld, j, aValue)
   end
end
NThFTab.SetEditable(false)
N3DThm = FTheme.Make(NThFTab)
AkSzene.AddTheme(N3DThm)
```

```
'ptprjkt.ave
'Punkte in einem aktiven Karten-View werden auf eine
'Gefälls-Linie projiziert.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc     'ein aktives Karten-View
myScript=theProject.FindScript("ptprjkt")
myScript.SetNumberFormat( "d.dd") ' script default

thePtThm = theView.GetActiveThemes.Get(0)
ThStr = thePtThm.AsString

kt00=MsgBox.YesNo("Ist das aktive Thema"++ThStr
     +NL+"zur Projektion der Punkte auf eine Gefälles-Linie richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
            "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

PtFTab = thePtThm.GetFTab          'Tabelle für das Punkt-Thema
ListofPtFlds = PtFTab.GetFields    'Liste der Überschrift der Tabelle

PtShpFld = ListofPtFlds.Get(0)
PtIdFld = PtFTab.FindField("ID")

AnzPt = 0
for each rec in PtFTab
   AnzPt = AnzPt + 1
end
IdxPt = AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"++ThStr)

'Eingabe eines Themas für ein Gefälle im aktiven Karten-View

ListofThms = theView.GetThemes
ListofPLThms = {}

for each aT in ListofThms
   if (aT.Is(FTheme)) then
     aFTab = aT.GetFTab
     aClNm = aFTab.GetShapeClass.GetClassName
     if (aClNm = "PolyLine") then
       ListofPLThms.Add(aT)
     end
   end
end

GfTheme = MsgBox.ChoiceAsString(ListofPLThms,
     "das Gefälle-Linien enthält",
     "Auswahl eines Themas im View"++theView.AsString)

GfFTab = GfTheme.GetFTab
ListofGfFld = GfFTab.GetFields

GfShpFld = GfFTab.FindField("Shape")

AnzGf = 0              'Anzahl der Datensätze
for each rec in GfFTab
  AnzGf = AnzGf + 1
```

```
end
IdxGf = AnzGf - 1

AnzGfFld = ListofGfFld.Count   'Anzahl der Felder
IdxGfFld = AnzGfFld - 1

'Die Bezeichnung der Felder wird in der Tabelle gesucht.

'Feststellung der Felder und des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxGfFld
   aFld = ListofGfFld.Get(i)
   aFldStr = aFld.GetName
   if (aFldStr <> "Shape") then
      FldStr = FldStr + aFldStr+"; "
      aValue = GfFTab.ReturnValue(aFld, 0)
      DtStr = DtStr + aValue.AsString+"; "
   end
end

MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
            +NL+FldStr+NL+NL+
            "Der erste Datensatz:"+NL+DtStr,
            "Information")

aKWFld = MsgBox.ListAsString(ListofGfFld,
       "um eine Gefälls-Linie auszuwählen",
       "Auswahl eines Feldes des Themas"++GfTheme.AsString)

ListofGfFL = {}
for each rec in 0..IdxGf
   aGfFL = GfFTab.ReturnValue(aKWFld, rec)
   ListofGfFL.Add(aGfFL)
end

'Auswahl eines Datensatzes, um den als Gefälle zu benutzen.
theGf = MsgBox.ChoiceAsString(ListofGfFL,
    "um den als Gefälle zu benutzen",
    "Auswahl eines Datensatzes im Thema:"++GfTheme.AsString)
GfIdx = ListofGfFL.FindByValue(theGf)
theShape = GfFTab.ReturnValue(GfShpFld, GfIdx)
theProfilList={}
theProfilList.Add({theShape})
ListofVt={}
'2D-PolyLine wird in Liste der Koordinaten umgewandelt.
if (theProfilList <> 0) then
  for each q in theProfilList
     aListofList=q.Get(0).AsList
     for each aList in aListofList
        for each apt in aList
           ListofVt.Add(apt)
        end
     end
  end
end

AnzGfPt = ListofVt.Count
IdxGfPt = AnzGfPt - 1

AnzGf = IdxGfPt
IdxAnzGf = IdxGfPt - 1
```

```
MsgBox.Info(AnzGf.AsString,
      "Anzahl der Richtungen des Gefälles"++GfTheme.AsString)
LofIdAnfNr={}
LofIdEndNr={}
AnfK=(PtFTab.ReturnValue(PtIdFld, 0).SetFormat("").SetFormat("d")).AsString
EndK=((PtFTab.ReturnValue(PtIdFld, IdxPt))
    .SetFormat("").SetFormat("d")).AsString

'Eingabe des Bereiches für Gefälle durch Eingabe
'der Indexnummer des Punkt-Themas
for each aGf in 0..IdxAnzGf
   aRichtNr=(aGf+1).SetFormat("").SetFormat("d")
   AnfStr=MsgBox.Input("Indexnummer des Punktes des Punkt-Themas"
       ++ThStr+NL+
        "am Anfang der"++aRichtNr.AsString+". Richtung",
        "Eingabe des Bereiches für das Gefälle"++GfTheme.AsString, AnfK)
   LofIdAnfNr.Add(AnfStr.AsNumber)
   EndStr=MsgBox.Input("Indexnummer des Punktes des Punkt-Themas"
       ++ThStr+NL+
        "am Ende der"++aRichtNr.AsString+". Richtung",
        "Eingabe des Bereiches für das Gefälle"++GfTheme.AsString, EndK)
   LofIdEndNr.Add(EndStr.AsNumber)
   AnfKNr=(EndStr.AsNumber+1).SetFormat("").SetFormat("d")
   AnfK=AnfKNr.AsString
end
for each aKGf in 0..IdxAnzGf
   aRichtNr=(aKGf+1).SetFormat("").SetFormat("d")
   AnfNrKont=LofIdAnfNr.Get(aKGf).SetFormat("").SetFormat("d")
   EndNrKont=LofIdEndNr.Get(aKGf).SetFormat("").SetFormat("d")
   MsgBox.Report("Das Gefälle-Thema:"++GfTheme.AsString++NL+
         "Das Punkt-Thema:"++ThStr++NL+
         "Indexnummer am Anfang der"++aRichtNr.AsString+". Richtung:  "
        ++AnfNrKont.AsString+NL+
         "Indexnummer am Ende der"++aRichtNr.AsString+". Richtung:  "
        ++EndNrKont.AsString, "Kontrolle der Eingabedaten")
end
Kontr1=MsgBox.YesNo("Sind alle Indexnummer richtig eingegeben?",
     "Kontrolle der Eingabe", True)
if (not Kontr1) then
  MsgBox.Error("Zunächst wird das Programm unterbrochen!"+NL+
            "Die Indexnummer sind beim Punkt-Thema"
            ++ThStr++"festzustellen."+NL+
            "Dann ist das Programm neu zu starten.", "")
  Exit
end
'Berechnung der projizierten Punkte auf Gefälle im Karten-View
Listofx={}
Listofy={}
Listofxprj={}
Listofyprj={}
ListofPtprj = {}
for each aGf in 0..IdxAnzGf
   GfPt00 = ListofVt.Get(aGf)
   GfPtx00 = GfPt00.Getx
   GfPty00 = GfPt00.Gety

   aGf1 = aGf+1
   GfPt11 = ListofVt.Get(aGf1)
   GfPtx11 = GfPt11.Getx
   GfPty11 = GfPt11.Gety
   if (GfPtx00 <> GfPtx11) then
     if (GfPtx00 > GfPtx11) then
```

```
        x2krd=GfPtx00
        x1krd=GfPtx11
        y2krd=GfPty00
        y1krd=GfPty11
      elseif (GfPtx00 < GfPtx11) then
        x2krd=GfPtx11
        x1krd=GfPtx00
        y2krd=GfPty11
        y1krd=GfPty00
      end
      NeigGf=(y2krd-y1krd)/(x2krd-x1krd)
      'MsgBox.Info(NeigGf.AsString,"Neigung des"
      '++aGf1.AsString+". Gefälles"++GfTheme.AsString)
      BGf=y2krd-(NeigGf*x2krd)
    end
    IdAnf=LofIdAnfNr.Get(aGf)
    IdEnd=LofIdEndNr.Get(aGf)
    for each aPt in 0..IdxPt
      IdaPt = PTFTab.ReturnValue(PtIdFld, aPt)
      if (((IdaPt > IdAnf) or (IdaPt = IdAnf)) and
        ((IdaPt < IdEnd) or (IdaPt = IdEnd))) then
        Pt3krd = PTFTab.ReturnValue(PtShpFld, aPt)
        x3krd = Pt3krd.Getx
        y3krd = Pt3krd.Gety
        if (GfPtx00 <> GfPtx11) then
          'Berechnung der Linie
          NeigPT=(-1/NeigGf)
          BPT=y3krd-(NeigPT*x3krd)
          'Berechnung der projizierten Punkte
          xprjkrd=(BPT-BGf)/(NeigGf-NeigPT)
          yprjkrd=((BGf*NeigPT)-(BPT*NeigGf))/(NeigPT-NeigGf)
        elseif (GfPtx00 = GfPtx11) then
          xprjkrd=GfPtx00
          yprjkrd=y3krd
        end
        Listofx.Add(x3krd)
        Listofy.Add(y3krd)
        Listofxprj.Add(xprjkrd)
        Listofyprj.Add(yprjkrd)
        ListofPtprj.Add(xprjkrd@yprjkrd)
      end
    end
end

av.ShowMsg("Speicherung der projizierten Punkte ...")

'Ein Feature-Shape-File für einen auf das Gefälle projizierten
'Profilschnitt (Punkt) wird im Profilschnitt-View hergestellt.
WDStr = theProject.GetWorkDir.AsString
fn00 = ("Prj"+(ThStr.LCase)).Left(6)
fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
fName=FileDialog.Put(fnStr, "*.shp",
    "Output shape File (Punkte für Profilsschnitt)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PrjFTab = FTab.MakeNew(fName, Point)
PrjShpFld = PrjFTab.FindField("shape")

AnzPtFld = ListofPtFlds.Count
IdxPtFld = AnzPtFld - 1
```

```
Listof2DFld = {}
Listof2DprjFld = {}

'Feststellung der Felder und des ersten Datensatzes
'Herstellung der Felder für das neue Thema durch Clone
FldStr=""
DtStr=""

for each i in 0..IdxPtFld
  aPtFld = ListofPtFlds.Get(i)
  aPtFldStr = aPtFld.GetName
  if (aPtFldStr <> "Shape") then
    FldStr = FldStr + aPtFldStr+"; "
    aValue = PtFTab.ReturnValue(aPtFld, 0)
    DtStr = DtStr + aValue.AsString+"; "
    aNPtFld = aPtFld.Clone
    Listof2DFld.Add(aPtFld)
    Listof2DprjFld.Add(aNPtFld)
  end
end

MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
           +NL+FldStr+NL+NL+
           "Der erste Datensatz:"+NL+DtStr,
           "Information")

Fr11 = MsgBox.YesNo("Gibt es ein Feld für"++
      "projizierte Koordinaten im Punkt-Thema:"++ThStr+"?",
      "Feststellung des Feldes", false)
if (Fr11) then
  RWPrjFld = MsgBox.ListAsString(Listof2DprjFld,
           "für RW der auf Gefälle projizierten Punkte",
           "Auswahl eines Feldes im Thema"++ThStr)
  HWPrjFld = MsgBox.ListAsString(Listof2DprjFld,
          "für HW der auf Gefälle projizierten Punkte",
          "Auswahl eines Feldes im Thema"++ThStr)
elseif (Not Fr11) then
  RWPrjFld = Field.Make("RWprj", #Field_Float, 10, 2)
  HWPrjFld = Field.Make("HWprj", #Field_Float, 10, 2)
  Listof2DprjFld.Add(RWPrjFld)
  Listof2DprjFld.Add(HWPrjFld)
end

PrjFTab.AddFields(Listof2DprjFld)

Anz2DFld = Listof2DFld.Count
Idx2DFld = Anz2DFld - 1

PrjFTab.SetEditable(false)
PrjFTab.SetEditable(true)

for each j in 0..IdxPt
   aPtprjShp = ListofPtprj.Get(j)
   PrjFTab.AddRecord
   PrjFTab.SetValue(PrjShpFld, j, aPtprjShp)
   for each i in 0..Idx2DFld
     aFld = Listof2DFld.Get(i)
     aNFld = Listof2DprjFld.Get(i)
     aValue = PtFTab.ReturnValue(aFld, j)
     PrjFTab.SetValue(aNFld, j, aValue)
   end
   aPtprjRW = aPtprjShp.Getx
```

```
    aPtprjHW = aPtprjShp.Gety
    PrjFTab.SetValue(RWPrjFld, j, aPtprjRW)
    PrjFTab.SetValue(HWPrjFld, j, aPtprjHW)
end

PrjFTab.SetEditable(false)

PrjTheme = FTheme.Make(PrjFTab)
theView.AddTheme(PrjTheme)
PrjTheme.UpdateLegend




'ptprjpft.ave
'Punkte auf einer Profilschnittlinie, z.B. MT, werden auf eine
'Profilschnittlinie (Punkte), z.B. NT, senkrecht projiziert,
'um die MT zum Vergleich mit NT zu zeichnen.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc              'Aktives Karten-View

'Eingabe der Datei, die Punktdaten, z.B. der MT enthält,
'um sie auf eine Profilschnittlinie (Punkte), z.B. NT, zu projizieren.
ListofThms=theView.GetThemes
ListofFThm = {}
for each aT in ListofThms
    if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofFThm.Add(aT)
      end
    end
end

MTThm = MsgBox.ChoiceAsString(ListofFThm,
      "das die Punkte auf einer Profilschnittlinie,"
      ++"z.B. MT, zur Projektion enthält",
      "Auswahl eines Themas im View"++theView.AsString)

MTFTab = MTThm.GetFTab
ListofMTFlds = MTFTab.GetFields

MTShpFld = ListofMTFlds.Get(0)

'Feststellung der Anzahl der Punkte in dem Thema MTThm
AnzMTPt=0
for each rec in MTFTab
    AnzMTPt=AnzMTPt+1
end
IdxMTPt=AnzMTPt-1
'MsgBox.Info(AnzMTPt.AsString, "Anzahl der Punkte in dem Thema"
'       ++MTThm.AsString)

'Eingabe der Datei, die Punkte, z.B. von NT,
'auf einer beliebigen Profillinie enthält.

NTThm = MsgBox.ChoiceAsString(ListofFThm,
      "das Punkte, z.B. von NT,"+NL+
      "auf einer Profilschnittlinie enthält","Eingabe eines Themas:")
```

```
NTFTab=NTThm.GetFTab
ListofNTFlds=NTFTab.GetFields

NTShpFld=NTFTab.FindField("Shape")
NTEfFld=MsgBox.ChoiceAsString(ListofNTFlds,
       "Eingabe des Feldes für Entfernungen vom Anfang",
       "Auswahl eines Feldes im Thema:"++NTThm.AsString)

'Feststellung der Anzahl der Punkte in dem Thema NTThm
AnzNTPt=0
for each rec in NTFTab
   AnzNTPt=AnzNTPt+1
end
IdxNTPt=AnzNTPt-1
'MsgBox.Info(AnzNTPt.AsString,
'     "Anzahl der Punkte in dem Thema"++NTThm.AsString)

av.ShowMsg("Bestimmung der Lage der Punkte, z.B. von MT,"
        ++"auf einer Profilschnittlinie, z.B. von NT...")
av.ShowStopButton

ListofNEntf={}
ListofPtprj = {}
ListofMTIdx = {}
for each j in 0..IdxMTPt
  Ng=j+1
  theMTPt = MTFTab.ReturnValue(MTShpFld, j)

  MTPtX = theMTPt.Getx
  MTPtY = theMTPt.Gety
  minAbstA = 1000000

  'Suche nach dem Punkt im Punkt-Thema, z.B. von NT,
  'mit dem kleinsten Abstand
  for each i in 0..IdxNTPt
     theNTPt=NTFTab.ReturnValue(NTShpFld, i)
     NTPtX=theNTPt.Getx
     NTPtY=theNTPt.Gety
     DifX2=((NTPtX-MTPtX)*(NTPtX-MTPtX))
     DifY2=((NTPtY-MTPtY)*(NTPtY-MTPtY))
     AbstA=(DifX2+DifY2).Sqrt.Abs
     if (AbstA < minAbstA) then
       minAbstA=AbstA
       aIdxNTPt=i
     end
  end

  'Berechnung der Entfernung der Punkte, z.B. von MT,
  'die auf der Profilschnittlinie, z.B. von NT, projiziert sind,
  'vom Anfang der Profilschnittlinie, z.B. von NT

  if ((aIdxNTPt > 0) and (aIdxNTPt < IdxNTPt)) then
    if (minAbstA <> 0) then
      VIdx=aIdxNTPt-1
      NIdx=aIdxNTPt+1

      PTV=NTFTab.ReturnValue(NTShpFld, VIdx)
      PTN=NTFTab.ReturnValue(NTShpFld, NIdx)
      PTI=NTFTab.ReturnValue(NTShpFld, aIdxNTPt)

      PTVX=PTV.Getx
      PTVY=PTV.Gety
```

```
  PTNX=PTN.Getx
  PTNY=PTN.Gety

  DifXV2=((PTVX-MTPtX)*(PTVX-MTPtX))
  DifYV2=((PTVY-MTPtY)*(PTVY-MTPtY))
  AbstV=(DifXV2+DifYV2).Sqrt.Abs

  DifXN2=((PTNX-MTPtX)*(PTNX-MTPtX))
  DifYN2=((PTNY-MTPtY)*(PTNY-MTPtY))
  AbstN=(DifXN2+DifYN2).Sqrt.Abs

  if (AbstV < AbstN) then
    VIdxPt=aIdxNTPt-1
    NIdxPt=aIdxNTPt
  elseif (AbstV > AbstN) then
    VIdxPt=aIdxNTPt
    NIdxPt=aIdxNTPt+1
  elseif (AbstV = AbstN) then
    VIdxPt=aIdxNTPt
    NIdxPt=aIdxNTPt
  end
  PTVA=NTFTab.ReturnValue(NTShpFld, VIdxPt)
  PTNA=NTFTab.ReturnValue(NTShpFld, NIdxPt)

  PTVAX=PTVA.Getx
  PTVAY=PTVA.Gety

  PTNAX=PTNA.Getx
  PTNAY=PTNA.Gety

  if (AbstV <> AbstN) then
    if (PTVAX <> PTNAX) then
      if (PTVAY <> PTNAY) then
        Neiga=(PTNAY-PTVAY)/(PTNAX-PTVAX)
        Achsb=PTVAY-(Neiga*PTVAX)
        Neigag=(-1)/Neiga
        Achsbg=MTPtY-(Neigag*MTPtX)
        BPtprjx=(Achsbg-Achsb)/(Neiga-Neigag)
        BPtprjy=(Neiga*BPtprjx)+Achsb
      elseif (PTVAY = PTNAY) then
        BPtprjx=MTPtX
        BPtprjy=PTVAY
      end
    elseif (PTVAX = PTNAX) then
      BPtprjx=PTVAX
      BPtprjy=MTPtY
    end
  elseif (AbstV = AbstN) then
    BPtprjx=PTVAX
    BPtprjy=PTVAY
  end
  DifXVA2=((PTVAX-BPtprjx)*(PTVAX-BPtprjx))
  DifYVA2=((PTVAY-BPtprjy)*(PTVAY-BPtprjy))

  AbstVA=(DifXVA2+DifYVA2).Sqrt.Abs
  EntfVA=NTFTab.ReturnValue(NTEfFld, VIdxPt)
  EntfPTA=EntfVA+AbstVA
elseif (minAbstA = 0) then
  BPtprjx=MTPtX
  BPtprjy=MTPtY
  EntfPTA=NTFTab.ReturnValue(NTEfFld, aIdxNTPt)
```

```
    end
    ListofNEntf.Add(EntfPTA)
    ListofPtprj.Add(BPtprjx@BPtprjy)
    ListofMTIdx.Add(j)
  end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzMTPt*100)
  if (not more) then
    break
  end
end

av.ShowMsg("Speicherung der projizierten Punkte,"
        ++"z.B. von MT, auf einer Profilschnittlinie ...")
av.ShowStopButton

'Ein Feature-Shape-File für einen auf eine Profilschnittlinie (Punkte)
'projizierten Profilschnitt (Punkt) wird im Karten-View hergestellt.
WDStr = theProject.GetWorkDir.AsString
fn00 = ("Prj"+(MTThm.AsString.LCase)).Left(6)
fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
fName=FileDialog.Put(fnStr, "*.shp",
      "Output shape File (Punkte für Profilsschnitt)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PrjFTab = FTab.MakeNew(fName, Point)
PrjShpFld = PrjFTab.FindField("shape")

AnzPtFld = ListofMTFlds.Count
IdxPtFld = AnzPtFld - 1

Listof2DFld = {}
Listof2DprjFld = {}

'Feststellung der Felder und des ersten Datensatzes
FldStr=""
DtStr=""

for each i in 0..IdxPtFld
  aPtFld = ListofMTFlds.Get(i)
  aPtFldStr = aPtFld.GetName
  if (aPtFldStr <> "Shape") then
    FldStr = FldStr + aPtFldStr+"; "
    aValue = MTFTab.ReturnValue(aPtFld, 0)
    DtStr = DtStr + aValue.AsString+"; "
    aNPtFld = aPtFld.Clone
    Listof2DFld.Add(aPtFld)
    Listof2DprjFld.Add(aNPtFld)
  end
end

MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
           +NL+FldStr+NL+NL+
           "Der erste Datensatz:"+NL+DtStr,
           "Information")

Fr11 = MsgBox.YesNo("Gibt es ein Feld für"++
      "projizierte Koordinaten im Punkt-Thema:"++MTThm.AsString+"?",
      "Feststellung des Feldes", false)
if (Fr11) then
  RWPrjFld = MsgBox.ListAsString(Listof2DprjFld,
```

```
            "für RW der auf einen Profilschnitt projizierten Punkte",
            "Auswahl eines Feldes im Thema"++MTThm.AsString)
  HWPrjFld = MsgBox.ListAsString(Listof2DprjFld,
            "für HW der auf einen Profilschnitt projizierten Punkte",
            "Auswahl eines Feldes im Thema"++MTThm.AsString)
elseif (Not Fr11) then
  RWPrjFld = Field.Make("RWprj", #Field_Float, 10, 2)
  HWPrjFld = Field.Make("HWprj", #Field_Float, 10, 2)
  Listof2DprjFld.Add(RWPrjFld)
  Listof2DprjFld.Add(HWPrjFld)
end


Fr22 = MsgBox.YesNo("Gibt es ein Feld für"++
      "Entfernungen der Punkt vom Anfang im Punkt-Thema:"
      ++MTThm.AsString+"?", "Feststellung des Feldes", false)
if (Fr22) then
  EntfFld = MsgBox.ListAsString(Listof2DprjFld,
           "für Entfernungen der Punkt vom Anfang im Punkt-Thema:"
          ++MTThm.AsString,
           "Auswahl eines Feldes im Thema"++MTThm.AsString)
elseif (Not Fr22) then
  EntfFld = Field.Make("Entfern_m", #Field_Float, 9, 2)
  Listof2DprjFld.Add(EntfFld)
end

PrjFTab.AddFields(Listof2DprjFld)

Anz2DFld = Listof2DFld.Count
Idx2DFld = Anz2DFld - 1

PrjFTab.SetEditable(false)
PrjFTab.SetEditable(true)

AnzNPt=ListofNEntf.Count
IdxNPt=AnzNPt-1

MsgBox.Report("Anzahl der Punkte der MT:"++AnzMTPt.AsString+NL+
            "Anzahl der Punkte der NT:"++AnzNTPt.AsString+NL+
            "Anzahl der neuen Punkte: "++AnzNPt.AsString,
            "Information")

for each j in 0..IdxNPt
   aIdx = ListofMTIdx.Get(j)
   aPtprjShp = ListofPtprj.Get(j)
   PrjFTab.AddRecord
   PrjFTab.SetValue(PrjShpFld, j, aPtprjShp)
   for each i in 0..Idx2DFld
     aFld = Listof2DFld.Get(i)
     aNFld = Listof2DprjFld.Get(i)
     aValue = MTFTab.ReturnValue(aFld, aIdx)
     PrjFTab.SetValue(aNFld, j, aValue)
   end
   aPtprjRW = aPtprjShp.Getx
   aPtprjHW = aPtprjShp.Gety
   PrjFTab.SetValue(RWPrjFld, j, aPtprjRW)
   PrjFTab.SetValue(HWPrjFld, j, aPtprjHW)
   aEntf = ListofNEntf.Get(j)
   PrjFTab.SetValue(EntfFld, j, aEntf)
end

PrjFTab.SetEditable(false)
```

```
PrjTheme = FTheme.Make(PrjFTab)
theView.AddTheme(PrjTheme)
PrjTheme.UpdateLegend




'ptprjprf.ave
'Punkte auf einem Profilschnitt werden auf eine Gefälls-Linie projiziert.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject = av.GetProject
PfView = av.GetActiveDoc         'Ein aktives Profilschnitt-View
myScript=theProject.FindScript("ptprjprf")
myScript.SetNumberFormat( "d.dd")   ' script default

thePtThm = PfView.GetActiveThemes.Get(0)
ThStr = thePtThm.AsString

kt00=MsgBox.YesNo("Ist das aktive Thema"++ThStr
     +NL+"zur Projektion der Punkte auf eine Gefälles-Linie richtig?",
     "Kontrolle des aktiven Themas", true)
if (Not kt00) then
  MsgBox.Error("Das aktive Thema"++ThStr++"ist falsch!"+NL+
            "Das aktive Thema ist neu auszuwählen!", "")
  exit
end

PtFTab = thePtThm.GetFTab          'Tabelle für das Punkt-Thema
ListofPtFlds = PtFTab.GetFields    'Liste der Überschrift der Tabelle

PtShpFld = ListofPtFlds.Get(0)

YFaktorStr = MsgBox.Input("zur Überhöhung der Höhen im Profil",
           "Eingabe eines Faktors", "50")
YFkt = YFaktorStr.AsNumber

AnzPt = 0
for each rec in PtFTab
   AnzPt = AnzPt + 1
end
IdxPt = AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"++ThStr)

'Eingabe eines Themas für ein Gefälle im aktiven Profilschnitt-View

ListofThms = PfView.GetThemes
ListofPLThms = {}

for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aClNm = aFTab.GetShapeClass.GetClassName
      if (aClNm = "PolyLine") then
        ListofPLThms.Add(aT)
      end
   end
end
```

```
GfTheme = MsgBox.ChoiceAsString(ListofPLThms,
      "das Gefälle-Linien enthält",
      "Auswahl eines Themas im View"++PfView.AsString)

GfFTab = GfTheme.GetFTab
ListofGfFld = GfFTab.GetFields

GfShpFld = GfFTab.FindField("Shape")

AnzGf = 0               'Anzahl der Datensätze
for each rec in GfFTab
  AnzGf = AnzGf + 1
end
IdxGf = AnzGf - 1

AnzGfFld = ListofGfFld.Count   'Anzahl der Felder
IdxGfFld = AnzGfFld - 1

'Die Bezeichnung der Felder wird in der Tabelle gesucht.

'Feststellung der Felder und des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxGfFld
   aFld = ListofGfFld.Get(i)
   aFldStr = aFld.GetName
   if (aFldStr <> "Shape") then
      FldStr = FldStr + aFldStr+"; "
      aValue = GfFTab.ReturnValue(aFld, 0)
      DtStr = DtStr + aValue.AsString+"; "
   end
end

MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
           +NL+FldStr+NL+NL+
           "Der erste Datensatz:"+NL+DtStr,
           "Information")

aKWFld = MsgBox.ListAsString(ListofGfFld,
       "um eine Gefälls-Linie auszuwählen",
       "Auswahl eines Feldes des Themas"++GfTheme.AsString)

ListofGfFL = {}
for each rec in 0..IdxGf
   aGfFL = GfFTab.ReturnValue(aKWFld, rec)
   ListofGfFL.Add(aGfFL)
end

'Auswahl eines Datensatzes, um den als Gefälle zu benutzen.
theGf = MsgBox.ChoiceAsString(ListofGfFL,
    "um den als Gefälle zu benutzen",
    "Auswahl eines Datensatzes im Thema:"++GfTheme.AsString)
GfIdx = ListofGfFL.FindByValue(theGf)
theShape = GfFTab.ReturnValue(GfShpFld, GfIdx)
theProfilList={}
theProfilList.Add({theShape})
ListofVt={}
'2D-PolyLine wird in Liste der Koordinaten umgewandelt.
if (theProfilList <> 0) then
  for each q in theProfilList
     aListofList=q.Get(0).AsList
     for each aList in aListofList
```

```
          for each apt in aList
             ListofVt.Add(apt)
          end
        end
      end
end
PfAnz=ListofVt.Count
PfIndAnz=PfAnz-1
MsgBox.Info(PfAnz.AsString,
     "Anzahl der Stützpunkte auf der Gefälles-Linie")

av.ShowMsg("Die Punkte werden auf die Gefälls-Linie"
         ++aKWFld.AsString++"projiziert ...")
av.ShowStopButton
ListofGfHPt={}
Ng=0
For each aRPt in 0..IdxPt
   Ng = Ng+1
   aPt = PtFTab.ReturnValue(PtShpFld, aRPt)
   theEntf = aPt.Getx  'Entfernung des Punktes vom Anfang
   theHFkt = aPt.Gety  'des Profilschnittes
   for each aGPt in 0..PfIndAnz
      aIdx = aGPt
      theGPtV = ListofVt.Get(aGPt)
      xkrd1 = theGPtV.Getx   'Entfernung vom Vertex des Gefälles
      ykrd1 = theGPtV.Gety
      if (aIdx < PfIndAnz) then
         theGPtN = ListofVt.Get(aGPt+1)
         xkrd2 = theGPtN.Getx
         ykrd2 = theGPtN.Gety
         if (theEntf = xkrd1) then
            theGfH = ykrd1
         elseif ((theEntf > xkrd1) and (theEntf < xkrd2)) then
            if (ykrd1 > ykrd2) then
               theGfH=ykrd2-(((ykrd2-ykrd1)/(xkrd2-xkrd1))
                     *(xkrd2-theEntf))
            elseif (ykrd1 < ykrd2) then
               theGfH=(((ykrd2-ykrd1)/(xkrd2-xkrd1))
                     *(theEntf-xkrd1))+ykrd1
            end
         end
      elseif (aIdx = PfIndAnz) then
         if (theEntf = xkrd1) then
            theGfH = ykrd1
         end
      end
   end
   ListofGfHPt.Add(theEntf@theGfH)
   'Show percentage complete with enabled stop button
   more=av.SetStatus((Ng/(AnzPt))*100)
   if (not more) then
      break
   end
end

'Speicherung der Daten im Profilschnitt-View
av.ShowMsg("Speicherung der Punkte in einem neuen Thema im View"
         ++PfView.AsString++"...")

'Ein Feature-Shape-File für einen auf das Gefälle projizierten
'Profilschnitt (Punkt) wird im Profilschnitt-View hergestellt.
WDStr = theProject.GetWorkDir.AsString
```

```
fn00 = ("Prj"+(ThStr.LCase)).Left(6)
fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
fName=FileDialog.Put(fnStr, "*.shp",
      "Output shape File (Punkte für Profilsschnitt)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PfFTab=FTab.MakeNew(fName, Point)
PfShpFld=PfFTab.FindField("shape")

AnzPtFld = ListofPtFlds.Count
IdxPtFld = AnzPtFld - 1

Listof2DFld = {}
Listof2DprjFld = {}
FldStr = ""
DtStr = ""

'Herstellung der Felder für das neue Thema durch Clone
for each i in 0..IdxPtFld
  aPtFld = ListofPtFlds.Get(i)
  aPtFldStr = aPtFld.GetName
  if (aPtFldStr <> "Shape") then
    FldStr = FldStr + aPtFldStr+"; "
    aValue = PtFTab.ReturnValue(aPtFld, 0)
    DtStr = DtStr + aValue.AsString+"; "
    aNPtFld = aPtFld.Clone
    Listof2DFld.Add(aPtFld)
    Listof2DprjFld.Add(aNPtFld)
  end
end

MsgBox.Report("Die Namen der Felder ohne Shape-Felder"
           +NL+FldStr+NL+NL+
           "Der erste Datensatz:"+NL+DtStr,
           "Information")

Fr11 = MsgBox.YesNo("Gibt es ein Feld für"++
      "projizierte Höhe im Punkt-Thema:"++ThStr+"?",
      "Feststellung des Feldes", false)
if (Fr11) then
  GfPrjHmFld = MsgBox.ListAsString(Listof2DprjFld,
            "für Höhe [m] der auf Gefälle projizierten Punkte",
            "Auswahl eines Feldes im Thema"++ThStr)
  GfPrjHFktFld = MsgBox.ListAsString(Listof2DprjFld,
  "für Höhe (Hoehe * Faktor) der auf Gefälle projizierten Punkte",
  "Auswahl eines Feldes im Thema"++ThStr)
elseif (Not Fr11) then
  GfPrjHmFld=Field.Make("GfprjHm", #Field_Float, 7, 2)
  GfPrjHFktFld=Field.Make("GfprjHFkt", #Field_Float, 9, 2)
  Listof2DprjFld.Add(GfPrjHmFld)
  Listof2DprjFld.Add(GfPrjHFktFld)
end

PfFTab.AddFields(Listof2DprjFld)

Anz2DFld = Listof2DFld.Count
Idx2DFld = Anz2DFld - 1

PfFTab.SetEditable(false)
PfFTab.SetEditable(true)
```

```
for each j in 0..IdxPt
   aPtprjShp = ListofGfHPt.Get(j)
   PfFTab.AddRecord
   PfFTab.SetValue(PfShpFld, j, aPtprjShp)
   for each i in 0..Idx2DFld
     aFld = Listof2DFld.Get(i)
     aNFld = Listof2DprjFld.Get(i)
     aValue = PtFTab.ReturnValue(aFld, j)
     PfFTab.SetValue(aNFld, j, aValue)
   end
   aPtprjyFkt = aPtprjShp.Gety
   aPtprjym = aPtprjyFkt/YFkt
   PfFTab.SetValue(GfPrjHmFld, j, aPtprjym)
   PfFTab.SetValue(GfPrjHFktFld, j, aPtprjyFkt)
end

PfFTab.SetEditable(false)

PfTheme = FTheme.Make(PfFTab)
PfView.AddTheme(PfTheme)
PfTheme.UpdateLegend




'ptqpf1ds.ave
'Alle Punkte in einem Datensatz der Querprofilschnitte im aktiven Karten-View
'werden in einem Punkt-Thema gesammelt. Wenn mehr als ca. 51
'Querprofilschnitt-Dateien bei der Sammelung der Punkte bei einer Sitzung
'des Projektes geöffnet werden, kann eine Fehlermeldung "No free Channels"
'entstehen. Dann wird das Programm abgebrochen. Die Ursache ist eine
'Einschränkung der Anzahl der Dateien, die vom Betiebsystem Windows
'insgesamt geöffnet werden können. Bei dem Fall soll man entweder
'die Anzahl der Dateien, die vom Betriebsystem geöffnet werden können,
'in der Konfigurationsdatei des Betirebsystems erhöhen, oder jedesmal
'das Projekt schließen und wieder öffnen, wenn die Punkte von
'ca. 50 Querprofilen gesammelt worden sind.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc     'ein aktives Querprofilschnitt-View
myScript=theProject.FindScript("ptqpf1ds")
myScript.SetNumberFormat( "d.dd") ' script default

YFtStr=MsgBox.Input("zur Überhöhung der Höhe",  "Eingabe eines Faktors", "50")
YFt=YFtStr.AsNumber

ListofViews={"kT1_gg", "Kt1_ggd", "Kt1_hg", "Kt1_hgd", "Kt1_tga", "Kt1_tgd"}
KViewStr=MsgBox.ChoiceAsString(ListofViews,
        "wo die Themen liegen oder entstehen,"+NL+
        "um die Punkte der Querprofilschnitte zu sammeln",
        "Name eines Karten-Views")

aNViewStr = MsgBox.Input("Ein View für eine Karten-Darstellung",
          "Veränderungsmöglichkeit", KViewStr)

theKView=av.FindDoc(aNViewStr)

ListofBst={"P5", "X5", "U5", "V5", "S5"}
myBst=MsgBox.ChoiceAsString(ListofBst,
    "im View"++theView.AsString+NL+
```

```
    "Die ersten zwei Buchstaben der Querprofilschnitte",
    "Auswahl der Querprofilschnitte")
my1B=myBst.Left(1)

ListofProfile=theView.GetThemes
AnzProfile=ListofProfile.Count
IdxProfile=AnzProfile-1
MinHw=5642000
MaxHw=5618000
AnzPTh=0
ListofPThemes={}

for each aPrf in 0..IdxProfile
   theTheme=ListofProfile.Get(aPrf)
   ThStr=theTheme.AsString
   B2=ThStr.Left(2)
   B1=ThStr.Left(1)
   if (B2 = myBst) then
      ListofThStr=ThStr.AsTokens(B1+".shp")
      ThStr2=ListofThStr.Get(0)
      if (ThStr2.IsNumber) then
         ListofPThemes.Add(theTheme)
         AnzPTh=AnzPTh+1
         ThNr2=ThStr2.AsNumber
         if (ThNr2 < MinHw) then
            MinHw=ThNr2
         end
         if (ThNr2 > MaxHw) then
            MaxHw=ThNr2
         end
      end
   end
end

MiHW=MinHw.SetFormat("").SetFormat("d")
MaHW=MaxHw.SetFormat("").SetFormat("d")
MsgBox.Report("Der kleinste HW des aktiven Views:"++MiHw.AsString+NL+
           "Der größte HW des aktiven Views:  "++MaHw.AsString,
           "Kontrolle der Querprofile im View"++theView.AsString)

'Die Punkte werden von Profilschnitten gesammelt und in das Punkt-Thema gespeichert.
AnzPrfTh=((MaxHw-MinHw)/50)+1
if (AnzPTh <> AnzPrfTh) then
  MsgBox.Error("Die Anzahl der Themen im View"++theView.AsString++"ist falsch!"+NL+
            "Es gibt Themen, die nicht dem View zugehören.", "")
  exit
end

MsgBox.Info(AnzPrfTh.AsString,
     "die Anzahl der Querprofilschnitte im View"++theView.AsString)

Frg1=MsgBox.YesNo("Soll die Anzahl der Querprofile kleiner werden?",
    "Kontrolle der Anzahl der Querprofilschnitte", FALSE)
if (Frg1) then
  MinHWStr=MsgBox.Input("Der kleinste HW der Querprofile",
       "Auswahl der Querprofile im View"++theView.AsString, MinHW.AsString)
  MaxHWStr=MsgBox.Input("Der größte HW der Querprofile",
       "Auswahl der Querprofile im View"++theView.AsString, MinHW.AsString)
  MinHW=MinHWStr.AsNumber
  MaxHW=MaxHWStr.AsNumber
  AnzPrfTh=((MaxHw-MinHw)/50)+1
end
```

```
MsgBox.Info(AnzPrfTh.AsString,
      "die Anzahl der Querprofilschnitte im View"++theView.AsString)

IdxPrfTh=AnzPrfTh-1

EinThm=ListofPThemes.Get(0)
EinFTab=EinThm.GetFTab
ListofEThmFld=EinFTab.GetFields
myFLFld=MsgBox.ChoiceAsString(ListofEThmFld,
       "Das Feld für Flächennamen im Thema:"++EinThm.AsString,
       "Auswahl eines Feldes in Querprofilen")
myFLFldStr=myFLFld.AsString

ListofFlnamen={}

for each record in EinFTab
   for each afld in EinFTab.GetFields
      aValue = EinFTab.ReturnValue(afld, record)
      afldnmStr = afld.GetName
      if (afldnmStr <> "Shape") then
        ListofFlnamen.Add(aValue)
      end
   end
end

'Auswahl einer Flaeche im Querprofilschnitt
ListofSchichten={"Geländehöhe", "Terrassenhöhe",
              "Quartärbasishöhe", "NT abgedeckte Höhe"}
theScht=MsgBox.ChoiceAsString(ListofSchichten,
      "in Querprofilschnitt-Themen"+NL+
      "Der Name der Flaeche",
      "Auswahl einer Fläche")
if (theScht = "Geländehöhe") then
  theFl = MsgBox.ChoiceAsString(ListofFlnamen,
         "für die Querprofilschnitte"+NL+
         "Der Name der Fläche für Geländehöhe",
         "Auswahl einer Fläche")
elseif (theScht = "Terrassenhöhe") then
  theFl = MsgBox.ChoiceAsString(ListofFlnamen,
         "für die Querprofilschnitte"+NL+
         "Der Name der Fläche für Terrassenoberkante",
         "Auswahl einer Fläche")
elseif (theScht = "Quartärbasishöhe") then
  theFl = MsgBox.ChoiceAsString(ListofFlnamen,
         "für die Querprofilschnitte"+NL+
         "Der Name der Fläche für Quartärbasis",
         "Auswahl einer Fläche")
elseif (theScht = "NT abgedeckte Höhe") then
  theFl = MsgBox.ChoiceAsString(ListofFlnamen,
         "für die Querprofilschnitte"+NL+
         "Der Name der Fläche für NT abgd. Höhe",
         "Auswahl einer Fläche")
end


ListofSp11 = {"Herstellung als neue Themen",
           "Speicherung in vorhandenen Themen"}

AW11 = MsgBox.ListAsString(ListofSp11,
     "zur Speicherung der gesammelten Punkte",
     "Auswahl der Themen")
```

```
if (AW11 = "Herstellung als neue Themen") then
  WDStr=av.GetProject.GetWorkDir.AsString

  'ein neues Thema für die geologische Fläche
  fn00 = "Pt"+theScht+"fl"
  fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
  'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
  if (fName =nil) then exit end
  fName.SetExtension("shp")
  PTFTab = FTab.MakeNew(fName, Point)

  PTShpFld = PTFTab.FindField("shape")
  PTIdFld = Field.Make("ID", #Field_LONG, 7, 0)
  PTRWFld = Field.Make("RW", #Field_Float, 10, 2)
  PTHWFld = Field.Make("HW", #Field_Float, 10, 2)
  PTZWFld = Field.Make("Hoehe50", #Field_Float, 9, 2)
  PTZMFld = Field.Make("Hoehem", #Field_Float, 7, 2)

  ListofNFlds={PTIdFld, PTRWFld, PTHWFld, PTZWFld, PTZMFld}
  PTFTab.AddFields(ListofNFlds)
  thePTTheme = FTheme.Make(PTFTab)
  theKView.AddTheme(thePTTheme)

  recNr = -1

elseif (AW11 = "Speicherung in vorhandenen Themen") then

  ListofKThemes = theKView.GetThemes
  ListofPtThms = {}

  for each aT in ListofKThemes
    if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofPtThms.Add(aT)
      end
    end
  end

  'ein Thema für die geologische Fläche

  thePTTheme = MsgBox.ChoiceAsString(ListofPtThms,
        "um"++theScht++"zu speichern"+NL+
        "Der Name eines Punktthemas in View"++theKView.AsString,
        "Auswahl eines Themas")

  PTFTab=thePTTheme.GetFTab
  PTShpFld=PTFTab.FindField("Shape")
  PTIdFld=PTFTab.FindField("ID")
  PTRWFld=PTFTab.FindField("RW")
  PTHWFld=PTFTab.FindField("HW")
  PTZWFld=PTFTab.FindField("Hoehe50")
  PTZMFld=PTFTab.FindField("Hoehem")

  'Feststellung der Anzahl der in dem Punkt-Thema schon vorhandenen Daten
  Anz3DPT=0
  for each rec in PTFTab
    Anz3DPT=Anz3DPT+1
  end
  Anz3DPTIdx=Anz3DPT-1
  recNr = Anz3DPTIdx
```

```
  MsgBox.Info(Anz3DPT.AsString, "die Anzahl der Datensätze im Thema"
        ++thePTTheme.AsString)
end


av.ShowMsg("Die Daten der Profilschnitte werden in ein Punkt-Thema für"
      ++theScht++thePTTheme.AsString++"gespeichert.")
av.ShowStopButton

PTFTab.SetEditable(False)
PTFTab.SetEditable(true)

Ng=0
EntfernNr=50

for each aNr in 0..IdxPrfTh
   Ng=Ng+1
   ThNr2N=(MinHw+(aNr*EntfernNr))
   ThStr2N=(ThNr2N.SetFormat("").SetFormat("d")).AsString
   PThStr2N=my1B+ThStr2N+".shp"
   NThTheme=theView.FindTheme(PThStr2N)

   FTab1=NThTheme.GetFTab
   ShpFld1=FTab1.FindField("Shape")
   HWFld1=FTab1.FindField("HW")
   FLFld1=FTab1.FindField(myFLFldStr)
   'MsgBox.Info(NThTheme.AsString, "Der Name des Profils")

   'Feststellung der Anzahl der 2D-PolyLine in dem Profilschnitt-Thema
   Anz2DPL=0
   for each rec in FTab1
      Anz2DPL=Anz2DPL+1
   end
   Anz2DPLIdx=Anz2DPL-1

  for each i in 0..Anz2DPLIdx
    theFlaeche=FTab1.ReturnValue(FLFld1, i)
    if (theFl = theFlaeche) then
      theShape1=FTab1.ReturnValue(ShpFld1, i)
      theProfilList1={}
      theProfilList1.Add({theShape1})
      theHW=FTab1.ReturnValue(HWFld1, i)
      if (theHW <> ThNr2N) then
        MsgBox.Error("Der Hochwert des Themas"++NThTheme.AsString
                ++"ist falsch!"+NL+
                "Die Daten müssen kontrolliert werden!", "")
        exit
      end

      '2D-PolyLine im Profilschnitt wird im Punkt-Thema gespeichert

      if (theProfilList1 <> 0) then
        for each q in theProfilList1
          theLines=q.Get(0).AsList
          for each m in theLines
            for each ptx in m
              myx=ptx.Getx
              myy=ptx.Gety
              RWkrd=myx
              HWkrd=ThNr2N
              ZWkrd=myy
              ZMkrd=myy/ YFt
```

```
                a2DP=Point.Make(RWkrd, HWkrd)
                recNr=recNr+1
                PTFTab.AddRecord
                PTFTab.SetValue(PTShpFld, recNr, a2DP)
                PTFTab.SetValue(PTIdFld, recNr, recNr)
                PTFTab.SetValue(PTRWFld, recNr, RWkrd)
                PTFTab.SetValue(PTHWFld, recNr, HWkrd)
                PTFTab.SetValue(PTZWFld, recNr, ZWkrd)
                PTFTab.SetValue(PTZMFld, recNr, ZMkrd)
              end
            end
          end
        end
      end
    end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzPrfTh*100)
  if (not more) then
    break
  end
end

PTFTab.SetEditable(False)
thePTTheme.UpdateLegend
```

```
'ptqpf4ds.ave
'Alle Punkte in 4 Datensätzen der Querprofilschnitte im aktiven Karten-View
'werden in einem Punkt-Thema gesammelt. Wenn mehr als ca. 51
'Querprofilschnitt-Dateien bei der Sammelung der Punkte bei einer Sitzung
'des Projektes geöffnet werden, kann eine Fehlermeldung "No free Channels"
'entstehen. Dann wird das Programm abgebrochen. Die Ursache ist eine
'Einschränkung der Anzahl der Dateien, die vom Betiebsystem Windows
'insgesamt geöffnet werden können. Bei dem Fall soll man entweder
'die Anzahl der Dateien, die vom Betriebsystem geöffnet werden können,
'in der Konfigurationsdatei des Betirebsystems erhöhen, oder jedesmal
'das Projekt schließen und wieder öffnen, wenn die Punkte von
'ca. 50 Querprofilen gesammelt worden sind.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc     'ein aktives Querprofilschnitt-View
myScript=theProject.FindScript("ptqpf4ds")
myScript.SetNumberFormat( "d.dd") ' script default
YFtStr=MsgBox.Input("zur Überhöhung der Höhe",  "Eingabe eines Faktors", "50")
YFt=YFtStr.AsNumber

ListofViews={"kT1_gg", "Kt1_ggd", "Kt1_hg", "Kt1_hgd", "Kt1_tga", "Kt1_tgd"}
KViewStr=MsgBox.ChoiceAsString(ListofViews,
       "wo die Themen liegen oder entstehen,"+NL+
       "um die Punkte der Querprofilschnitte zu sammeln",
       "Name eines Karten-Views")

aNViewStr = MsgBox.Input("Ein View für eine Karten-Darstellung",
         "Veränderungsmöglichkeit", KViewStr)

theKView=av.FindDoc(aNViewStr)

ListofThemes=theView.GetThemes
```

```
AnzThemes=ListofThemes.Count
IdxThemes=AnzThemes-1
MinHw=5642000
MaxHw=5618000
ListofPThemes={}

for each aTh in 0..IdxThemes
   theTheme=ListofThemes.Get(aTh)
   ThStr=theTheme.AsString
   B2=ThStr.Left(2)
   B1=ThStr.Left(1)
   if (B2 = "P5") then
     ListofThStr=ThStr.AsTokens(B1+".shp")
     ThStr2=ListofThStr.Get(0)
     if (ThStr2.IsNumber) then
       ListofPThemes.Add(theTheme)
       ThNr2=ThStr2.AsNumber
       if (ThNr2 < MinHw) then
         MinHw=ThNr2
       end
       if (ThNr2 > MaxHw) then
         MaxHw=ThNr2
       end
     end
   end
end
MiHW=MinHw.SetFormat("").SetFormat("d")
MaHW=MaxHw.SetFormat("").SetFormat("d")
MsgBox.Report("Der kleinste HW des aktiven Views:"++MiHw.AsString+NL+
           "Der größte HW des aktiven Views:  "++MaHw.AsString,
           "Kontrolle der Querprofile im View"++theView.AsString)
AnzPThms=ListofPThemes.Count

AnzPThemes=((MaxHw-MinHw)/50)+1
if (AnzPThms <> AnzPThemes) then
  MsgBox.Error("Die Anzahl der Themen im View"++theView.AsString++"ist falsch!"+NL+
            "Es gibt Themen, die nicht dem View zugehören.", "")
  exit
end

MsgBox.Info(AnzPThemes.AsString,
     "die Anzahl der Querprofilschnitte im View"++theView.AsString)
Frg1=MsgBox.YesNo("Soll die Anzahl der Querprofilschnitte kleiner werden?",
    "Kontrolle der Anzahl der Querprofilschnitte", FALSE)
if (Frg1) then
  MinHWStr=MsgBox.Input("Der kleinste HW der Querprofile",
       "Auswahl der Querprofile im View"++theView.AsString, MinHW.AsString)
  MaxHWStr=MsgBox.Input("Der größte HW der Querprofile",
       "Auswahl der Querprofile im View"++theView.AsString, MinHW.AsString)
  MinHW=MinHWStr.AsNumber
  MaxHW=MaxHWStr.AsNumber
  AnzPThemes=((MaxHw-MinHw)/50)+1
end
MsgBox.Info(AnzPThemes.AsString,
  "die Anzahl der Querprofile im View"++theView.AsString)

IdxPThemes=AnzPThemes-1
EntfernNr=50

EinThm=ListofPThemes.Get(0)
EinFTab=EinThm.GetFTab
ListofEThmFld=EinFTab.GetFields
```

```
myFLFld=MsgBox.ChoiceAsString(ListofEThmFld,
       "Das Feld für Flächennamen im Thema:"++EinThm.AsString,
       "Auswahl eines Feldes in Querprofilen")
myFLFldStr=myFLFld.AsString
ListofFlnamen={}

for each record in EinFTab
   for each afld in EinFTab.GetFields
      aValue = EinFTab.ReturnValue(afld, record)
      afldnmStr = afld.GetName
      if (afldnmStr <> "Shape") then
        ListofFlnamen.Add(aValue)
      end
   end
end

GF=MsgBox.ChoiceAsString(ListofFlnamen,
  "Der Name der Fläche für Geländehöhe",
  "Auswahl einer Fläche für die Querprofile")
TF=MsgBox.ChoiceAsString(ListofFlnamen,
  "Der Name der Fläche für Terrassenoberkante",
  "Auswahl einer Fläche für die Querprofile")
AF=MsgBox.ChoiceAsString(ListofFlnamen,
  "Der Name der Fläche für NT abgd. Höhe",
  "Auswahl einer Fläche für die Querprofile")
QF=MsgBox.ChoiceAsString(ListofFlnamen,
  "Der Name der Fläche für Quartärbasis",
  "Auswahl einer Fläche für die Querprofile")

av.ShowMsg("Die Koordinaten der Profilschnitte werden in Liste gespeichert.")
av.ShowStopButton

ListofG3DP={}
ListofT3DP={}
ListofQ3DP={}
ListofA3DP={}
Ng=0

for each aNr in 0..IdxPThemes
   Ng=Ng+1
   ThNr2N=(MinHw+(aNr*EntfernNr))
   ThStr2N=(ThNr2N.SetFormat("").SetFormat("d")).AsString
   PThStr2N="P"+ThStr2N+".shp"
   NThTheme=theView.FindTheme(PThStr2N)
   FTab1=NThTheme.GetFTab
   ShpFld1=FTab1.FindField("Shape")
   HWFld1=FTab1.FindField("HW")
   FLFld1=FTab1.FindField(myFLFldStr)
   Listof2DFld1={ShpFld1, HWFld1, FLFld1}
   'MsgBox.Info(NThTheme.AsString, "Der Name des Profils")

   'Feststellung der Anzahl der 2D-PolyLine in dem Thema
   Anz2DPL=0
   for each rec in FTab1
      Anz2DPL=Anz2DPL+1
   end
   Anz2DPLIdx=Anz2DPL-1

  for each i in 0..Anz2DPLIdx
    theShape1=FTab1.ReturnValue(ShpFld1, i)
    theProfilList1={}
    theProfilList1.Add({theShape1})
```

```
      theHW=FTab1.ReturnValue(HWFld1, i)
      if (theHW <> ThNr2N) then
        MsgBox.Error("Der Hochwert des Themas"++NThTheme.AsString
                  ++"ist falsch!"+NL+
                 "Die Daten müssen kontrolliert werden!", "")
        exit
      end
      theFlaeche=FTab1.ReturnValue(FLFld1, i)
      '2D-PolyLine wird in Liste der 3D-Punkte umgewandelt.
      if (theProfilList1 <> 0) then
        for each q in theProfilList1
          theLines=q.Get(0).AsList
          for each m in theLines
            for each ptx in m
              myx=ptx.Getx
              myy=ptx.Gety
              if (theFlaeche=GF) then
                ListofG3DP.Add(myx@ThNr2N@myy)
              elseif (theFlaeche=TF) then
                ListofT3DP.Add(myx@ThNr2N@myy)
              elseif (theFlaeche=QF) then
                ListofQ3DP.Add(myx@ThNr2N@myy)
              elseif (theFlaeche=AF) then
                ListofA3DP.Add(myx@ThNr2N@myy)
'             elseif
'               MsgBox.Error("Die Flaeche des Themas"
'                         ++NThTheme.AsString++
'               "ist falsch!"+NL+
'               "Die Daten müssen kontrolliert werden!", "")
'               exit
              end
            end
          end
        end
      end
    end
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzPThemes*100)
    if (not more) then
      break
    end
  end
  'MsgBox.ListAsString(ListofG3DP,"ListofG3DP","Liste der Punkte")
  'MsgBox.ListAsString(ListofT3DP,"ListofT3DP","Liste der Punkte")
  'MsgBox.ListAsString(ListofQ3DP,"ListofQ3DP","Liste der Punkte")
  'MsgBox.ListAsString(ListofA3DP,"ListofA3DP","Liste der Punkte")

  ListofSp11 = {"Herstellung als neue Themen",
              "Speicherung in vorhandenen Themen"}

  AW11 = MsgBox.ListAsString(ListofSp11,
      "zur Speicherung der gesammelten Punkte",
      "Auswahl der Themen")

  if (AW11 = "Herstellung als neue Themen") then
    WDStr=av.GetProject.GetWorkDir.AsString

    'ein neues Thema für Geländehöhe
    fn00 = "Ptgfl00"
    fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
    'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
    fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
```

```
if (fName =nil) then exit end
fName.SetExtension("shp")
GPTFTab = FTab.MakeNew(fName, Point)

GPTShpFld = GPTFTab.FindField("shape")
GPTIdFld = Field.Make("ID", #Field_LONG, 7, 0)
GPTRWFld = Field.Make("RW", #Field_Float, 10, 2)
GPTHWFld = Field.Make("HW", #Field_Float, 10, 2)
GPTZWFld = Field.Make("Hoehe50", #Field_Float, 9, 2)
GPTZMFld = Field.Make("Hoehem", #Field_Float, 7, 2)

ListofNGFlds={GPTIdFld, GPTRWFld, GPTHWFld, GPTZWFld, GPTZMFld}
GPTFTab.AddFields(ListofNGFlds)
theGTheme = FTheme.Make(GPTFTab)
theKView.AddTheme(theGTheme)

'ein neues Thema für Terrassenhöhe (TOK)
fn00 = "Pttfl00"
fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName =nil) then exit end
fName.SetExtension("shp")
TPTFTab = FTab.MakeNew(fName, Point)

TPTShpFld = TPTFTab.FindField("shape")
TPTIdFld = Field.Make("ID", #Field_LONG, 7, 0)
TPTRWFld = Field.Make("RW", #Field_Float, 10, 2)
TPTHWFld = Field.Make("HW", #Field_Float, 10, 2)
TPTZWFld = Field.Make("Hoehe50", #Field_Float, 9, 2)
TPTZMFld = Field.Make("Hoehem", #Field_Float, 7, 2)

ListofNTFlds={TPTIdFld, TPTRWFld, TPTHWFld, TPTZWFld, TPTZMFld}
TPTFTab.AddFields(ListofNTFlds)
theTTheme = FTheme.Make(TPTFTab)
theKView.AddTheme(theTTheme)

'ein neues Thema für Quartärbasis
fn00 = "Ptqfl00"
fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName =nil) then exit end
fName.SetExtension("shp")
QPTFTab = FTab.MakeNew(fName, Point)

QPTShpFld = QPTFTab.FindField("shape")
QPTIdFld = Field.Make("ID", #Field_LONG, 7, 0)
QPTRWFld = Field.Make("RW", #Field_Float, 10, 2)
QPTHWFld = Field.Make("HW", #Field_Float, 10, 2)
QPTZWFld = Field.Make("Hoehe50", #Field_Float, 9, 2)
QPTZMFld = Field.Make("Hoehem", #Field_Float, 7, 2)

ListofNQFlds={QPTIdFld, QPTRWFld, QPTHWFld, QPTZWFld, QPTZMFld}
QPTFTab.AddFields(ListofNQFlds)
theQTheme = FTheme.Make(QPTFTab)
theKView.AddTheme(theQTheme)

'ein neues Thema für NT abgedeckte Höhe
fn00 = "Ptafl00"
fnStr=FileName.Make(WDStr).MakeTmp(fn00,"shp")
'fnStr=FileName.Make("C:\Verz1\Verz2\Verz3\").MakeTmp(fn00,"shp")
```

```
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
  if (fName =nil) then exit end
  fName.SetExtension("shp")
  APTFTab = FTab.MakeNew(fName, Point)

  APTShpFld = APTFTab.FindField("shape")
  APTIdFld = Field.Make("ID", #Field_LONG, 7, 0)
  APTRWFld = Field.Make("RW", #Field_Float, 10, 2)
  APTHWFld = Field.Make("HW", #Field_Float, 10, 2)
  APTZWFld = Field.Make("Hoehe50", #Field_Float, 9, 2)
  APTZMFld = Field.Make("Hoehem", #Field_Float, 7, 2)

  ListofNAFlds={APTIdFld, APTRWFld, APTHWFld, APTZWFld, APTZMFld}
  APTFTab.AddFields(ListofNAFlds)
  theATheme = FTheme.Make(APTFTab)
  theKView.AddTheme(theATheme)

  GrecNr = -1
  TrecNr = -1
  QrecNr = -1
  ArecNr = -1

elseif (AW11 = "Speicherung in vorhandenen Themen") then

  ListofKThemes = theKView.GetThemes
  ListofPtThms = {}

  for each aT in ListofKThemes
    if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofPtThms.Add(aT)
      end
    end
  end

  'ein Thema für Geländehöhe

  theGTheme=MsgBox.ChoiceAsString(ListofPtThms,
        "für Geländehöhe"+NL+
        "Der Name eines Punktthemas in View"++theKView.AsString,
        "Auswahl eines Themas")
  GPTFTab=theGTheme.GetFTab
  GPTShpFld=GPTFTab.FindField("Shape")
  GPTIdFld=GPTFTab.FindField("ID")
  GPTRWFld=GPTFTab.FindField("RW")
  GPTHWFld=GPTFTab.FindField("HW")
  GPTZWFld=GPTFTab.FindField("Hoehe50")
  GPTZMFld=GPTFTab.FindField("Hoehem")

  'Feststellung der Anzahl der in dem Thema schon vorhandenen Daten
  AnzG3DPT=0
  for each rec in GPTFTab
    AnzG3DPT = AnzG3DPT + 1
  end
  IdxG3DPT = AnzG3DPT - 1
  GrecNr = IdxG3DPT
  'MsgBox.Info(AnzG3DPT.AsString,
  '     "die Anzahl der Datensätze im Thema"++theGTheme.AsString)

  'ein Thema für Terrassenhöhe (TOK)
```

```
theTTheme=MsgBox.ChoiceAsString(ListofPtThms,
      "für Terrassenhöhe"+NL+
      "Der Name eines Punktthemas in View"++theKView.AsString,
      "Auswahl eines Themas")
TPTFTab=theTTheme.GetFTab
TPTShpFld=TPTFTab.FindField("Shape")
TPTIdFld=TPTFTab.FindField("ID")
TPTRWFld=TPTFTab.FindField("RW")
TPTHWFld=TPTFTab.FindField("HW")
TPTZWFld=TPTFTab.FindField("Hoehe50")
TPTZMFld=TPTFTab.FindField("Hoehem")

'Feststellung der Anzahl der in dem Thema schon vorhandenen Daten
AnzT3DPT = 0
for each rec in TPTFTab
  AnzT3DPT = AnzT3DPT + 1
end
IdxT3DPT = AnzT3DPT - 1
TrecNr = IdxT3DPT
'MsgBox.Info(AnzT3DPT.AsString,
'     "die Anzahl der Datensätze im Thema"++theTTheme.AsString)

'ein Thema für Quartärbasishöhe

theQTheme=MsgBox.ChoiceAsString(ListofPtThms,
      "für Quartärbasishöhe"+NL+
      "Der Name eines Punktthemas in View"++theKView.AsString,
      "Auswahl eines Themas")
QPTFTab=theQTheme.GetFTab
QPTShpFld=QPTFTab.FindField("Shape")
QPTIdFld=QPTFTab.FindField("ID")
QPTRWFld=QPTFTab.FindField("RW")
QPTHWFld=QPTFTab.FindField("HW")
QPTZWFld=QPTFTab.FindField("Hoehe50")
QPTZMFld=QPTFTab.FindField("Hoehem")


'Feststellung der Anzahl der in dem Thema schon vorhandenen Daten
AnzQ3DPT=0
for each rec in QPTFTab
  AnzQ3DPT = AnzQ3DPT + 1
end
IdxQ3DPT = AnzQ3DPT - 1
QrecNr = IdxQ3DPT
'MsgBox.Info(AnzQ3DPT.AsString,
'     "die Anzahl der Datensätze im Thema"++theQTheme.AsString)

'ein Thema für NT abgedeckte Höhe

theATheme=MsgBox.ChoiceAsString(ListofPtThms,
     "für NT abgedeckte Höhe"+NL+
     "Der Name eines Punktthemas in View"++theKView.AsString,
     "Auswahl eines Themas")
APTFTab=theATheme.GetFTab
APTShpFld=APTFTab.FindField("Shape")
APTIdFld=APTFTab.FindField("ID")
APTRWFld=APTFTab.FindField("RW")
APTHWFld=APTFTab.FindField("HW")
APTZWFld=APTFTab.FindField("Hoehe50")
APTZMFld=APTFTab.FindField("Hoehem")

'Feststellung der Anzahl der in dem Thema schon vorhandenen Daten
```

```
  AnzA3DPT=0
  for each rec in APTFTab
    AnzA3DPT = AnzA3DPT + 1
  end
  IdxA3DPT = AnzA3DPT - 1
  ArecNr = IdxA3DPT
  'MsgBox.Info(AnzA3DPT.AsString,
  '     "die Anzahl der Datensätze im Thema"++theATheme.AsString)

end


av.ShowMsg("Die Daten der Profilschnitte werden in ein Punkt-Thema"
     ++"für Geländehöhe"++theGTheme.AsString++"gespeichert.")
av.ShowStopButton


'Speicherung der Daten in das Thema für die Geländehöhe

GPTFTab.SetEditable(False)
GPTFTab.SetEditable(true)
GHAnz=ListofG3DP.Count
GHIndAnz=GHAnz-1
Ng=0
'MsgBox.Info(GHAnz.AsString,
'       "Anzahl der gesammelten Punkte für Geländehöhe")

for each Elm in 0..GHIndAnz
    Ng=Ng+1
    aPZ=ListofG3DP.Get(Elm)
    RWkrd=aPZ.Getx
    HWkrd=aPZ.Gety
    ZWkrd=aPZ.Getz
    ZMkrd=ZWkrd/ YFt
    a2DP=Point.Make(RWkrd, HWkrd)
    GrecNr = GrecNr+1
    GPTFTab.AddRecord
    GPTFTab.SetValue(GPTShpFld, GrecNr, a2DP)
    GPTFTab.SetValue(GPTIdFld, GrecNr, GrecNr)
    GPTFTab.SetValue(GPTRWFld, GrecNr, RWkrd)
    GPTFTab.SetValue(GPTHWFld, GrecNr, HWkrd)
    GPTFTab.SetValue(GPTZWFld, GrecNr, ZWkrd)
    GPTFTab.SetValue(GPTZMFld, GrecNr, ZMkrd)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/GHAnz*100)
  if (not more) then
     break
  end
end
GPTFTab.SetEditable(False)
theGTheme.UpdateLegend

'Speicherung der Daten in das Thema für die Terrassenhöhe

av.ShowMsg("Die Daten der Profilschnitte werden in ein Punkt-Thema"
      ++"für Terrassenoberkante"++theTTheme.AsString++"gespeichert.")
av.ShowStopButton

  TPTFTab.SetEditable(False)
  TPTFTab.SetEditable(true)
  THAnz=ListofT3DP.Count
  THIndAnz=THAnz-1
```

```
  Ng=0
'  MsgBox.Info(THAnz.AsString,
'       "Anzahl der gesammelten Punkte für Terrassenhöhe")

for each Elm in 0..THIndAnz
    Ng=Ng+1
    aPZ=ListofT3DP.Get(Elm)
    RWkrd=aPZ.Getx
    HWkrd=aPZ.Gety
    ZWkrd=aPZ.Getz
    ZMkrd=ZWkrd/ YFt
    a2DP=Point.Make(RWkrd, HWkrd)
    TrecNr = TrecNr+1
    TPTFTab.AddRecord
    TPTFTab.SetValue(TPTShpFld, TrecNr, a2DP)
    TPTFTab.SetValue(TPTIdFld, TrecNr, TrecNr)
    TPTFTab.SetValue(TPTRWFld, TrecNr, RWkrd)
    TPTFTab.SetValue(TPTHWFld, TrecNr, HWkrd)
    TPTFTab.SetValue(TPTZWFld, TrecNr, ZWkrd)
    TPTFTab.SetValue(TPTZMFld, TrecNr, ZMkrd)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/THAnz*100)
  if (not more) then
    break
  end
end
TPTFTab.SetEditable(False)
theTTheme.UpdateLegend

'Speicherung der Daten in das Thema für Quartärbasishöhe

av.ShowMsg("Die Daten der Profile werden in ein Punkt-Thema"
      ++"für Quartärbasishöhe"++theQTheme.AsString++"gespeichert.")
av.ShowStopButton

QPTFTab.SetEditable(False)
QPTFTab.SetEditable(true)
QHAnz=ListofQ3DP.Count
QHIndAnz=QHAnz-1
Ng=0
'MsgBox.Info(QHAnz.AsString,
'       "Anzahl der gesammelten Punkte für Quartärbasis")

for each Elm in 0..QHIndAnz
    Ng=Ng+1
    aPZ=ListofQ3DP.Get(Elm)
    RWkrd=aPZ.Getx
    HWkrd=aPZ.Gety
    ZWkrd=aPZ.Getz
    ZMkrd=ZWkrd/ YFt
    a2DP=Point.Make(RWkrd, HWkrd)
    QrecNr = QrecNr+1
    QPTFTab.AddRecord
    QPTFTab.SetValue(QPTShpFld, QrecNr, a2DP)
    QPTFTab.SetValue(QPTIdFld, QrecNr, QrecNr)
    QPTFTab.SetValue(QPTRWFld, QrecNr, RWkrd)
    QPTFTab.SetValue(QPTHWFld, QrecNr, HWkrd)
    QPTFTab.SetValue(QPTZWFld, QrecNr, ZWkrd)
    QPTFTab.SetValue(QPTZMFld, QrecNr, ZMkrd)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/QHAnz*100)
  if (not more) then
```

```
      break
    end
  end
  QPTFTab.SetEditable(False)
  theQTheme.UpdateLegend

  'Speicherung der Daten in das Thema für NT-abgedeckte Höhe

  av.ShowMsg("Die Daten der Profile werden in ein Punkt-Thema"
       ++"für NT abgd. Höhe"++theATheme.AsString++"gespeichert.")
  av.ShowStopButton

  APTFTab.SetEditable(False)
  APTFTab.SetEditable(true)
  AHAnz=ListofA3DP.Count
  AHIndAnz=AHAnz-1
  Ng=0
  'MsgBox.Info(AHAnz.AsString,
  '       "Anzahl der gesammelten Punkte für NT abgd. Höhe")

  for each Elm in 0..AHIndAnz
     Ng=Ng+1
     aPZ=ListofA3DP.Get(Elm)
     RWkrd=aPZ.Getx
     HWkrd=aPZ.Gety
     ZWkrd=aPZ.Getz
     ZMkrd=ZWkrd/ YFt
     a2DP=Point.Make(RWkrd, HWkrd)
     ArecNr = ArecNr+1
     APTFTab.AddRecord
     APTFTab.SetValue(APTShpFld, ArecNr, a2DP)
     APTFTab.SetValue(APTIdFld, ArecNr, ArecNr)
     APTFTab.SetValue(APTRWFld, ArecNr, RWkrd)
     APTFTab.SetValue(APTHWFld, ArecNr, HWkrd)
     APTFTab.SetValue(APTZWFld, ArecNr, ZWkrd)
     APTFTab.SetValue(APTZMFld, ArecNr, ZMkrd)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AHAnz*100)
    if (not more) then
      break
    end
  end
  APTFTab.SetEditable(False)
  theATheme.UpdateLegend




  'ptsammel.ave
  'Gleichartige Punkte, die in mehreren Themen in einem View
  'enthalten sind, werden teilweise in einem Punkt-Thema gesammelt.
  'Dieses Script wird als ein Menü zum Anklicken
  'in einem aktiven Karten-View benutzt.

  theProject=av.GetProject
  theView=av.GetActiveDoc  'ein aktives View
  myScript=theProject.FindScript("ptsammel")
  myScript.SetNumberFormat( "d.dd") ' script default

  MsgBox.Report("Gleichartige Punkte,"
          +NL+"z.B. der Rinnen oder der Bohrungen,"
```

```
          +NL+"die in mehreren Themen in einem View"
          +NL+"enthalten sind, werden teilweise"
          +NL+"in einem Punkt-Thema gesammelt",
            "Aufgabe dieses Programms")

'Eingabe eines Themas, um Punkte zu sammeln
ListofThemen = theView.GetThemes
ListofPtThemes = {}
for each aTh in ListofThemen
   if (aTh.Is(FTheme)) then
     aFTab = aTh.GetFTab
     if (aFtab.GetShapeClass.IsSubclassOf(Point)) then
       ListofPtThemes.Add(aTh)
     end
   end
end

PtTheme = MsgBox.ChoiceAsString(ListofPtThemes,
     "Name eines Punkt-Themas, um Punkte zu sammeln",
     "Auswahl eines Themas im View"++theView.AsString)
PtStr = PtTheme.AsString
PtFTab = PtTheme.GetFTab
ListofPtFlds = PtFTab.GetFields
AnzPtFlds = ListofPtFlds.Count 'Anzahl der Felder
IdxPtFlds = AnzPtFlds - 1

PtShpFld = PtFTab.FindField("Shape")

'Feststellung der Anzahl der Punkte in dem Thema
AnzPt = 0
for each rec in PtFTab
   AnzPt = AnzPt + 1
end
IdxPt = AnzPt - 1

BerNrStr = MsgBox.Input("um Punkte zu sammeln",
       "Anzahl der Bereiche im Thema"++PtStr, "1")
BerNr = BerNrStr.AsNumber
BerIdx = BerNr-1
ListofAnfId = {}
ListofEndId = {}
AnzGsUebern = 0
TeilAnzUn = 0
minIdStr = "0"
maxId = IdxPt.SetFormat("").SetFormat("d")

for each aBr in 0..BerIdx
  aN=(aBr+1).SetFormat("").SetFormat("d")
  AId = MsgBox.Input("Der kleinste Index des"++aN.AsString+". Bereiches",
        "Auswahl des Bereiches zur Punkt-sammlung", minIdStr)
  EId = MsgBox.Input("Der größte Index des"++aN.AsString+". Bereiches",
        "Auswahl des Bereiches zur Punkt-Sammlung", maxId.AsString)
  AnfId = AId.AsNumber
  EndId = EId.AsNumber
  ListofAnfId.Add(AnfId)
  ListofEndId.Add(EndId)
  TeilAnzUn = (EndId - AnfId) + 1
  AnzGsUebern = AnzGsUebern + TeilAnzUn
  minIdStr = ((EndId + 1).SetFormat("").SetFormat("d")).AsString
end

av.ShowMsg("Die Punkte werden aus dem Thema"
```

```
        ++PtStr++"gesammelt...")
av.ShowStopButton

ListofPts = {}
ListofPtIdx = {}
Ng=0

for each j in 0..BerIdx
  aBAnfId = ListofAnfId.Get(j)
  aBEndId = ListofEndId.Get(j)
  for each i in 0..IdxPt
    if ((i >= aBAnfId) and (i <= aBEndId)) then
      Ng = Ng+1
      aShp = PTFTab.ReturnValue(PtShpFld, i)
      ListofPts.Add(aShp)
      ListofPtIdx.Add(i)
      'Show percentage complete with enabled stop button
      more=av.SetStatus((Ng/(AnzGsUebern))*100)
      if (not more) then
        exit
      end
    end
  end
end

'Eingabe eines Themas, um Punkte zu speichern
ListofAW1 = {"in einem neuen Thema", "in einem im"++theView.AsString
         ++"vorhandenen Thema"}
AW1 = MsgBox.ChoiceAsString(ListofAW1,
  "um gesammelte Punkte zu speichern","Auswahl eines Punkt-Themas")
IdxAW1 = ListofAW1.FindByValue(AW1)

if (IdxAW1 = 0) then
  'Ein neues Feature-Shape-File (Point) wird hergestellt.
   aWD = av.GetProject.GetWorkDir.AsString
   afn = ("Pt"+(PtStr.LCase)).Left(6)
   fnStr = FileName.Make(aWD).MakeTmp(afn, "shp")
   fName = FileDialog.Put(fnStr, "*.shp", "Output Shape File (Point)")
   if (fName = nil) then exit end
   fName.SetExtension("shp")
   NPtFTab = FTab.MakeNew(fName, Point)

   ListofNFlds  = {}
   ListofNFlds2  = {}
   NIdgFld = Field.Make("IDg", #Field_LONG, 7, 0)
   ListofNFlds.Add(NIdgFld)

   for each i in 0..IdxPtFlds
     aFld = ListofPtFlds.Get(i)
     aFldStr = aFld.GetName
     if (aFldStr <> "Shape") then
       aNFld = aFld.Clone
       ListofNFlds.Add(aNFld)
       ListofNFlds2.Add(aNFld)
     end
   end

   NKwFld = Field.Make("Kennwort", #Field_Char, 10, 0)
   ListofNFlds.Add(NKwFld)

   NPtFTab.AddFields(ListofNFlds)
   NPtTheme = FTheme.Make(NPtFTab)
```

```
  theView.AddTheme(NPtTheme)
  NPtTheme.UpdateLegend
  recNr = -1

elseif (IdxAW1 = 1) then

  NPtTheme = MsgBox.ChoiceAsString(ListofPtThemes,
      "im View"++theView.AsString+NL+
      "Name eines Punkt-Themas, um Punkte zu speichern",
      "Auswahl eines Themas")

  NPtFTab = NPtTheme.GetFTab
  ListofNPtFlds = NPtFTab.GetFields
  NIdgFld = MsgBox.ListAsString(ListofNPtFlds,
    "im Thema"++NPtTheme.AsString+NL+
    "für Id der gesamten Punkte","Auswahl eines Feldes")
  NKwFld = MsgBox.ListAsString(ListofNPtFlds,
    "im Thema"++NPtTheme.AsString+NL+
    "für Kennwort der gesammelten Punkte", "Auswahl eines Feldes")

  'Feststellung der Anzahl der gesamten Punkte in dem Thema
  AnzPtG = 0
  for each rec in NPtFTab
     AnzPtG = AnzPtG + 1
  end
  PtGIdx = AnzPtG - 1
  recNr = PtGIdx

  AnzNPtFlds = ListofNPtFlds.Count
  IdxNPtFlds = AnzNPtFlds - 1
  ListofNFlds2  = {}
  for each i in 0..IdxNPtFlds
    aNFld = ListofNPtFlds.Get(i)
    aNFldStr = aNFld.GetName
    if (aNFldStr <> "Shape") then
      if (aNFldStr <> "IDg") then
        if (aNFldStr <> "Kennwort") then
          ListofNFlds2.Add(aNFld)
        end
      end
    end
  end
end
NPtStr = NPtTheme.AsString

ListofKw = {"1Z", "1L", "1R", "2Z", "2L", "2R", "3Z", "3L", "3R", "4Z", "4L", "4R",
         "5Z", "5L", "5R", "6Z", "6L", "6R", "7Z", "7L", "7R", "8Z", "8L", "8R",
         "9Z", "9L", "9R", "10Z", "10L", "10R", "11Z", "11L", "11R", "Bohr"}
theKw00 = MsgBox.ChoiceAsString(ListofKw, "Aus dem Thema"++PtStr+NL+
      "Auswahl der Kennzeichnung", "Herkunft der Punkte:")
theKw = MsgBox.Input("Eingabe einer Kennzeichnung für die gesammelten Punkte",
                "Veränderungsmöglichkeit", theKw00)

'Speicherung der gesammelten Punkte im Punkt-Thema
av.ShowMsg("Speicherung der gesammelten Punkte im Thema"++NPtStr++"...")

AnzNPt = ListofPts.Count
IdxNPt = AnzNPt-1

MsgBox.Info(PtStr++"gesammelten Punkte:"+NL+AnzNPt.AsString,
        "Anzahl der vom Thema")
```

```
ListofPtFlds2 = {}            'Liste der Felder im Herkunft-Thema
for each i in 0..IdxPtFlds
  aFld = ListofPtFlds.Get(i)
  aFldStr = aFld.GetName
  if (aFldStr <> "Shape") then
    ListofPtFlds2.Add(aFld)
  end
end
AnzFlds2 = ListofPtFlds2.Count
IdxFlds2 = AnzFlds2 - 1

ListofNPtFlds = NPtFTab.GetFields
NPtShpFld = ListofNPtFlds.Get(0)

NPtFTab.SetEditable(False)
NPtFTab.SetEditable(true)

for each j in 0..IdxNPt
  aPt = ListofPts.Get(j)
  recNr = recNr + 1
  NPtFTab.AddRecord
  NPtFTab.SetValue(NPtShpFld, recNr, aPt)
  NPtFTab.SetValue(NIdgFld, recNr, recNr)
  aIdx = ListofPtIdx.Get(j)

  for each i in 0..IdxFlds2
    aFld = ListofPtFlds2.Get(i)
    aValue = PtFTab.ReturnValue(aFld, aIdx)
    aNFld = ListofNFlds2.Get(i)
    NPtFTab.SetValue(aNFld, recNr, aValue)
  end
  NPtFTab.SetValue(NKwFld, recNr, theKw)
end

NPtFTab.SetEditable(False)
NPtTheme.UpdateLegend
```

```
'ptschtgr.ave
'Ein neues Punkt-Thema wird hergestellt, um Schichtengrenzen in einem
'Profilschnitt zu bestimmen. Die Schichtengrenzen werden durch Polygone
'für die Schichten in einem Profilschnitt-View und Bohrdaten eines Gebietes
'bestimmt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
myScript=theProject.FindScript("ptschtgr")
myScript.SetNumberFormat( "d.dd") ' script default

theView=av.GetActiveDoc  'ein aktives Karten-View für ein neues Punkt-Thema

'Eingabe eines Polygon-Themas für die stratigraphischen Schichten
'in einem Profilschnitt-View

ListofView={"Bprf_hg", "Eprf_hg", "Lprf_hg", "Qprf_hg",
          "Bprf_hgd", "Eprf_hgd", "Lprf_hgd", "Qprf_hgd"}
aViewStr=MsgBox.ChoiceAsString(ListofView, "für ein Profilschnitt-View"+NL+
```

```
       "um ein Polygon-Thema für Schichten zu wählen",
       "Auswahl eines Views")
aNViewStr = MsgBox.Input("Ein View für einen Profilschitt",
         "Veränderungsmöglichkeit", aViewStr)
thePgView = theProject.FindDoc(aNViewStr)

ListofThemes = thePgView.GetThemes
ListofPgThms = {}

for each aT in ListofThemes
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      aCNm = aFTab.GetShapeClass.GetClassName
      if (aCNm = "Polygon") then
         ListofPgThms.Add(aT)
      end
   end
end

thePgTheme=MsgBox.ChoiceAsString(ListofPgThms,
        "um ein Polygon-Thema für Schichten zu wählen",
        "Auswahl eines Themas")

PgFTab = thePgTheme.GetFTab
PgShpFld = PgFTab.FindField("Shape")
PgIdFld = PgFTab.FindField("ID")
PgSchtFld = PgFTab.FindField("Schichten")
PgKWFld = PgFTab.FindField("Kennwort")
PgAnzBFld = PgFTab.FindField("Anz-Bohr")

AnzPgs=0
For each rec in PgFTab
   AnzPgs=AnzPgs+1
end
IdxPgs=AnzPgs-1

ListofKW={}
ListofSchtN={}
For each Zeile in 0..IdxPgs
   theZKW=PgFTab.ReturnValue(PgKWFld, Zeile)
   theZSchtN=PgFTab.ReturnValue(PgSchtFld, Zeile)
   ListofKW.Add(theZKW)
   ListofSchtN.Add(theZSchtN)
end

'Eingabe der dBase-Tabelle auf der Festplatte,
'um die Daten für eine Tabelle zu erhalten,
'die durch dieses Programm hergestellt wird.
theFileName=FileDialog.Show("*.dbf", "dBase-File",
         "Eingabe der Tabelle für Bohrungsdaten")
if (theFileName=nil) then exit end

theVtab=Vtab.Make(theFileName,false,false)

'Feststellung der Anzahl der Datensätze in der dBase-Tabelle
AnzRec=0
for each rec in theVtab
   AnzRec=AnzRec+1
end
idxAnzRec=AnzRec-1

FldList=theVtab.GetFields
```

```
AnzFld=FldList.Count
IdxFld=AnzFld-1

theVtab.SetEditable(false)
ListofBNr={}
ListofRW={}
ListofHW={}
ListofGh={}
VBNRFld=theVtab.FindField("Nr")
VRWFld=theVtab.FindField("Rw")
VHWFld=theVtab.FindField("Hw")
VGhFld=theVtab.FindField("Ghoehe")
AnzBohr=0

for each aRec in theVTab
   AnzBohr=AnzBohr+1
   aVBNr=theVtab.ReturnValue(VBNRFld,aRec)
   aVRW=theVtab.ReturnValue(VRWFld,aRec)
   aVHW=theVtab.ReturnValue(VHWFld,aRec)
   aVGh=theVtab.ReturnValue(VGhFld,aRec)
   ListofBNr.Add(aVBNr)
   ListofRW.Add(aVRW)
   ListofHW.Add(aVHW)
   ListofGh.Add(aVGh)
end
IdxBohr=AnzBohr-1

'Umwandlung der Polygone in Liste der Stützpunkte
ListofListofListofPgPts={}
for each aPg in 0..IdxPgs
   Ng=aPg+1
   thePg=PgFTab.ReturnValue(PgShpFld, aPg)
   ListofthePg=thePg.AsList
   ListofListofPgPts={}
   ULNr=0
   for each aList in ListofthePg
      ULNr=ULNr+1
      ListofPgPts={}
      for each aPt in aList
         xKrd=aPt.Getx
         yKrd=aPt.Gety/50
         ListofPgPts.Add(xKrd@yKrd)
      end
      AnzPgPts=ListofPgPts.Count
      IdxPgPts=AnzPgPts-1
      ListofPgPts.Remove(IdxPgPts)
      MsgBox.ListAsString(ListofPgPts, ULNr.AsString
      ++". Unter-Liste in der Liste", Ng.AsString++". Polygon")
      ListofListofPgPts.Add(ListofPgPts)
   end
   ListofListofListofPgPts.Add(ListofListofPgPts)
end

av.ShowMsg("Bestimmung der Profilschnitt-Grenzen"
       ++"der Schichten als Punkte...")
av.ShowStopButton
Ng=0

ListofListofOkPt={}
ListofListofUkPt={}

for each aPg in 0..IdxPgs
```

```
Ng=Ng+1
ListofthePg=ListofListofListofPgPts.Get(aPg)
ListofOkPt={}
ListofUkPt={}
for each aBr in 0..IdxBohr
   'MsgBox.Info(aBr.AsString++". Bohrung", "Kontrolle")
   theOkyKrd=0
   theUkyKrd=0
   theRW=ListofRW.Get(aBr)
   theHW=ListofHW.Get(aBr)
   theGh=ListofGh.Get(aBr)
   theHWR=theHW*100.Round/100 'Genauigkeit der Daten
   IdxaList=-1
   ErsteHoehe=0
   ZweiteHoehe=0
   for each aList in ListofthePg
      IdxaList=IdxaList+1
      theSNrL=-1
      PolyPtIdx=-1
      GleicherHW=false
      theVxKrd=0
      for each aPt in aList
         PolyPtIdx=PolyPtIdx+1
         theSNrL=theSNrL+1
         xKrd=aPt.Getx
         xKrdR=xKrd*100.Round/100
         yKrd=aPt.Gety
         yKrdR=yKrd*100.Round/100
         if (xKrdR = theHWR) then
           if (theVxKrd = xKrd) then
             GleicherHW=true
           elseif (theVxKrd <> xKrd) then
             GleicherHW=false
           end
           if (Not GleicherHW) then
             if (ErsteHoehe = 0) then
               ErsteHoehe=yKrd
             elseif (ErsteHoehe <> 0) then
               ZweiteHoehe=yKrd
             end
           elseif (GleicherHW) then
             for each af in 0..IdxFld
                if (af > 2) then
                  aFld=FldList.Get(af)
                  if (aFld.IsTypeNumber) then
                    aHValue=theVtab.ReturnValue(aFld, aBr)
                    if (aHValue <> 0) then
                      if (af = 3) then
                        aPtH=theGh
                      elseif (af > 3) then
                        aPtH=(theGh-aHValue)
                      end
                      aPtHR=aPtH*100.Round/100
                      if (yKrdR = aPtHR) then
                        if (ZweiteHoehe = 0) then
                          ErsteHoehe=yKrd
                        elseif (ZweiteHoehe <> 0) then
                          ZweiteHoehe=yKrd
                        end
                      end
                    end
                  end
```

```
                end
              end
            end
          end
          theVxKrd=xKrd
        end
      end
      if (ErsteHoehe >= ZweiteHoehe) then
        theOkyKrd=ErsteHoehe
        theUkyKrd=ZweiteHoehe
      elseif (ErsteHoehe < ZweiteHoehe) then
        theOkyKrd=ZweiteHoehe
        theUkyKrd=ErsteHoehe
      end
      ListofOkPt.Add(theRW@theHW@theOkyKrd)
      ListofUkPt.Add(theRW@theHW@theUkyKrd)
    end
    ListofListofOkPt.Add(ListofOkPt)
    ListofListofUkPt.Add(ListofUkPt)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzPgs*100)
    if (not more) then
      break
    end
end

'Kontrolle der Daten
Rf=0
for each aP in 0..IdxPgs
    Rf=Rf+1
    KontrOkPt=ListofListofOkPt.Get(aP)
    KontrUkPt=ListofListofUkPt.Get(aP)
    MsgBox.ListAsString(KontrOkPt,
    "Liste der Oberkante der"++Rf.AsString+". Schichten",
          "Kontrolle der Daten")
    MsgBox.ListAsString(KontrUkPt,
    "Liste der Basis"++Rf.AsString+". Schichten", "Kontrolle der Daten")
end

'Die Punkte werden  in ein Thema gespeichert
'Ein Feature-Shape-File  (Point) für Schichtengrenzen wird hergestellt.

aWDStr=theProject.GetWorkDir.AsString
fnStr00 = thePgTheme.AsString.Left(4)

fnDefault=FileName.Make(aWDStr).MakeTmp("Ptgr"+fnStr00,"shp")
fnOutPut=FileDialog.Put(fnDefault, "*.shp", "Output shape File (Point)")
if (fnOutPut=nil) then exit end
fnOutPut.SetExtension("shp")
ftbOutPut=FTab.MakeNew(fnOutPut, Point)
ShapeFld=ftbOutPut.FindField("shape")
IDFld=Field.Make("ID", #Field_Short, 3, 0)
BNrFld=Field.Make("Bohr_Nr", #Field_Byte, 3, 0)
RWFld=Field.Make("RW", #Field_FLOAT, 10, 2)
HWFld=Field.Make("HW", #Field_FLOAT, 10, 2)
GhFld=Field.Make("GHoehe", #Field_FLOAT, 6, 2)
DeckFld=Field.Make("Deck_UMT2_B", #Field_FLOAT, 6, 2)
UMT2Fld=Field.Make("UMT2_Basis", #Field_FLOAT, 6, 2)
HlstFld=Field.Make("Deck_MMT_B", #Field_FLOAT, 6, 2)
MMTFld=Field.Make("MMT_Basis", #Field_FLOAT, 6, 2)

ListofFlds={IDFld, BNrFld, RWFld, HWFld, GhFld, DeckFld, UMT2Fld,
```

```
          HlstFld, MMTFld}
ListofBasisFld={DeckFld, UMT2Fld, HlstFld, MMTFld}
ftbOutput.AddFields(ListofFlds)

av.ShowMsg("Herstellung des neuen Point-Themes für Bohrungen ...")
av.ShowStopButton

ftbOutPut.SetEditable(false)
ftbOutPut.SetEditable(true)

Ng=0
recNr=-1

for each aBrIdx in 0..IdxBohr
   Ng=Ng+1
   theBNr=ListofBNr.Get(aBrIdx)
   theId=aBrIdx
   theRW=ListofRW.Get(aBrIdx)
   theHW=ListofHW.Get(aBrIdx)
   theShp=Point.Make(theRW, theHW)
   theGh=ListofGh.Get(aBrIdx)
   recNr=recNr+1
   ftbOutput.AddRecord
   ftbOutPut.SetValue(ShapeFld, recNr, theShp)
   ftbOutPut.SetValue(IDFld, recNr, theId)
   ftbOutPut.SetValue(BNrFld, recNr, theBNr)
   ftbOutPut.SetValue(RWFld, recNr, theRW)
   ftbOutPut.SetValue(HWFld, recNr, theHW)
   ftbOutPut.SetValue(GhFld, recNr, theGh)
   aBIdx=aBrIdx
   for each aPgL in 0..IdxPgs
      ListofSchtBasis=ListofListofUkPt.Get(aPgL)
      thePt=ListofSchtBasis.Get(aBIdx)
      PgRW=thePt.Getx
      PgHW=thePt.Gety
      PgHh=thePt.Getz
      if ((PgRW = theRW) and (PgHW = theHW)) then
        theyKrd=PgHh
      else
        theyKrd=0
      end
      theFld=ListofBasisFld.Get(aPgL)
      ftbOutPut.SetValue(theFld, recNr, theyKrd)
   end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzBohr*100)
  if (not more) then
    break
  end
end

ftbOutPut.SetEditable(false)
thmNew=FTheme.Make(ftbOutPut)
theView.AddTheme(thmNew)

'Veränderung der Legende des Themes im Profil
av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_Marker)
theMPalette=av.GetSymbolWin.GetPalette
theBasicMarker=(theMPalette.GetList(#PALETTE_LIST_Marker).Get(37))
theLegend=thmNew.GetLegend
theLegend.SetLegendType(#Legend_Type_Simple)
theSymbol=theLegend.GetSymbols.Get(0)
```

```
theSymbol.Copy(theBasicMarker)
theSymbol.SetColor(Color.GetBlack)
theSymbol.SetSize(7)

thmNew.UpdateLegend



'ptviewws.ave
'Ein neues Punkt-Thema wird im aktiven View aus einem Punkt-Thema
'in einem anderen View hergestellt.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc   'ein aktives View

myScript=theProject.FindScript("ptviewws")
myScript.SetNumberFormat( "d.dd") ' script default

ListofViews={"Qprfg_gg", "Qprf0_gg", "Qprf1_gg", "Qprf2_gg", "Qprf3_gg",
           "Qprf4_gg", "Qprf5_gg", "Qprf6_gg", "Qprf7_gg", "Qprf8_gg",
           "Qprf9_gg", "Bprf_gg", "Eprf_gg", "Lprf_gg", "Kt1_gg"}

aView = MsgBox.ChoiceAsString(ListofViews,
     "Ein View mit einem Punkt-Thema", "Auswahl eines Views")
aViewStr = aView.AsString

aNViewStr = MsgBox.Input("Ein View mit einem Punkt-Thema",
         "Veränderungsmöglichkeit", aViewStr)

aNView = theProject.FindDoc(aNViewStr)

ListofThms = aNView.GetThemes
ListofPtThms = {}

for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
         ListofPtThms.Add(aT)
      end
   end
end

aPtThm = MsgBox.ChoiceAsString(ListofPtThms,
       "im View"++aNView.AsString, "Eingabe einer Punkt-Datei")

PtFTab = aPtThm.GetFTab
ListofPtFlds = PtFTab.GetFields

AnzPtFlds = ListofPtFlds.Count
IdxPtFlds = AnzPtFlds - 1

'Feststellung des ersten Datensatzes
FldStr=""
DtStr=""
for each i in 0..IdxPtFlds
   aFld = ListofPtFlds.Get(i)
   aFldStr = aFld.GetName
```

```
   if (aFldStr <> "Shape") then
      FldStr = FldStr + aFldStr+"; "
      aValue = PtFTab.ReturnValue(aFld, 0)
      DtStr=DtStr+aValue.AsString+"; "
   end
end

MsgBox.Report("Die Namen der Felder"+NL+FldStr+NL+NL+
           "Der erste Datensatz:"+NL+DtStr,
           "Information vom Thema"++aPtThm.AsString)

xFld = MsgBox.ListAsString(ListofPtFlds,
     "für x-Koordinate im aktiven View",
     "Auswahl eines Feldes im Thema"++aPtThm.AsString)
yFld = MsgBox.ListAsString(ListofPtFlds,
     "für y-Koordinate im aktiven View",
     "Auswahl eines Feldes im Thema"++aPtThm.AsString)

AnzPt=0
for each Pt in PtFTab
   AnzPt=AnzPt+1
end
AnzPtIdx=AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"
    ++aPtThm.AsString)

'Ein neues Punkt-Thema wird in einem aktiven View hergestellt.
aWDStr=theProject.GetWorkDir.AsString
fnStr00 = aPtThm.AsString.Left(6)
fnStr=FileName.Make(aWDStr).MakeTmp(fnStr00,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName=nil) then exit end
fName.SetExtension("shp")

NPtFTab = aPtThm.ExportToFTab (fName)
ListofNPtFlds = NPtFTab.GetFields

NPtShpFld = NPtFTab.FindField("shape")

av.ShowMsg("Herstellung eines neuen Punkt-Themas ...")

NPtFTab.SetEditable(false)
NPtFTab.SetEditable(true)

for each j in 0..AnzPtIdx
  theX = PtFTab.ReturnValue(xFld, j)
  theY = PtFTab.ReturnValue(yFld, j)
  theNPt = Point.Make(theX, theY)
  NPtFTab.SetValue(NPtShpFld, j, theNPt)
end

NPtFTab.SetEditable(false)
NewPtThm = FTheme.Make(NPtFTab)
theView.AddTheme(NewPtThm)




'schmodg.ave
'Punkte eines Schichtenmodells und Punkte einer Verbreitungsgrenze
'der Schicht werden zusammengefügt. Dabei werden die Punkte,
```

```
'die sich in den beiden Themen befinden, nur einmal aufgenommen.
'Die Höhen der Punkte werden von der Verbreitungsgrenze übernommen.
'Ein neues Schichtenmodell wird damit hergestellt.
'Dieses Script wird als ein Menü zum Anklicken
'in einem aktiven Karten-View benutzt.

theProject=av.GetProject
myScript=theProject.FindScript("schmodmg")
myScript.SetNumberFormat( "d.dd") ' script default
thePtView=av.GetActiveDoc    'ein aktives Karten-View

ListofThms=thePtView.GetThemes
ListofFThm = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFTab = aT.GetFTab
      if (aFTab.GetShapeClass.IsSubclassOf(Point)) then
        ListofFThm.Add(aT)
      end
   end
end

Pt2Thm=MsgBox.ChoiceAsString(ListofFThm,
       "Der Name eines Punkt-Themas in View"++thePtView.AsString+NL+
       "für ein Schichtenmodell",
       "Eingabe einer Punkt-Datei (Input)")
if (Pt2Thm = nil) then
  MsgBox.Error("Der Name der Punkte-Datei fehlt!", "")
  exit
end

Pt2FTab=Pt2Thm.GetFTab
Listof2Flds=Pt2FTab.GetFields

Pt2ShpFld = Pt2FTab.FindField("Shape")
Pt2GHFld=MsgBox.ChoiceAsString(Listof2Flds, "für Geländehöhe",
       "Auswahl eines Feldes im Thema"++Pt2Thm.AsString)
Pt2THFld=MsgBox.ChoiceAsString(Listof2Flds, "für Höhe der Terrassenoberkante",
       "Auswahl eines Feldes im Thema"++Pt2Thm.AsString)
Pt2BHFld=MsgBox.ChoiceAsString(Listof2Flds, "für Höhe der Terrassenbasis",
       "Auswahl eines Feldes im Thema"++Pt2Thm.AsString)

Anz2Pt=0
for each rPt in Pt2FTab
   Anz2Pt=Anz2Pt+1
end
Anz2PtIdx=Anz2Pt-1
'MsgBox.Info(Anz2Pt.AsString, "Die Anzahl der Punkte im Thema"++Pt2Thm.AsString)

av.ShowMsg("Eine Liste für eine Verbreitungsgrenze wird hergestellt ...")
av.ShowStopButton

theGrPt=MsgBox.ChoiceAsString(ListofFThm,
    "Der Name eines Punkt-Themas in View"++thePtView.AsString+NL+
    "um die Punkte in Schichtenmodell hinzufügen",
    "Eingabe einer Punkt-Datei (Verbreitungsgrenze)")
if (theGrPt = nil) then
  MsgBox.Error("Der Name der Punkt-Datei fehlt!", "")
  exit
end

PtTheme=theGrPt
```

```
PtFTab=PtTheme.GetFTab
ListofFlds=PtFTab.GetFields

'Diese Grenz-Punkte haben gleiche Höhen der Ober- und Unterkante.
'Die beiden Höhen werden deshalb als eine Terrassenhöhe bezeichnet.

PtShpFld = PtFTab.FindField("Shape")
PtGHFld=MsgBox.ChoiceAsString(ListofFlds, "für Geländehöhe [m]",
      "Auswahl eines Feldes im Thema"++PtTheme.AsString)
PtTHFld=MsgBox.ChoiceAsString(ListofFlds, "für Höhe der Terrassenoberkante",
      "Auswahl eines Feldes im Thema"++PtTheme.AsString)
PtBHFld=MsgBox.ChoiceAsString(ListofFlds, "für Höhe der Terrassenbasis",
      "Auswahl eines Feldes im Thema"++PtTheme.AsString)

AnzPt=0
for each Pt in PtFTab
   AnzPt=AnzPt+1
end
AnzPtIdx=AnzPt-1
MsgBox.Info(AnzPt.AsString, "Die Anzahl der Punkte im Thema"++PtTheme.AsString)

ListofSp={}
ListofGH={}
ListofTH={}
ListofBH={}

Ng=0
for each aMP in 0..AnzPtIdx
   Ng=Ng+1
   thePt=PtFTab.ReturnValue(PtShpFld, aMP)
   theGH=PtFTab.ReturnValue(PtGHFld, aMP)
   theTH=PtFTab.ReturnValue(PtTHFld, aMP)
   theBH=PtFTab.ReturnValue(PtBHFld, aMP)
   ListofSp.Add(thePt)
   ListofGH.Add(theGH)
   ListofTH.Add(theTH)
   ListofBH.Add(theBH)
   'Show percentage complete with enabled stop button
   more=av.SetStatus((Ng/(AnzPt))*100)
   if (not more) then
      exit
   end
end
AnzNeuPt=ListofSp.Count
IdxNeuPt=AnzNeuPt-1

'Vergleich der beiden Themen.
'Die Punkte, die sich in den beiden Themen befinden,
'werden herausgefunden und in eine Liste gespeichert.
av.ShowMsg("Die Punkte in den beiden Themen werden verglichen ...")
av.ShowStopButton

ListofPtextra = {}
Ng = 0
for each i in 0..IdxNeuPt
   Ng = Ng + 1
   aGrPt = ListofSp.Get(i)
   Pt2FTab.SelectByPoint(aGrPt, 0, #VTAB_SELTYPE_NEW)
   if (PT2FTab.GetSelection <> nil) then
      for each rec in PT2FTab.GetSelection
         thePt=Pt2FTab.ReturnValue(Pt2ShpFld, rec)
         ListofPtextra.Add(thePt)
```

```
      end
    end
    'Show percentage complete with enabled stop button
    more=av.SetStatus((Ng/(AnzNeuPt))*100)
    if (not more) then
      exit
    end
  end
  Pt2Thm.ClearSelection

  av.ShowMsg("Die Punkte, die sich nur im Schichtenmodell"
    ++"befinden, werden in eine Liste gespeichert  ...")
  av.ShowStopButton

  'Die Punkte im Schichtenmodell, die sich in der gleichen Stelle
  'der Verbreitungsgrenze befinden, werden nicht aufgenommen.

  AnzPtextra = ListofPtextra.Count
  IdxPtextra = AnzPtextra - 1
  ListofPt3 = {}
  ListofGH3 = {}
  ListofTH3 = {}
  ListofBH3 = {}

  Ng = 0
  for each rNr in 0..Anz2PtIdx
    Ng = Ng + 1
    thePt=Pt2FTab.ReturnValue(Pt2ShpFld, rNr)
    theRW=thePt.Getx
    theHW=thePt.Gety
    SW = "ein"
    aCircle = Circle.Make(thePt, 10)
    for each i in 0..IdxPtextra
      aGlPt = ListofPtextra.Get(i)
      Kr1 = aCircle.Contains(aGlPt)
      if (Kr1) then
        aGlx = aGlPt.Getx
        aGly = aGlPt.Gety
        if ((theRW = aGlx) and (theHW = aGly)) then
          SW = "aus"
        end
      end
    end
    if (SW = "ein") then
      ListofPt3.Add(thePt)
      theGH=Pt2FTab.ReturnValue(Pt2GHFld, rNr)
      theTH=Pt2FTab.ReturnValue(Pt2THFld, rNr)
      theBH=Pt2FTab.ReturnValue(Pt2BHFld, rNr)
      ListofGH3.Add(theGH)
      ListofTH3.Add(theTH)
      ListofBH3.Add(theBH)
    end
    'Show percentage complete with enabled stop button
    more=av.SetStatus((Ng/(Anz2Pt))*100)
    if (not more) then
       exit
    end
  end

  'Speicherung der Punkte in das neue Thema für Schichtenmodell
  'Ein neues Thema für Punkte wird in einem Karten-View hergestellt.
  'Ein Feature Shape (Point) File für Verbreitungsgrenze wird hergestellt.
```

```
aWDStr=theProject.GetWorkDir.AsString
afnD=Pt2Thm.AsString.Left(6)
fnStr=FileName.Make(aWDStr).MakeTmp(afnD,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PtFTab3=FTab.MakeNew(fName, Point)
PtShpFld3=PtFTab3.FindField("shape")
PtIDFld3=Field.Make("ID", #Field_long, 7, 0)
PtRWFld3=Field.Make("RW", #Field_Float, 10, 2)
PtHWFld3=Field.Make("HW", #Field_Float, 10, 2)
PtGHFld3=Field.Make("GH2_m", #Field_Float, 7, 2)
PtTHFld3=Field.Make("TOKH2_m", #Field_Float, 7, 2)
PtBHFld3=Field.Make("TBas2_m", #Field_Float, 7, 2)
ListofPtFlds={PtIDFld3, PtRWFld3, PtHWFld3, PtGHFld3, PtTHFld3, PtBHFld3}
PtFTab3.AddFields(ListofPtFlds)

av.ShowMsg("Herstellung des neuen Point-Themas für Schichtenmodell ...")
PtFTab3.SetEditable(false)
PtFTab3.SetEditable(true)
AnzPt3=ListofPt3.Count
IdxPt3=AnzPt3 - 1
for each rNr in 0..IdxPt3
  thePt=ListofPt3.Get(rNr)
  theRW=thePt.Getx
  theHW=thePt.Gety
  theGH=ListofGH3.Get(rNr)
  theTH=ListofTH3.Get(rNr)
  theBH=ListofBH3.Get(rNr)
  PtFTab3.AddRecord
  PtFTab3.SetValue(PtShpFld3, rNr, thePt)
  PtFTab3.SetValue(PtIDFld3, rNr, rNr)
  PtFTab3.SetValue(PtRWFld3, rNr, theRW)
  PtFTab3.SetValue(PtHWFld3, rNr, theHW)
  PtFTab3.SetValue(PtGHFld3, rNr, theGH)
  PtFTab3.SetValue(PtTHFld3, rNr, theTH)
  PtFTab3.SetValue(PtBHFld3, rNr, theBH)
end
for each aPt in 0..IdxNeuPt
  thePt=ListofSp.Get(aPt)
  theRW=thePt.Getx
  theHW=thePt.Gety
  theGH=ListofGH.Get(aPt)
  theTH=ListofTH.Get(aPt)
  theBH=ListofBH.Get(aPt)
  recNr=AnzPt3+aPt
  PtFTab3.AddRecord
  PtFTab3.SetValue(PtShpFld3, recNr, thePt)
  PtFTab3.SetValue(PtIDFld3, recNr, recNr)
  PtFTab3.SetValue(PtRWFld3, recNr, theRW)
  PtFTab3.SetValue(PtHWFld3, recNr, theHW)
  PtFTab3.SetValue(PtGHFld3, recNr, theGH)
  PtFTab3.SetValue(PtTHFld3, recNr, theTH)
  PtFTab3.SetValue(PtBHFld3, recNr, theBH)
end
PtFTab3.SetEditable(false)
NewPtThm=FTheme.Make(PtFTab3)
thePtView.AddTheme(NewPtThm)
```

```
'thkopie.ave
'Ein FThema wird kopiert und in ein neues Thema gespeichert.
'Ein Menü oder eine Schaltfläche in einem aktiven View zum Anklicken.

theProject=av.GetProject
theView=av.GetActiveDoc   'Ein aktives View
aAktThm=theView.GetActiveThemes.Get(0) ' Ein aktives Thema zur Kopie

Frg11=MsgBox.YesNo("Ist das aktive Thema"++aAktThm.AsString
    ++"richtig?", "Frage zur Kopie", true)
if (Frg11) then
  aFThm=aAktThm
elseif (Not Frg11) then
  'Eingabe eines Themas zur Kopie
  ListofThms=theView.GetThemes
  ListofFThm = {}
  for each aT in ListofThms
     if (aT.Is(FTheme)) then
       ListofFThm.Add(aT)
     end
  end
  aFThm=MsgBox.ChoiceAsString(ListofFThm,
       "zur Kopie",
       "Auswahl eines FThemas im View"++theView.AsString)
end

'Ein Feature-Shape-File für das neue Thema wird hergestellt
aWDStr=theProject.GetWorkDir.AsString
afnD=aFThm.AsString
fnStr=FileName.Make(aWDStr).MakeTmp(afnD,"shp")
fnName=FileDialog.Put(fnStr, "*.shp", "Output shape File")
if (fnName=nil) then exit end
fnName.SetExtension("shp")
NFTab=aFThm.ExportToFTab (fnName)

thmNew=FTheme.Make(NFTab)
theView.AddTheme(thmNew)




'thm1vsb.ave
'Ein weiterer Querprofilschnitt, der in der bestimmten Entfernung vom HW
'vom aktiven Querprofilschnitt liegt, wird in View gezeigt oder ausgeschaltet.
'Dieses Script wird als ein Menü oder als eine Schaltfläche zum Anklicken
'in einem aktiven Querprofilschnitt-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc    'ein aktives Querprofilschnitt-View
myScript=theProject.FindScript("thm1vsb1")
myScript.SetNumberFormat( "d") ' script default

theTheme=theView.GetActiveThemes.Get(0)  'ein aktives Querprofilschnitt-Thema
ThStr=theTheme.AsString
FR=MsgBox.YesNo("Ist das aktive Thema"++ThStr++"richtig?", "Zwischen-Kontrolle", TRUE)
if (Not FR) then
  MsgBox.Error("Der Name ist falsch!"+NL+"Das aktive Thema muss neu gewählt werden!", "")
  exit
end
ListofWahl={"wird sichtbar.", "wird unsichtbar."}
Frg0=MsgBox.ChoiceAsString(ListofWahl, "Ein weiterer Querprofilschnitt",
```

```
          "Auswahl der Aufgabe")
B1=ThStr.Left(1)
ListofThStr=ThStr.AsTokens(B1+".shp")
AnzThStr = ListofThStr.Count
IdxThStr = AnzThStr - 1
GzThStr = ""
for each i in 0..IdxThStr
   aThStr = ListofThStr.Get(i)
   GzThStr = GzThStr + aThStr +NL
end
'MsgBox.Report(GzThStr, "Die Teile des aktiven Themas")
ThStr2=ListofThStr.Get(0)
ThNr2=ThStr2.AsNumber
'MsgBox.Info(ThNr2.AsString, "der HW des aktiven Themas")

'Berechnung des gesuchten Themas
EntfernStr=MsgBox.Input("Der Abstand des gesuchten Profils"
                  +NL+"vom aktiven Thema (+ oder -)",
                  "Eingabe eines Abstandes (HW in m)", "100")
EntfernNr=EntfernStr.AsNumber
ThNr2V=(ThNr2+EntfernNr)
ThStr2V=ThNr2V.AsString
nBStr=MsgBox.Input("um einen Querprofilschnitt damit zu finden",
      "Eingabe des ersten Buchstabens", "P")
PThStr2V=nBStr+ThStr2V+".shp"
VTh=PThStr2V
'FR=MsgBox.YesNo("Ist das gesuchte Thema"++VTh++"richtig?", "Zwischen-Kontrolle", TRUE)
'if (Not FR) then
'   MsgBox.Error("Der Name ist falsch!"+NL+"Der Name muss neu eingegeben werden!", "")
'   VTh=MsgBox.Input("Der Name des letzten Themas","Eingabe eines Themas", PThStr2V)
'end
if (VTh <> Nil) then
  VThTheme=theView.FindTheme(VTh)
  if (Frg0 = "wird sichtbar.") then
    VThTheme.SetVisible(TRUE)
  elseif (Frg0 = "wird unsichtbar.") then
    VThTheme.SetVisible(false)
  end
end
av.GetProject.SetModified(true)
```

```
'thm3vsb.ave
'Zwei benachbarte Querprofilschnitte, die in den bestimmten Entfernungen vom HW
'vom aktiven Querprofilschnitt liegen, werden in View gezeigt oder ausgeschaltet.
'Dieses Script wird als ein Menü oder als eine Schaltfläche zum Anklicken
'in einem aktiven Querprofilschnitt-View benutzt.

'Die Profile des aktiven, des letzten und des nächsten Themas werden in View gezeigt.
'Am 29.06.2003 hergestellt

theProject=av.GetProject
theView=av.GetActiveDoc    'ein aktives Querprofilschnitt-View
myScript=theProject.FindScript("thm3vsb1")
myScript.SetNumberFormat( "d") ' script default

theTheme=theView.GetActiveThemes.Get(0)
```

```
ThStr=theTheme.AsString

FR=MsgBox.YesNo("Ist das aktive Thema"++ThStr++"richtig?", "Zwischen-Kontrolle", TRUE)
if (Not FR) then
  MsgBox.Error("Der Name ist falsch!"+NL+"Das aktive Thema muss neu gewählt werden!", "")
  exit
end

ListofWahl={"werden sichtbar.", "werden unsichtbar."}
Frg0=MsgBox.ChoiceAsString(ListofWahl, "Zwei benachbarte Querprofilschnitte",
            "Auswahl der Aufgabe")

B1=ThStr.Left(1)
ListofThStr=ThStr.AsTokens(B1+".shp")
AnzThStr = ListofThStr.Count
IdxThStr = AnzThStr - 1
GzThStr = ""
for each i in 0..IdxThStr
   aThStr = ListofThStr.Get(i)
   GzThStr = GzThStr + aThStr +NL
end
'MsgBox.Report(GzThStr, "Die Teile des aktiven Themas")

ThStr2=ListofThStr.Get(0)
ThNr2=ThStr2.AsNumber
MsgBox.Info(ThNr2.AsString, "der HW des aktiven Themas")

'Auswahl des Abstandes zwischen den beiden benachbarten Themen
ListofAbst={50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 1000}
AbstNr=MsgBox.ChoiceAsString(ListofAbst,
         "Der Abstand zu den benachbarten Profilen",
         "Auswahl der Querprofilschnitte")
nBStr=MsgBox.Input("um benachbarte Querprofilschnitte damit zu finden",
      "Eingabe des ersten Buchstabens", "P")

'Berechnung des letzten Themas
ThNr2V=(ThNr2-AbstNr)
ThStr2V=ThNr2V.AsString
PThStr2V=nBStr+ThStr2V+".shp"
VTh=PThStr2V
FR=MsgBox.YesNo("Ist das letzte Thema"++VTh++"richtig?", "Zwischen-Kontrolle", TRUE)
if (Not FR) then
  MsgBox.Error("Der Name ist falsch!"+NL+"Der Name muss neu eingegeben werden!", "")
  VTh=MsgBox.Input("Der Name des letzten Themas","Eingabe eines Themas", PThStr2V)
end
if (VTh <> Nil) then
  VThTheme=theView.FindTheme(VTh)
  if (Frg0 = "werden sichtbar.") then
    VThTheme.SetVisible(TRUE)
  elseif (Frg0 = "werden unsichtbar.") then
    VThTheme.SetVisible(false)
  end
end

'Berechnung des nächsten Themas
ThNr2N=(ThNr2+AbstNr)
ThStr2N=ThNr2N.AsString
PThStr2N=nBStr+ThStr2N+".shp"
NTh=PThStr2N
FR=MsgBox.YesNo("Ist das nächste Thema"++NTh++"richtig?", "Zwischen-Kontrolle", TRUE)
if (Not FR) then
  MsgBox.Error("Der Name ist falsch!"+NL+"Der Name muss neu eingegeben werden!", "")
```

```
  NTh=MsgBox.Input("Der Name des nächsten Themas","Eingabe eines Themas", PThStr2N)
end

if (NTh <> Nil) then
  NThTheme=theView.FindTheme(NTh)
  if (Frg0 = "werden sichtbar.") then
    NThTheme.SetVisible(TRUE)
  elseif (Frg0 = "werden unsichtbar.") then
    NThTheme.SetVisible(false)
  end
end

av.GetProject.SetModified(true)
'VThTheme.UpdateLegend
'NThTheme.UpdateLegend




'thmbvsb.ave
'Querprofilschnitte, die in einem bestimmten Bereich vom HW in einem
'aktiven View liegen, werden in View gezeigt oder ausgeschaltet.
'Dieses Script wird als ein Menü oder als eine Schaltfläche
'zum Anklickenin einem aktiven Querprofilschnitt-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc    'ein aktives Querprofilschnitt-View
myScript=theProject.FindScript("thmbvsb1")
myScript.SetNumberFormat("d") ' script default

ListofWahl={"werden sichtbar.", "werden unsichtbar."}
Frg0=MsgBox.ChoiceAsString(ListofWahl,
  "Querprofilschnitte in einem bestimmten Bereich",
  "Auswahl der Aufgabe")

ListofBst={"P5", "X5", "U5", "V5", "S5", "R5", "F5", "Z5", "B5", "C5"}
myBst=MsgBox.ChoiceAsString(ListofBst,
    "Die ersten zwei Buchstaben der Querprofilschnitte",
    "Auswahl der Querprofilschnitte im View"++theView.AsString)
my1B=myBst.Left(1)

'Auswahl eines Abstandes zwischen den beiden benachbarten Themen
ListofAbst={50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 1000}
AbstNr=MsgBox.ChoiceAsString(ListofAbst,
    "Der Abstand zwischen den beiden benachbarten Profilschnitten",
    "Auswahl der Querprofilschnitte")

ListofProfile=theView.GetThemes
AnzProfile=ListofProfile.Count
IdxProfile=AnzProfile-1
MinHw=5642000
MaxHw=5618000
AnzPTh=0
ListofPThemes={}

for each aPrf in 0..IdxProfile
  theTheme=ListofProfile.Get(aPrf)
  ThStr=theTheme.AsString
  B2=ThStr.Left(2)
  B1=ThStr.Left(1)
  if (B2 = myBst) then
```

```
      ListofThStr=ThStr.AsTokens(B1+".shp")
      ThStr2=ListofThStr.Get(0)
      if (ThStr2.IsNumber) then
        ListofPThemes.Add(theTheme)
        AnzPTh=AnzPTh+1
        ThNr2=ThStr2.AsNumber
        if (ThNr2 < MinHw) then
          MinHw=ThNr2
        end
        if (ThNr2 > MaxHw) then
          MaxHw=ThNr2
        end
      end
    end
end

MiHW=MinHw.SetFormat("").SetFormat("d")
MaHW=MaxHw.SetFormat("").SetFormat("d")
MsgBox.Report("Der kleinste HW des aktiven Views:"++MiHw.AsString+NL+
            "Der größte HW des aktiven Views:  "++MaHw.AsString,
            "Kontrolle der Querprofilschnitte im View"++theView.AsString)

'Die Profile werden ausgewählt und gezeigt.
AnzPrfTh=((MaxHw-MinHw)/AbstNr)+1
if (AnzPTh <> AnzPrfTh) then
  MsgBox.Error("Die Anzahl der Themen im View"++theView.AsString++"ist falsch!"+NL+
              "Es gibt Themen, die nicht dem View zugehören.", "")
  exit
end

MsgBox.Info(AnzPrfTh.AsString,
     "die Anzahl der Querprofilschnitte im View"++theView.AsString)

Frg1=MsgBox.YesNo("Soll die Anzahl der Querprofilschnitte verkleinert werden?",
    "Kontrolle der Anzahl der Querprofilschnitte", FALSE)
if (Frg1) then
  MinHWStr=MsgBox.Input("Der kleinste HW der Querprofilschnitte",
    "Auswahl der Querprofilschnitte im View"++theView.AsString, MinHW.AsString)
  MaxHWStr=MsgBox.Input("Der größte HW der Querprofilschnitte",
    "Auswahl der Querprofilschnitte im View"++theView.AsString, MinHW.AsString)
  MinHW=MinHWStr.AsNumber
  MaxHW=MaxHWStr.AsNumber
  AnzPrfTh=((MaxHw-MinHw)/AbstNr)+1
end
MsgBox.Info(AnzPrfTh.AsString,
   "die Anzahl der Querprofilschnitte im View"++theView.AsString)

IdxPrfTh=AnzPrfTh-1

for each aPf in 0..IdxPrfTh
   theHW=MinHW+(aPf*AbstNr)
   HWStr=theHW.AsString
   PfStr=my1B+HWStr+".shp"
   PfTheme=theView.FindTheme(PfStr)
   if (Frg0 = "werden sichtbar.") then
     PfTheme.SetVisible(TRUE)
   elseif (Frg0 = "werden unsichtbar.") then
     PfTheme.SetVisible(false)
   end
end
av.GetProject.SetModified(true)
```

```
'thmnzg1.ave
'Ein nächster Querprofilschnitt nach dem sichtbaren Querprofilschnitt wird gezeigt.
'Dieses Script wird als ein Menü oder als eine Schaltfläche
'zum Anklicken in einem aktiven Querprofilschnitt-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("thmnzg1")
myScript.SetNumberFormat( "d") ' script default

aVisThm=theView.GetVisibleThemes.Get(0)

aVisThmStr=aVisThm.AsString
B1=aVisThmStr.Left(1)
ListofThStr=aVisThmStr.AsTokens(B1+".shp")
'MsgBox.ListAsString(ListofThStr, "Liste der Buchstaben", "Das sichtbare Thema")
ThStr=ListofThStr.Get(0)
ThNr=ThStr.AsNumber
'MsgBox.Info(ThNr.AsString, "der HW des sichtbaren Themas")

'Auswahl eines Abstandes zwischen den beiden benachbarten Themen
ListofAbst={50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 1000}
AbstNr=MsgBox.ChoiceAsString(ListofAbst,
    "Der Abstand (HW) zwischen den"
    +NL+"beiden benachbarten Querprofilschnitten",
    "Auswahl der Querprofilschnitte")

theHW=ThNr+(AbstNr)
theHWStr=theHW.AsString
theNThStr=B1+theHWStr+".shp"
NTheme=theView.FindTheme(theNThStr)

av.ShowMsg("Nächster Querprofilschnitt"++NTheme.AsString++"wird gezeigt.")

aVisThm.SetVisible(false)
av.GetProject.SetModified(true)
NTheme.SetVisible(true)
av.GetProject.SetModified(true)
```

```
'thmsuch.ave
'Dieses Script findet ein Thema in Table of Content.
'Dieses Script wird als ein Menü oder als eine Schaltfläche
'zum Anklicken in einem aktiven Karten-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc 'ein aktives Thema

theListofThms=theView.GetThemes
AnzThms=theListofThms.Count
IdxThms=AnzThms-1

'Bestimmung der Art der Suche
ListofSuche={"Ein Thema mit einem bestimmten Namen",
        "Das im View gezeigte Thema",
```

```
          "Das Thema mit den ersten 4 Buchstaben"}

SucheW=MsgBox.ChoiceAsString(ListofSuche,
          "Auswahl der Art der Suche",
          "Suche nach einem Thema im aktiven View")
IdxW = ListofSuche.FindByValue(SucheW)
if (IdxW = 0) then
  theStr=MsgBox.Input("Der Name des gesuchten Themas: ",
       "Suche eines Themas", "Grptmtlo.shp")
  gesThm=theView.FindTheme(theStr)
elseif (IdxW = 1) then
  ListofVisibleThms=theView.GetVisibleThemes
  gesThm=MsgBox.ChoiceAsString(ListofVisibleThms,
      "Ein im View gezeigtes Thema", "Auswahl eines Themas")
  theStr=gesThm.AsString
elseif (IdxW = 2) then
  theStr=MsgBox.Input("Die ersten 4 Buchstaben des Themas: ",
       "Suche nach einem Thema", "P562")
  Listof4StrThms={}
  for each aThm in 0..IdxThms
    theThm=theListofThms.Get(aThm)
    theThmStr=theThm.AsString
    theThm4Str=theThmStr.Left(4)
    if (theThm4Str = theStr) then
      Listof4StrThms.Add(theThm)
    end
  end
  gesThm=MsgBox.ChoiceAsString(Listof4StrThms,
    "das die 4 Buchstaben enthält.", "Auswahl eines Themas")
end
if (gesThm <> nil) then
  gesStr=gesThm.AsString
end
theThIdx=theListofThms.Find(gesThm)
theQu=((theThIdx+1)/(AnzThms)).SetFormat("").SetFormat("d.dd")
theVorIdx=theThIdx-1
theNachIdx=theThIdx+1

if ((theVorIdx > -1) and (theVorIdx < AnzThms)) then
  theVThm=theListofThms.Get(theVorIdx)
elseif ((theVorIdx < 0) or (theVorIdx > (AnzThms-1))) then
  theVThm=" "
end
if ((theNachIdx > -1) and (theNachIdx < AnzThms)) then
  theNThm=theListofThms.Get(theNachIdx)
elseif ((theNachIdx < 0) or (theNachIdx > (AnzThms-1))) then
  theNThm=" "
end

if (theThIdx <> -1) then
  MsgBox.Report("Der Name des Themas:"++gesStr++NL+
            "Die Indexnummer des Themas:"++theThIdx.AsString++NL+
            "von den"++AnzThms.AsString++"Themen"+NL+
            "Stelle des Themas:"++theQu.AsString+NL+NL+
            "Das letzte Thema:"++theVThm.AsString+NL+
            "Das jetzige Thema:"++gesStr++NL+
            "Das nächste Thema:"++theNThm.AsString, "Show Theme & Index")
elseif (theThIdx = -1) then
  MsgBox.Info("Es gibt kein Thema im View mit einem solchen Namen",
          "Ergebnis der Suche nach dem Thema"++theStr)
end
```

```
'tickktag.ave
'Skalenlinien werden für einen Karten-Rahmen in einem
'bestimmten Abstand von RW und HW als ein Thema hergestellt.
'Dieses Script wird als ein Menü zum Anklicken
'in einem aktiven Karten-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc  'ein aktives Karten-View
myScript=theProject.FindScript("tickktag")
myScript.SetNumberFormat( "d.dddd") ' script default

av.ShowMsg("Bestimmung der Koordinaten des Karten-Rahmens ...")
ListofRW = {"2558600.0000", "2582450.0000",
          "2563000.0000", "2568000.0000",
          "2577000.0000", "2580500.0000"}

ListofHW = {"5618450.0000", "5641050.0000",
          "5637000.0000", "5641000.0000",
          "5637900.0000", "5641000.0000"}

akRWStr = MsgBox.ChoiceAsString(ListofRW, "der kleinste RW",
      "Auswahl der Koordinaten des Rahmens")
agRWStr = MsgBox.ChoiceAsString(ListofRW, "der größte RW",
      "Auswahl der Koordinaten des Rahmens")
akHWStr = MsgBox.ChoiceAsString(ListofHW, "der kleinste HW",
      "Auswahl der Koordinaten des Rahmens")
agHWStr = MsgBox.ChoiceAsString(ListofHW, "der größte HW",
      "Auswahl der Koordinaten des Rahmens")

'Veränderungsmöglichkeit
kRWStr = MsgBox.Input("der kleinste RW (einen anderen Wert?)",
    "Veränderungsmöglichkeit des Karten-Rahmens", akRWStr)
gRWStr = MsgBox.Input("der größte RW (einen anderen Wert?)",
    "Veränderungsmöglichkeit des Karten-Rahmens", agRWStr)
kHWStr = MsgBox.Input("der kleinste HW (einen anderen Wert?)",
    "Veränderungsmöglichkeit des Karten-Rahmens", akHWStr)
gHWStr = MsgBox.Input("der größte HW (einen anderen Wert?)",
    "Veränderungsmöglichkeit des Karten-Rahmens", agHWStr)

kRW = kRWStr.AsNumber
gRW = gRWStr.AsNumber
kHW = kHWStr.AsNumber
gHW = gHWStr.AsNumber

ListofRW = {"2560000.0000", "2580000.0000",
          "2563000.0000", "2568000.0000",
          "2577000.0000", "2580000.0000"}

ListofHW = {"5620000.0000", "5640000.0000",
          "5637000.0000", "5641000.0000",
          "5638000.0000", "5641000.0000"}

av.ShowMsg("Bestimmung der Koordinaten der Skalenlinien ...")
akRWStr = MsgBox.ChoiceAsString(ListofRW, "der kleine RW",
      "Auswahl der Koordinaten der RW-Skalenlinien")
agRWStr = MsgBox.ChoiceAsString(ListofRW, "der große RW",
      "Auswahl der Koordinaten der RW-Skalenlinien")
akHWStr = MsgBox.ChoiceAsString(ListofHW, "der kleine HW",
```

```
      "Auswahl der Koordinaten der HW-Skalenlinien")
agHWStr = MsgBox.ChoiceAsString(ListofHW, "der große HW",
      "Auswahl der Koordinaten der HW-Skalenlinien")

'Veränderungsmöglichkeit
kRWStr = MsgBox.Input("der kleine RW (einen anderen Wert?)",
     "Veränderungsmöglichkeit der RW-Skalenlinien", akRWStr)
gRWStr = MsgBox.Input("der große RW (einen anderen Wert?)",
     "Veränderungsmöglichkeit der RW-Skalenlinien", agRWStr)
kHWStr = MsgBox.Input("der kleine HW (einen anderen Wert?)",
     "Veränderungsmöglichkeit der HW-Skalenlinien", akHWStr)
gHWStr = MsgBox.Input("der große HW (einen anderen Wert?)",
     "Veränderungsmöglichkeit der HW-Skalenlinien", agHWStr)

kRWSk = kRWStr.AsNumber
gRWSk = gRWStr.AsNumber
kHWSk = kHWStr.AsNumber
gHWSk = gHWStr.AsNumber

'Bestimmung eines Abstandes der Skalenlinien
ListofAbst = {"1000.0000", "2000.0000", "4000.0000"}

aRWabsStr = MsgBox.ChoiceAsString(ListofAbst,
      "für die RW-Skalenlinien",
      "Auswahl eines Abstandes der Skalenlinien")
aHWabsStr = MsgBox.ChoiceAsString(ListofAbst,
      "für die HW-Skalenlinien",
      "Auswahl eines Abstandes der Skalenlinien")

'Veränderungsmöglichkeit
RWabsStr = MsgBox.Input("die RW-Skalenlinien (einen anderen Wert?)",
       "Veränderungsmöglichkeit", aRWabsStr)
HWabsStr = MsgBox.Input("die HW-Skalenlinien (einen anderen Wert?)",
       "Veränderungsmöglichkeit", aHWabsStr)

RWabs = RWabsStr.AsNumber
HWabs = HWabsStr.AsNumber

'Eine Länge der Skalenlinien
aHWgL = gHW - kHW
LSK = (aHWgL/22600)*300.0000
LSK1 = (aHWgL/22600)*1425.0000
LSK2 = (aHWgL/22600)*950.0000
LSK3 = (aHWgL/22600)*3920.0000
LSK4 = (aHWgL/22600)*1150.0000

ListofPL2={}
ListofZahl = {}
ListofPts = {}
'RW-Skalenlinien für den Karten-Rahmen
ypt1=kHW-LSk
ypt2=kHW
ypt3=gHW
ypt4=gHW+LSk
xpt=kRWSk
ypts1=kHW-LSk1
ypts4=gHW+LSk2
xpts=kRWSk-((aHWgL/22600)*1425)

maxRW = gRW + 1
while (xpt < maxRW )
   ListofPL1={}
```

```
    ListofPL1.Add(xpt@ypt1)
    ListofPL1.Add(xpt@ypt2)
    ListofPL2.Add(ListofPL1)
    ListofZahl.Add(xpt.SetFormat("").SetFormat("d"))
    ListofPts.Add(xpts@ypts1)
    ListofPL1={}
    ListofPL1.Add(xpt@ypt3)
    ListofPL1.Add(xpt@ypt4)
    ListofPL2.Add(ListofPL1)
    ListofZahl.Add(xpt.SetFormat("").SetFormat("d"))
    ListofPts.Add(xpts@ypts4)
    xpt=xpt+RWabs
    xpts=xpt-((aHWgL/22600)*1425)
end

'HW-Skalenlinien für den Karten-Rahmen
xpt1=kRW-LSk
xpt2=kRW
xpt3=gRW
xpt4=gRW+LSk
ypt=kHWSk
xpts1=kRW-LSk3
xpts4=gRW+LSk4
ypts=kHWSk-260

maxHW = gHW + 1
while (ypt < maxHW )
    ListofPL1={}
    ListofPL1.Add(xpt1@ypt)
    ListofPL1.Add(xpt2@ypt)
    ListofPL2.Add(ListofPL1)
    ListofZahl.Add(ypt.SetFormat("").SetFormat("d"))
    ListofPts.Add(xpts1@ypts)
    ListofPL1={}
    ListofPL1.Add(xpt3@ypt)
    ListofPL1.Add(xpt4@ypt)
    ListofPL2.Add(ListofPL1)
    ListofZahl.Add(ypt.SetFormat("").SetFormat("d"))
    ListofPts.Add(xpts4@ypts)
    ypt=ypt+HWabs
    ypts=ypt-260
end
AnzSkL = ListofPL2.Count
IdxSkL = (AnzSkL - 1).SetFormat("").SetFormat("d")

'Ein Feature-Shape-File für Skalenlinien (Polyline) wird hergestellt.
aWDStr=av.GetProject.GetWorkDir.AsString
fnStr=FileName.Make(aWDStr).MakeTmp("Tickktg1","shp")
fName=FileDialog.Put(fnStr, "*.shp",
      "Output Shape File (Skalenlinien der Karte)")
if (fName=nil) then exit end
fName.SetExtension("shp")
SkFTab=FTab.MakeNew(fName, Polyline)
theIDField=Field.Make("ID", #FIELD_SHORT, 3, 0)
theZField=Field.Make("RW, HW", #FIELD_Float, 8, 0)
SkFTab.AddFields({theIDField, theZField})
theShapeField=SkFTab.FindField("shape")

SkFTab.SetEditable(false)
SkFTab.SetEditable(true)
for each rec in 0..IdxSkL
  ListofListofaSKL = {}
```

```
  ListofaSKL = ListofPL2.Get(rec)
  ListofListofaSKL.Add(ListofaSKL)
  aPolyL=PolyLine.Make(ListofListofaSKL)
  aZahl = ListofZahl.Get(rec).SetFormat("").SetFormat("d")
  SkFTab.AddRecord
  SkFTab.SetValue(theShapeField, rec, aPolyL)
  SkFTab.SetValue(theIDField, rec, rec)
  SkFTab.SetValue(theZField, rec, aZahl)
end
SkFTab.SetEditable(false)
thmNew=FTheme.Make(SkFTab)
theView.AddTheme(thmNew)

'Ein Feature-Shape-File (Point) wird hergestellt.
fnStr=FileName.Make(aWDStr).MakeTmp("Skkdk1g1","shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName=nil) then exit end
fName.SetExtension("shp")
NPtFTab=FTab.MakeNew(fName, point)
ShpFld=NPtFTab.FindField("shape")
IdFld=Field.Make("ID", #FIELD_SHORT, 2, 0)
RWFld=Field.Make("RW", #FIELD_FLOAT, 10, 2)
HWFld=Field.Make("HW", #FIELD_FLOAT, 10, 2)
HWLbFld=Field.Make("Text", #FIELD_CHAR, 8, 0)
ListofFld={IdFld, RWFld, HWFld, HWLbFld}
NPtFTab.AddFields(ListofFld)
NPtFTab.SetEditable(false)
NPtFTab.SetEditable(true)

AnzPts=ListofPts.Count
IdxPts=AnzPts-1
for each aRec in 0..IdxPts
   aPt=ListofPts.Get(aRec)
   ax=aPt.Getx
   ay=aPt.Gety
   aZ=ListofZahl.Get(aRec).SetFormat("").SetFormat("d")
   aZStr=aZ.AsString
   NPtFTab.AddRecord
   NPtFTab.SetValue(ShpFld, aRec, aPt)
   NPtFTab.SetValue(IdFld, aRec, aRec)
   NPtFTab.SetValue(RWFld, aRec, ax)
   NPtFTab.SetValue(HWFld, aRec, ay)
   NPtFTab.SetValue(HWLbFld, aRec, aZStr)
end
NPtFTab.SetEditable(false)
thmPtNew=FTheme.Make(NPtFTab)
theView.AddTheme(thmPtNew)

theGraphicList=theView.GetGraphics
for each aPt in 0..IdxPts
  aZ=ListofZahl.Get(aPt).SetFormat("").SetFormat("d")
  theGString=aZ.AsString
  theGPoint=ListofPts.Get(aPt)
  theGText=GraphicText.Make(theGString, theGPoint)
  theGText.SetAlignment(#TEXTCOMPOSER_JUST_CENTER)
  theTextSymbol=theGText.ReturnSymbols.Get(0)
  theTextSymbol.SetSize(12)
  newFont=Font.Make("Arial", "normal")
  theTextSymbol.SetFont(newFont)
  theTextSymbol.SetColor(Color.GetBlack)
  theGraphicList.Add(theGText)
end
```

```
theDisplay=theView.GetDisplay
theDisplay.Invalidate(true)



'tocord2r.ave
'Die Stelle eines Themas in TOC wird geändert.
'Das Thema wird von der vorderen Stelle nach hinten oder umgekehrt verschoben.
'Dieses Script wird als ein Menü oder als eine Schaltfläche
'zum Anklicken in einem aktiven Querprofilschnitt-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc

ListofThemes=theView.GetThemes
Anz=ListofThemes.Count
AnzIdx=Anz-1
theThm=MsgBox.ChoiceAsString(ListofThemes,
     "Der Name eines zu verschiebenden Themas",
     "Änderung der Rheienfolge in TOC")
IdxThm=ListofThemes.FindByValue(theThm)
MsgBox.Info(IdxThm.AsString, "Die Indexnummer des Themas in TOC")
FR1=MsgBox.YesNo("Ist die Indexnummer des Themas richtig?", "Kontrolle", true)

StelleThm=MsgBox.ChoiceAsString(ListofThemes,
       "Der Name eines Themas an der neuen Stelle",
       "Suche nach der neuen Stelle in TOC")
IdxStl=ListofThemes.Find(StelleThm)
MsgBox.Info(IdxStl.AsString, "Die Indexnummer der neuen Stelle in TOC")
FR2=MsgBox.YesNo("Ist die Indexnummer des Themas richtig?", "Kontrolle", true)

if ((FR1) or (FR2)) then
  AlteStelle=IdxThm
  NeuStelle=IdxStl
  if (NeuStelle > AlteStelle) then
    NeuSIdx=NeuStelle-1
  elseif (AlteStelle > NeuStelle) then
    NeuSIdx=NeuStelle
  end
  ListofThemes.Shuffle(ListofThemes.Get(AlteStelle), (NeuStelle))
  ListofThemes.Get(NeuSIdx).SetActive(true)
  theView.InvalidateTOC(nil)
  theView.GetDisplay.Invalidate(true)
elseif ((not FR1) or (not FR2)) then
  MsgBox.Error("Die Indexnummer des Themas ist falsch!", "")
  Exit
end



'tocordsb.ave
'Die Reihenfolge der Querprofilschnitte in TOC
'wird als eine Stapel-Arbeit geändert.
'Eine Gruppe der Querprofilschnitte mit einem Buchstaben wird
'vor einer Gruppe der Querprofilschnitte mit einem anderen
'Buchstaben nach den steigenden Hochwerten geordnet.
'Dieses Script wird als ein Menü oder als eine Schaltfläche
'zum Anklicken in einem aktiven Querprofilschnitt-View benutzt.
```

```
theProject=av.GetProject
theView=av.GetActiveDoc
myScript=theProject.FindScript("tocordsb")
myScript.SetNumberFormat("d")

av.ShowMsg("Änderung der Reihenfolge in TOC ...")
'av.ShowStopButton

'Bestimmung des minimalen und maximalen Hochwertes
'der zu verschiebenden Dateien im View

Listof2B={"Z5", "R5", "V5", "F5", "P5", "X5", "S5",
         "K5", "B5", "C5", "G5"}
the2BN=MsgBox.ChoiceAsString(Listof2B,
      "Die ersten zwei Buchstaben der Profile",
      "Auswahl der Profile, um in TOC zu verschieben")
the1B=the2BN.Left(1)
St2BN=MsgBox.ChoiceAsString(Listof2B,
     "Die ersten zwei Buchstaben der Querprofilschnitte",
     "Auswahl eines Querprofiles an der neuen Stelle")
Stelle1B=St2BN.Left(1)

ListofThemes=theView.GetThemes
AnzThms=ListofThemes.Count
ThmsIdx=AnzThms-1
minVHW=5642000
maxVHW=5618000

VTockenStr=Stelle1B+".shp"
for each eThm in 0..ThmsIdx
   theagTh=ListofThemes.Get(eThm)
   agThStr=theagTh.AsString
   erst2B=agThStr.Left(2)
   if (erst2B = St2BN) then
      ListofagThStr=agThStr.AsTokens(VTockenStr)
      aHWdPP=ListofagThStr.Get(0)
      aHWdPPNr=aHWdPP.AsNumber
      if (aHWdPPNr < minVHW) then
         minVHW=aHWdPPNr
      end
      if (aHWdPPNr > maxVHW) then
         maxVHW=aHWdPPNr
      end
   end
end

minPHW=5642000
maxPHW=5618000
TokenStr=the1B+".shp"
for each eThm in 0..ThmsIdx
   theaSTh=ListofThemes.Get(eThm)
   aSThStr=theaSTh.AsString
   erstS2B=aSThStr.Left(2)
   if (erstS2B = the2BN) then
      ListofaSThStr=aSThStr.AsTokens(TokenStr)
      aHWdSP=ListofaSThStr.Get(0)
      aHWdSPNr=aHWdSP.AsNumber
      if (aHWdSPNr < minPHW) then
         minPHW=aHWdSPNr
      end
      if (aHWdSPNr > maxPHW) then
```

```
        maxPHW=aHWdSPNr
      end
    end
end

minHWStr=minPHW.AsString
maxHWStr=maxPHW.AsString
VHWA=MsgBox.Input("um die Profilschnitte in TOC im View"
   ++theView.AsString++"umzuordnen",
   "Eingabe des kleinsten Hochwertes der"
   ++the1B.AsString+"-Profilschnitte", minHWStr)
VHWANr=VHWA.AsNumber

VHWE=MsgBox.Input("um die Profile in TOC im View"
    ++theView.AsString++"umzuordnen",
    "Eingabe des größten Hochwertes der"
    ++the1B.AsString+"-Profilschnitte", maxHWStr)
VHWENr=VHWE.AsNumber

MsgBox.Report("Der Name von View:          "++theView.AsString+NL+NL+
          "Der HW der zu verschiebenden Profile:"+NL+
          "Der kleinste HW der"++the1B.AsString+"-Profile:     "++VHWA+NL+
          "Der größte HW der"++the1B.AsString+"-Profile:      "++VHWE+NL+NL+
          "Der HW der Stelle, wohin die Profile zu verschieben sind:"+NL+
          "Der kleinste HW der"++Stelle1B++"-Profile:"++minVHW.AsString+NL+
          "Der größte HW der"++Stelle1B++"-Profile:  "++maxVHW.AsString,
          "Kontrolle der Daten")

FR=MsgBox.YesNo("Sind die Daten richtig?", "Kontrolle der Daten", TRUE)
if (Not FR) then
  MsgBox.Error("Die Daten sind falsch eingegeben!"+NL+
           "Das Programm wird abgebrochen!", "")
  exit
end

'Bestimmung der Stelle, wohin die Profilschnitte verschoben werden sollen.
PrfAbstStr=MsgBox.Input("Der Abstand zwischen den Querprofilschnitten",
                  "Eingabe der Daten", "50.00")
PrfAbst=PrfAbstStr.AsNumber

AnzB=((VHWENr-VHWANr)/PrfAbst)+1
AnzBIdx=(AnzB-1)
Ng=0

for each aBr in 0..AnzBIdx
   Ng=Ng+1
   theHW=(VHWANr+(aBr*PrfAbst))
   theHWStr=theHW.AsString
   theThmStr=the1B+theHWStr+".shp"
   theThm=theView.FindTheme(theThmStr)
   'MsgBox.Info(theThm.AsString, "Ein Theme für Umordnung")

   ListofThemes=theView.GetThemes
   AnzThms=ListofThemes.Count
   ThmsIdx=AnzThms-1

   for each aTh in 0..ThmsIdx
      aThm=ListofThemes.Get(aTh)
      aThmStr=aThm.AsString
      the2B=aThmStr.Left(2)
      if (the2B = St2BN) then
        ListofaThmStr=aThmStr.AsTokens("KBCGPFfVX.shp")
```

```
        aKeyStr=ListofaThmStr.Get(0)

        if (aKeyStr = theHWStr) then
          B1=aThmStr.Left(1)
          if (B1 = Stelle1B) then
            theThmIdx=aTh
          end
        end
      end
    end

    Stelle=theThmIdx
    SIdx=Stelle-1

    ListofThemes.Shuffle(theThm, (Stelle))

    ListofThemes.Get(SIdx).SetActive(true)
    theView.InvalidateTOC(nil)
    theView.GetDisplay.Invalidate(true)

'    aKey=MsgBox.YesNo("Weiter? :", "Changing the Order of Themes", TRUE)
'    if (not aKey) then
'       break
'    end

    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzB*100)
    if (not more) then
      break
    end
end




'vbrsch1.ave
'Eine Fläche der geologischen Schichten an einer beliebigen Höhe
'wird als ein horizontaler Schnitt des Gebietes hergestellt.
'Als Daten werden eine Tabelle eines Schichtenmodells
'für die gesamten Schichten des Gebietes und Schichtenmodelle
'der einzelnen Schichten benutzt.
'Dieses Script wird als ein Menü zum Anklicken
'in einem aktiven Karten-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc      'ein aktives Karten-View
ListofThms=theView.GetThemes

'Eingabe einer Höhe zum horizontalen Schnitt des Gebietes
aHStr=MsgBox.Input("zum horizontalen Schnitt des Gebietes",
    "Eingabe einer Höhe [m ü NN]", "40")
theHoehe=aHStr.AsNumber

'Erzeugung eines Rasters von Punkten
'in einem bestimmten Rasterabstand im ganzen Modellierungsgebiet

'Eingabe der Gauß-Krüger-Koordinaten des Gebietes
minRWStr=MsgBox.Input("der kleinste RW des Gebietes",
       "Eingabe der Gauß-Krüger-Koordinaten", "2558600.00")
maxRWStr=MsgBox.Input("der größte RW des Gebietes",
       "Eingabe der Gauß-Krüger-Koordinaten", "2582450.00")
```

```
minHWStr=MsgBox.Input("der kleinste HW des Gebietes",
       "Eingabe der Gauß-Krüger-Koordinaten", "5618450.00")
maxHWStr=MsgBox.Input("der größte HW des Gebietes",
       "Eingabe der Gauß-Krüger-Koordinaten", "5641050.00")
minRW=minRWStr.AsNumber
maxRW=maxRWStr.AsNumber
minHW=minHWStr.AsNumber
maxHW=maxHWStr.AsNumber

RAbstStr=MsgBox.Input("in S-N und W-E Richtungen",
      "Eingabe eines Rasterabstandes [m]", "50.00")
RAbst=RAbstStr.AsNumber

'Anzahl der Reihen und Spalten des Rasters
AnzRh=(maxHW-minHW)/RAbst+1
AnzSp=(maxRW-minRW)/RAbst+1
IdxRh=AnzRh-1
IdxSp=AnzSp-1

'Eingabe einer Tabelle für ein Schichtenmodell des Gebietes
aTable=MsgBox.Input("für ein Schichtenmodell des Gebietes",
      "Eingabe einer Tabelle im Projekt", "Schmgebt.dbf")
aVTab = theProject.FindDoc(aTable).GetVTab
ListofFlds=aVTab.GetFields
aSchIdFld=MsgBox.ChoiceAsString(ListofFlds, "das die Identifikationsnummer"
        +NL+"der Schichten enthält",
        "Auswahl eines Feldes im:"++aTable.AsString)
aNmAbkFld=MsgBox.ChoiceAsString(ListofFlds, "das die Abkürzung des Namens"
        +NL+"der Schichten enthält",
        "Auswahl eines Feldes im:"++aTable.AsString)
aSchmFld=MsgBox.ChoiceAsString(ListofFlds,
       "welches das Schichtenmodell der Schichten enthält",
       "Auswahl eines Feldes im:"++aTable.AsString)
aVerbFld=MsgBox.ChoiceAsString(ListofFlds,
       "welches die Verbreitungsgrenze der Schichten enthält",
       "Auswahl eines Feldes im:"++aTable.AsString)

'Bestimmung der Anzahl der Schichten im Schichtenmodell des Gebietes
AnzSch=0
for each rec in aVTab
   AnzSch=AnzSch+1
end
IdxSch=AnzSch-1
MsgBox.Report("im Schichtenmodell"++aTable.AsString+":"++AnzSch.AsString,
      "Anzahl der Schichten")

av.ShowMsg("Eingabe der Felder für Ober- und Unterkante"++
         "im Schichtenmodell der einzelnen Schichen")
av.ShowStopButton

ListofListofaSchmFld={}
Ng=0

for each arec in 0..IdxSch
   Ng=Ng+1
   aSchmStr=aVTab.ReturnValue(aSchmFld, arec)
   aSchmThm=theView.FindTheme(aSchmStr)
   aSchmFTab=aSchmThm.GetFTab
   ListofSchmFlds=aSchmFTab.GetFields
   aSchmTOKHFld=MsgBox.ChoiceAsString(ListofSchmFlds,
        "welches die Oberkante der Schichten enthält",
        "Auswahl eines Feldes im:"++aSchmStr)
```

```
    aSchmBasHFld=MsgBox.ChoiceAsString(ListofSchmFlds,
        "welches die Unterkante der Schichten enthält",
        "Auswahl eines Feldes im:"++aSchmStr)
    ListofaSchmFld={}
    ListofaSchmFld.Add(aSchmTOKHFld)
    ListofaSchmFld.Add(aSchmBasHFld)
    ListofListofaSchmFld.Add(ListofaSchmFld)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzSch*100)
    if (not more) then
      exit
    end
end

av.ShowMsg("Bestimmung der Schichten"
        ++"an den Rasterpunkten ...")
av.ShowStopButton

'Suche nach den Schichten, in deren Verbreitungsgrenze
'sich die Rasterpunkte befinden.
'Die Identifikationsnummer und Abkürzung der Schichten am Punkt
'werden herausgefunden.
'Die herausgefundenen Schichten der Rasterdaten werden
'in einem neuen Punkt-Thema gespeichert.

HStr=theHoehe.SetFormat("").SetFormat("d").AsString
NmStr="Schn"+HStr+"1"
WDStr=theProject.GetWorkDir.AsString
fnStr=FileName.Make(WDStr).MakeTmp(NmStr,"shp")
fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PtFTab=FTab.MakeNew(fName, Point)
PtShpFld=PtFTab.FindField("shape")
PtSIDFld=Field.Make("Scht_ID", #Field_Short, 2, 0)
PtNmAbkFld=Field.Make("Namen_Abk", #Field_Char, 6, 0)

ListofPtFlds={PtSIDFld, PtNmAbkFld}
PtFTab.AddFields(ListofPtFlds)

Ng=0
AnzgPt=AnzRh*AnzSp

recNr=-1
ListofListofRecord={}

for each aHWI in 0..IdxRh
  theHW=(aHWI*RAbst)+minHW
  for each aRWI in 0..IdxSp
    theRW=(aRWI*RAbst)+minRW
    Ng=Ng+1
    thePt=Point.Make(theRW, theHW)
    vorhanden=false
    for each arec in 0..IdxSch
      aPgStr=aVTab.ReturnValue(aVerbFld, arec)
      aPgThm=theView.FindTheme(aPgStr)
      aPgFTab=aPgThm.GetFTab
      aPgVgrSchFld=aPgFTab.FindField("Shape")
      aPgVgr=aPgFTab.ReturnValue(aPgVgrSchFld, 0)
      Qt11=aPgVgr.Contains(thePt)
      if (Qt11) then
```

```
aSchmStr=aVTab.ReturnValue(aSchmFld, arec)
aSchmThm=theView.FindTheme(aSchmStr)
aSchmFTab=aSchmThm.GetFTab
aShpFld=aSchmFTab.FindField("Shape")
aSchmFTab.SelectByPoint(thePt, 5, #VTAB_SELTYPE_NEW)
SelBitm=aSchmFTab.GetSelection
AnzPtSel=SelBitm.Count
ListofaSchmFld=ListofListofaSchmFld.Get(arec)
aTOKHFld=ListofaSchmFld.Get(0)
aBasHFld=ListofaSchmFld.Get(1)

if (AnzPtSel = 0) then
  aSchmFTab.SelectByPoint(thePt, 30, #VTAB_SELTYPE_NEW)
  Sel2Bitm=aSchmFTab.GetSelection
  AnzPtSel2=Sel2Bitm.Count

  if (AnzPtSel2 = 0) then
    vorhanden=false
  elseif (AnzPtSel2 = 1) then
    Sel2BitList=Sel2Bitm.AsList
    IdxSel2=Sel2BitList.Find(true)
    aTOKH=aSchmFTab.ReturnValue(aTOKHFld, IdxSel2)
    aBasH=aSchmFTab.ReturnValue(aBasHFld, IdxSel2)
    if ((theHoehe <= aTOKH) and (theHoehe >= aBasH)) then
      aSchId=aVTab.ReturnValue(aSchIdFld, arec)
      aAbk=aVTab.ReturnValue(aNmAbkFld, arec)
      vorhanden=true
    end
  elseif (AnzPtSel2 > 1) then
    minAbst=10000
    for each ar in Sel2Bitm
      aShp=aSchmFTab.ReturnValue(aShpFld, ar)
      theX=aShp.Getx
      theY=aShp.Gety
      Abstx=(theX-theRW)*(theX-theRW)
      Absty=(theY-theHW)*(theY-theHW)
      Abst=(Abstx+Absty).Sqrt.Abs
      if (Abst < minAbst) then
        minAbst=Abst
        aTOKH=aSchmFTab.ReturnValue(aTOKHFld, ar)
        aBasH=aSchmFTab.ReturnValue(aBasHFld, ar)
        if ((theHoehe <= aTOKH) and (theHoehe >= aBasH)) then
          aSchId=aVTab.ReturnValue(aSchIdFld, arec)
          aAbk=aVTab.ReturnValue(aNmAbkFld, arec)
          vorhanden=true
        end
      end
    end
  end
  Sel2Bitm.ClearAll
elseif (AnzPtSel = 1) then
  SelBitList=SelBitm.AsList
  IdxSel=SelBitList.Find(true)
  aTOKH=aSchmFTab.ReturnValue(aTOKHFld, IdxSel)
  aBasH=aSchmFTab.ReturnValue(aBasHFld, IdxSel)
  if ((theHoehe <= aTOKH) and (theHoehe >= aBasH)) then
    aSchId=aVTab.ReturnValue(aSchIdFld, arec)
    aAbk=aVTab.ReturnValue(aNmAbkFld, arec)
    vorhanden=true
  end

elseif (AnzPtSel > 1) then
```

```
            minAbst=10000
            for each ar in SelBitm
              aShp=aSchmFTab.ReturnValue(aShpFld, ar)
              theX=aShp.Getx
              theY=aShp.Gety
              Abstx=(theX-theRW)*(theX-theRW)
              Absty=(theY-theHW)*(theY-theHW)
              Abst=(Abstx+Absty).Sqrt.Abs
              if (Abst < minAbst) then
                minAbst=Abst
                aTOKH=aSchmFTab.ReturnValue(aTOKHFld, ar)
                aBasH=aSchmFTab.ReturnValue(aBasHFld, ar)
                if ((theHoehe <= aTOKH) and (theHoehe >= aBasH)) then
                  aSchId=aVTab.ReturnValue(aSchIdFld, arec)
                  aAbk=aVTab.ReturnValue(aNmAbkFld, arec)
                  vorhanden=true
                end
              end
            end
            SelBitm.ClearAll
          end
        end
        if (Not vorhanden) then
          aSchId=AnzSch
          aAbk="?"
        end
        theRPt=Point.Make(theRW, theHW)

        ListofRecord={}
        ListofRecord.Add(theRPt)
        ListofRecord.Add(aSchId)
        ListofRecord.Add(aAbk)
        ListofListofRecord.Add(ListofRecord)

        'Show percentage complete with enabled stop button
        more=av.SetStatus(Ng/AnzgPt*100)
        if (not more) then
          break
        end
      end
    end

    av.ShowMsg("Speicherung der Schichten"
            ++"an den Rasterpunkten ...")
    av.ShowStopButton

    AnzRecord=ListofListofRecord.Count
    IdxRc=AnzRecord-1

    Ng=0

    PtFTab.SetEditable(false)
    PtFTab.SetEditable(true)

    for each aPt in 0..IdxRc
        Ng=Ng+1
        ListofRecord={}
        ListofRecord=ListofListofRecord.Get(aPt)
        theRPt=ListofRecord.Get(0)
        aSchId=ListofRecord.Get(1)
        aAbk=ListofRecord.Get(2)
```

```
      PtFTab.AddRecord
      PtFTab.SetValue(PtShpFld, aPt, theRPt)
      PtFTab.SetValue(PtSIDFld, aPt, aSchId)
      PtFTab.SetValue(PtNmAbkFld, aPt, aAbk)

      'Show percentage complete with enabled stop button
      more=av.SetStatus(Ng/AnzgPt*100)
      if (not more) then
        break
      end
end

PtFTab.SetEditable(false)
NewFThm=FTheme.Make(PtFTab)
theView.AddTheme(NewFThm)
theTheme=NewFThm

'Legende des Schichten-Themas als Stapel-Arbeit
av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

aListofdfm = {"GH",53,"Deck",53,"D",53,"De",53,"TOK",14,"NT",14,
            "lNT",14,"rNT",17,"NA",5,"NTabF",5,"MT",26,"UMT",26,
            "UMT III",26, "UMT IV",26,"OMT",41,"MTl",26,"MTr",20,
            "HTl",32,"lMTn",27,"lMTs",50,"rMTn",21,"rMTm",57,
            "rMTs1",35,"rMTs2",11,"Präm",3,"HT",32,"rHTl",32,
            "lHTr",32,"rHTr",32,"HTr",32,"MTabF",8,"QB",8,"NQ",8,
            "Präq",3,"?",9}

theLegend=theTheme.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(theTheme, "Namen_Abk")
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofNr={}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   aIdxLb = aListofdfm.FindByValue(theKlasseLb)
   if (aIdxLb <> -1) then
     aCNr = aListofdfm.Get(aIdxLb + 1)
   elseif (aIdxLb = -1) then
     aR = i Mod 60
     if (aR < 3) then
       aCNr = 2
     elseif (aR > 2) then
       aCNr = aR
     end
   end
   aListofNr.Add(aCNr)
end

aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
'MsgBox.Info(AnzSymb.AsString, "Anzahl der Symbole")
AnzSymbIdx=AnzSymb-2
aListofColor={}

for each Nmb in 0..IdxKlasse
  aNumb=aListofNr.Get(Nmb)
```

```
  theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
  theRgbList=theColor.GetRgbList
  aColor=Color.Make
  aColor.SetRgbList(theRgbList)
  aListofColor.Add(aColor)
end

for each symb in 0..AnzSymbIdx
   aListofSymbol.Get(symb).SetColor(aListofColor.Get(symb))
end

theTheme.UpdateLegend




'vbrsgr2.ave
'Verbreitungsgrenzen der geologischen Schichten an einer
'beliebigen Höhe werden als ein horizontaler Schnitt des
'Gebietes hergestellt. Als Daten werden eine Tabelle eines
'Schichtenmodells für die gesamten Schnichten des Gebietes
'und interpolierte Höhenlinien-Karten der Schichten benutzt.
'Dieses Script wird als ein Menü zum Anklicken
'in einem aktiven Karten-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc  'ein aktives Karten-View
aPrj = theView.GetProjection
ListofThms=theView.GetThemes

ListofPolyLThm = {}
for each aT in ListofThms
   if (aT.Is(FTheme)) then
      aFThmFTab=aT.GetFTab
      if (aFThmFTab.GetShapeClass.IsSubclassOf(Polyline)) then
        ListofPolyLThm.Add(aT)
      end
   end
end

'Eingabe einer Höhe zum horizontalen Schnitt des Gebietes
aHStr=MsgBox.Input("zum horizontalen Schnitt des Gebietes"+NL+"(ganze Zahl)",
    "Eingabe einer Höhe [m ü NN]", "40")
theHoehe=aHStr.AsNumber.SetFormat("d")

'Eingabe einer Tabelle für ein Schichtenmodell des Gebietes
aTable=MsgBox.Input("für ein Schichtenmodell des Gebietes",
      "Eingabe einer Tabelle im Projekt", "Schmgebt.dbf")
aVTab = theProject.FindDoc(aTable).GetVTab
ListofFlds=aVTab.GetFields
aSchIdFld=MsgBox.ChoiceAsString(ListofFlds, "das die Identifikationsnummer"
        +NL+"der Schichten enthält",
        "Auswahl eines Feldes im:"++aTable.AsString)
aNmAbkFld=MsgBox.ChoiceAsString(ListofFlds, "das die Abkürzung des Namens"
        +NL+"der Schichten enthält",
        "Auswahl eines Feldes im:"++aTable.AsString)
aNamenFld=MsgBox.ChoiceAsString(ListofFlds, "das den Namen der Schichten enthält",
        "Auswahl eines Feldes im:"++aTable.AsString)
aSchmFld=MsgBox.ChoiceAsString(ListofFlds,
       "welches das Schichtenmodell der Schichten enthält",
       "Auswahl eines Feldes im:"++aTable.AsString)
```

```
aVerbFld=MsgBox.ChoiceAsString(ListofFlds,
       "welches die Verbreitungsgrenze der Schichten enthält",
       "Auswahl eines Feldes im:"++aTable.AsString)

'Bestimmung der Anzahl der Schichten im Schichtenmodell des Gebietes
AnzSch=0
for each rec in aVTab
   AnzSch=AnzSch+1
end
IdxSch=AnzSch-1
MsgBox.Report("im Schichtenmodell"++aTable.AsString+":"++AnzSch.AsString,
      "Anzahl der Schichten")

av.ShowMsg("Suche nach dem Schichtenmodell,"++
          "das die eingegebene Höhe enthält")
av.ShowStopButton

Ng=0
ListofSID={}
ListofSchm={}
ListofAbks={}
ListofSNm={}
ListofVgr={}

for each arec in 0..IdxSch
   Ng=Ng+1
   aSchmStr=aVTab.ReturnValue(aSchmFld, arec)
   aSchmThm=theView.FindTheme(aSchmStr)
   aSchmFTab=aSchmThm.GetFTab
   ListofSchmFlds=aSchmFTab.GetFields
   aSchmTOKHFld=MsgBox.ChoiceAsString(ListofSchmFlds,
        "welches die Oberkante der Schichten enthält",
        "Auswahl eines Feldes im:"++aSchmStr)

   aSchmBasHFld=MsgBox.ChoiceAsString(ListofSchmFlds,
        "welches die Unterkante der Schichten enthält",
        "Auswahl eines Feldes im:"++aSchmStr)
   'Anzahl der Punkte im Schichtenmodell
   AnzSPt=0
   for each aSP in aSchmFTab
     AnzSPt=AnzSPt+1
   end
   IdxSPt=AnzSPt-1

   aSP=0
   While (aSP < AnzSPt)
     aObKH=aSchmFTab.ReturnValue(aSchmTOKHFld, aSP)
     aUntKH=aSchmFTab.ReturnValue(aSchmBasHFld, aSP)
     if ((theHoehe <= aObKH) and (theHoehe >= aUntKH)) then
       ListofSchm.Add(aSchmThm)
       aAbk=aVTab.ReturnValue(aNmAbkFld, arec)
       aSNm=aVTab.ReturnValue(aNamenFld, arec)
       aSID=aVTab.ReturnValue(aSchIdFld, arec)
       aVgr=aVTab.ReturnValue(aVerbFld, arec)
       ListofAbks.Add(aAbk)
       ListofSNm.Add(aSNm)
       ListofSID.Add(aSID)
       ListofVgr.Add(aVgr)
       break
     end
     aSP=aSP+1
   end
```

```
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzSch*100)
    if (not more) then
      exit
    end
end

'Anzahl der Schichten, die die eingegebene Höhe enthalten.
AnzSchm=ListofSchm.Count
IdxSchm=AnzSchm-1

'Suchen nach den Höhenlinien der rasterinterpolierten
'Oberfläche oder des TINes der Schichten

av.ShowMsg("Herstellung der Höhenlinien ...")
av.ShowStopButton

ListofSchapes={}
ListofNmAbk={}
ListofSNm2={}
ListofCtFL={}
ListofSID2={}
ListofVgr2={}
ListofFlaeche={"Oberkante", "Unterkante"}
Ng=0

for each aS in 0..IdxSchm
    Ng=aS+1
    aSchm=ListofSchm.Get(aS)
    aAbk=ListofAbks.Get(aS)
    aSNm=ListofSNm.Get(aS)
    aSID=ListofSID.Get(aS)
    aVgr=ListofVgr.Get(aS)
    aVgrThm=theView.FindTheme(aVgr)
    aVgrFTab=aVgrThm.GetFTab
    aVgrShpFld=aVgrFTab.FindField("Shape")
    aVgrPg=aVgrFTab.ReturnValue(aVgrShpFld, 0)

    for each aFL in 0..1
      theFL = ListofFlaeche.Get(aFL)
      Qt11=MsgBox.YesNo("Hat die Fläche eine interpolierte Höhenlinien"
          +NL+"(Höhenlinien oder Umrisse)?",
          theFL++"von"++aSNm.AsString, true)

      if (Qt11) then
        'Eingabe der interpolierten Linien-Datei (Umrisse)
        aPLTheme = MsgBox.ListAsString(ListofPolyLThm,
              "Auswahl eines Themas, um als Höhenlinien zu benutzen:",
              "Umrisse für:"++ aSchm.GetName ++ theFL)
        if (aPLTheme <> NIL) then
          aPLFTab=aPLTheme.GetFTab
          ListofaPLFTab = aPLFTab.GetFields
          aPLShpFld=aPLFTab.FindField("Shape")
          aPLCtFld = MsgBox.ChoiceAsString(ListofaPLFTab,
              "das die Höhenlinien"
              +NL+"der Schicht enthält",
              "Auswahl eines Feldes im:"++aPLTheme.AsString)
          AnzCt=0
          for each aCtIdx in aPLFTab
            AnzCt=AnzCt+1
          end
```

```
          IdxCt=AnzCt-1

          for each aIdx in 0..IdxCt
            aCtNr=aPLFTab.ReturnValue(aPLCtFld, aIdx).SetFormat("d")
            if (aCtNr = theHoehe) then
              theCtShp=aPLFTab.ReturnValue(aPLShpFld, aIdx)
              'Test, ob die herausgefundenen Linien
              'innerhalb der Verbreitungsgrenze liegen
              aIntersect=aVgrPg.Intersects(theCtShp)
              if (aIntersect) then
                theNCtShp=theCtShp.LineIntersection(aVgrPg)
                ListofSchapes.Add(theNCtShp)
                ListofNmAbk.Add(aAbk)
                ListofSNm2.Add(aSNm)
                ListofCtFL.Add(theFL)
                ListofSID2.Add(aSID)
              elseif (Not aIntersect) then
                if (aVgrPg.Contains(theCtShp)) then
                  ListofSchapes.Add(theCtShp)
                  ListofNmAbk.Add(aAbk)
                  ListofSNm2.Add(aSNm)
                  ListofCtFL.Add(theFL)
                  ListofSID2.Add(aSID)
                end
              end
            end
          end

        elseif (aPLTheme = NIL) then
          continue
        end
      end
    end
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/AnzSchm*100)
    if (not more) then
      exit
    end
  end


  av.ShowMsg("Speicherung der Höhenlinien ...")
  av.ShowStopButton

  'Die herausgefundenen Höhenlinien werden
  'in einem neuen Polyline-Thema gespeichert.

  HStr=theHoehe.SetFormat("").SetFormat("d").AsString
  NmStr="Schc"+HStr+"1"
  WDStr=theProject.GetWorkDir.AsString
  fnStr=FileName.Make(WDStr).MakeTmp(NmStr,"shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Polyline)")
  if (fName=nil) then exit end
  fName.SetExtension("shp")
  PLFTab=FTab.MakeNew(fName, Polyline)
  PLShpFld=PLFTab.FindField("shape")
  PLCIDFld=Field.Make("ID", #Field_Short, 5, 0)
  PLSIDFld=Field.Make("Scht_ID", #Field_Short, 2, 0)
  PLCtFld=Field.Make("Contour", #Field_Short, 5, 0)
  PLNmAbkFld=Field.Make("Namen_Abk", #Field_Char, 6, 0)
  PLNmFld=Field.Make("Namen", #Field_Char, 30, 0)
  PLFLFld=Field.Make("Flaeche", #Field_Char, 12, 0)
```

```
ListofPLFlds={PLCIDFld, PLSIDFld, PLCtFld, PLNmAbkFld, PLNmFld, PLFLFld}
PLFTab.AddFields(ListofPLFlds)

AnzPL=ListofSchapes.Count
IdxPL=AnzPL-1

PLFTab.SetEditable(false)
PLFTab.SetEditable(true)

for each aPLIdx in 0..IdxPL
   thePLShp=ListofSchapes.Get(aPLIdx)
   theSID=ListofSID2.Get(aPLIdx)
   theAbk=ListofNmAbk.Get(aPLIdx)
   theNm=ListofSNm2.Get(aPLIdx)
   theFL=ListofCtFL.Get(aPLIdx)

   PLFTab.AddRecord
   PLFTab.SetValue(PLShpFld, aPLIdx, thePLShp)
   PLFTab.SetValue(PLCIDFld, aPLIdx, aPLIdx)
   PLFTab.SetValue(PLSIDFld, aPLIdx, theSID)
   PLFTab.SetValue(PLCtFld, aPLIdx, theHoehe)
   PLFTab.SetValue(PLNmAbkFld, aPLIdx, theAbk)
   PLFTab.SetValue(PLNmFld, aPLIdx, theNm)
   PLFTab.SetValue(PLFLFld, aPLIdx, theFL)
end

PLFTab.SetEditable(false)
NewFThm=FTheme.Make(PLFTab)
theView.AddTheme(NewFThm)
theTheme=NewFThm

'Legende des Schichten-Themas als Stapel-Arbeit
av.GetSymbolWin.SetPanel(#SYMBOLWIN_PANEL_COLOR)
thePalette=av.GetSymbolWin.GetPalette

aListofdfm = {"GH",53,"Deck",53,"D",53,"De",53,"TOK",14,"NT",14,
            "lNT",14,"rNT",17,"NA",5,"NTabF",5,"MT",26,"UMT",26,
            "UMT III",26, "UMT IV",26,"OMT",41,"MTl",26,"MTr",20,
            "HTl",32,"lMTn",27,"lMTs",50,"rMTn",21,"rMTm",57,
            "rMTs1",35,"rMTs2",11,"Präm",3,"HT",32,"rHTl",32,
            "lHTr",32,"rHTr",32,"HTr",32,"MTabF",8,"QB",8,"NQ",8,
            "Präq",3,"?",9}

theLegend=theTheme.GetLegend
theLegend.SetLegendType(#Legend_Type_Unique)
theLegend.Unique(theTheme, "Namen_Abk")
ListofKlasse=theLegend.GetClassifications
AnzKlasse=ListofKlasse.Count
IdxKlasse=AnzKlasse-2
'MsgBox.Info(AnzKlasse.AsString, "Anzahl der Klasse")
aListofNr={}
for each i in 0..IdxKlasse
   theKlasseLb=ListofKlasse.Get(i).GetLabel
   'MsgBox.Info(theKlasseLb.AsString, "Name der Klasse")
   aIdxLb = aListofdfm.FindByValue(theKlasseLb)
   if (aIdxLb <> -1) then
      aCNr = aListofdfm.Get(aIdxLb + 1)
   elseif (aIdxLb = -1) then
      aR = i Mod 60
      if (aR < 3) then
         aCNr = 2
```

```
      elseif (aR > 2) then
        aCNr = aR
      end
    end
    aListofNr.Add(aCNr)
end

aListofSymbol=theLegend.GetSymbols
AnzSymb=aListofSymbol.Count
'MsgBox.Info(AnzSymb.AsString, "Anzahl der Symbole")
AnzSymbIdx=AnzSymb-2
aListofColor={}

for each Nmb in 0..IdxKlasse
  aNumb=aListofNr.Get(Nmb)
  theColor=(thePalette.GetList(#PALETTE_LIST_COLOR). Get(aNumb))
  theRgbList=theColor.GetRgbList
  aColor=Color.Make
  aColor.SetRgbList(theRgbList)
  aListofColor.Add(aColor)
end

for each symb in 0..AnzSymbIdx
   aListofSymbol.Get(symb).SetColor(aListofColor.Get(symb))
end

theTheme.UpdateLegend




'w_kor_g.ave
'Ein Kreis oder Radien eines Kreises in Abstand vom 10 Grad oder beides werden
'in einem aktiven Karten-View als eine Grafik gezeichnet. Dieses Script wird
'als ein Menü zum Anklicken in einem aktiven Karten-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc  'ein aktives Karten-View

ListofAufgaben={"eines Kreises", "der Radien eines Kreises",
              "eines Kreises und deren Radien"}
qt1=MsgBox.ChoiceAsString(ListofAufgaben, "Zeichnung",
  "Auswahl der Zeichnung als eine Grafik")

ListofRWs={"2565000.00", "2574000.00", "2579000.00"}
ListofHWs={"5632000.00", "5639000.00"}
ListofRds={"2000.00", "8000.00"}
theRWStr=MsgBox.ChoiceAsString(ListofRWs,
       "Rechtswert des Zentrums im View"++theView.AsString,
       "Zeichnung"++qt1)
theHWStr=MsgBox.ChoiceAsString(ListofHWs,
       "Hochwert des Zentrums im View"++theView.AsString,
       "Zeichnung"++qt1)
theRDStr=MsgBox.ChoiceAsString(ListofRds,
       "Radius des Kreises im View"++theView.AsString,
       "Zeichnung"++qt1)

'Veränderungsmöglichkeit
the2RWStr=MsgBox.Input("Rechtswert des Zentrums (einen anderen Wert?)",
        "Zeichnung"++qt1, theRWStr)
the2HWStr=MsgBox.Input("Hochwert des Zentrums (einen anderen Wert?)",
```

```
        "Zeichnung"++qt1, theHWStr)
the2RDStr=MsgBox.Input("Radius des Kreises (einen anderen Wert?)",
        "Zeichnung"++qt1, theRDStr)
theRW=the2RWStr.AsNumber
theHW=the2HWStr.AsNumber
theRD=the2RDStr.AsNumber
theGraphicList=theView.GetGraphics

if ((qt1 = "eines Kreises") or
   (qt1 = "eines Kreises und deren Radien")) then
   mycir=circle.Make(theRW@theHW,theRD)
   mycirshape=GraphicShape.Make(mycir)
   theGraphicList.Add(mycirshape)
end

if ((qt1 = "der Radien eines Kreises") or
   (qt1 = "eines Kreises und deren Radien")) then
   ListofPL={}
   for each aW in 0..35
      aRad=(aW*10).AsRadians
      aX=theRD*(aRad.Sin)+theRW
      aY=theRD*(aRad.Cos)+theHW
      ListofPT={}
      ListofPT.Add(theRW@theHW)
      ListofPT.Add(aX@aY)
      ListofPL.Add(ListofPT)
   end
   myPolyL=PolyLine.Make(ListofPL)
   theGraphicPolyLine=GraphicShape.Make(myPolyL)
   theGraphicList.Add(theGraphicPolyLine)
end




'w_lnrand.ave
'Die Linien der Radien eines Kreises in Abstand vom 10 Grad,
'die bis zum Rand des Modellierungsgebiet verlängert worden sind,
'werden im aktiven View als ein PolyLine-Thema hergestellt und gezeichnet.
'Dieses Script wird als ein Menü zum Anklicken in einem aktiven
'Karten-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc   'ein aktives Karten-View

ListofRWs={"2565000.00", "2574000.00", "2579000.00"}
ListofHWs={"5632000.00", "5639000.00"}
ListofRds={"2000.00", "8000.00"}
qt1="der Linien bis zum Rand"

theRWStr=MsgBox.ChoiceAsString(ListofRWs,
       "Rechtswert des Zentrums im View"++theView.AsString,
       "Zeichnung"++qt1)
theHWStr=MsgBox.ChoiceAsString(ListofHWs,
       "Hochwert des Zentrums im View"++theView.AsString,
       "Zeichnung"++qt1)
theRDStr=MsgBox.ChoiceAsString(ListofRds,
       "Radius des Kreises im View"++theView.AsString,
       "Zeichnung"++qt1)

'Veränderungsmöglichkeit
```

```
the2RWStr=MsgBox.Input("Rechtswert des Zentrums (einen anderen Wert?)",
      "Zeichnung"++qt1, theRWStr)
the2HWStr=MsgBox.Input("Hochwert des Zentrums (einen anderen Wert?)",
      "Zeichnung"++qt1, theHWStr)
the2RDStr=MsgBox.Input("Radius des Kreises (einen anderen Wert?)",
      "Zeichnung"++qt1, theRDStr)

theRW=the2RWStr.AsNumber
theHW=the2HWStr.AsNumber
theRD=the2RDStr.AsNumber

ListofMRWs={"2558600.00", "2582450.00"}
ListofMHWs={"5618450.00", "5641050.00"}

theMinXStr=MsgBox.ChoiceAsString(ListofMRWs,
      "Der kleinste Rechtswert des Gebietes im View"++theView.AsString,
      "Zeichnung"++qt1)
theMaxXStr=MsgBox.ChoiceAsString(ListofMRWs,
      "Der größte Rechtswert des Gebietes im View"++theView.AsString,
      "Zeichnung"++qt1)

theMinYStr=MsgBox.ChoiceAsString(ListofMHWs,
      "Der kleinste Hochwert des Gebietes im View"++theView.AsString,
      "Zeichnung"++qt1)
theMaxYStr=MsgBox.ChoiceAsString(ListofMHWs,
      "Der größte Hochwert des Gebietes im View"++theView.AsString,
      "Zeichnung"++qt1)

'Änderungsmöglichkeit
the2MinXStr=MsgBox.Input("Der kleinste Rechtswert des Gebietes",
      "Änderungsmöglichkeit", theMinXStr)
the2MaxXStr=MsgBox.Input("Der größte Rechtswert des Gebietes",
      "Änderungsmöglichkeit", theMaxXStr)

the2MinYStr=MsgBox.Input("Der kleinste Hochwert des Gebietes",
      "Änderungsmöglichkeit", theMinYStr)
the2MaxYStr=MsgBox.Input("Der größte Hochwert des Gebietes",
      "Änderungsmöglichkeit", theMaxYStr)

theMinX=the2MinXStr.AsNumber
theMaxX=the2MaxXStr.AsNumber
theMinY=the2MinYStr.AsNumber
theMaxY=the2MaxYStr.AsNumber

ListofgPT={}
ListofListofPL={}
ListofKPt={}
for each aW in 0..17
  aRad=(aW*10).AsRadians
  aX=theRD*(aRad.Sin)+theRW
  aY=theRD*(aRad.Cos)+theHW
  ListofKPt.Add(aX@aY)
  agRad=((aW+18)*10).AsRadians
  agX=theRD*(agRad.Sin)+theRW
  agY=theRD*(agRad.Cos)+theHW
  ListofKPt.Add(agX@agY)
  if (aW = 0) then
    theXkl=theRW
    theXgr=theRW
    theYkl=theMinY
    theYgr=theMaxY
  elseif ((aW > 0) and (aW < 9)) then
```

```
      Neig=(aY-theHW)/(aX-theRW)
      Achs=theHW-(Neig*theRW)
      aX1=(theMaxY-Achs)/Neig
      if ((aX1 < theMaxX) or (aX1 = theMaxX)) then
        theXgr=aX1
        theYgr=theMaxY
      elseif (aX1 > theMaxX) then
        theXgr=theMaxX
        theYgr=Neig*theXgr+Achs
      end
      aX2=(theMinY-Achs)/Neig
      if ((aX2 = theMinX) or (aX2 > theMinX)) then
        theXkl=aX2
        theYkl=theMinY
      elseif (aX2 < theMinX) then
        theXkl=theMinX
        theYkl=Neig*theXkl+Achs
      end
    elseif (aW = 9) then
      theXkl=theMinX
      theXgr=theMaxX
      theYkl=theHW
      theYgr=theHW
    elseif ((aW > 9) and (aW < 18)) then
      Neig=(aY-theHW)/(aX-theRW)
      Achs=theHW-(Neig*theRW)
      aX1=(theMinY-Achs)/Neig
      if ((aX1 < theMaxX) or (aX1 = theMaxX)) then
        theXkl=aX1
        theYkl=theMinY
      elseif (aX1 > theMaxX) then
        theXkl=theMaxX
        theYkl=Neig*theXkl+Achs
      end
      aX2=(theMaxY-Achs)/Neig
      if ((aX2 = theMinX) or (aX2 > theMinX)) then
        theXgr=aX2
        theYgr=theMaxY
      elseif (aX2 < theMinX) then
        theXgr=theMinX
        theYgr=Neig*theXgr+Achs
      end
    end

    ListofPT={}
    ListofPT.Add(theXkl@theYkl)
    ListofPT.Add(theXgr@theYgr)
    ListofgPT.Add(ListofPT)
    ListofPL={}
    ListofPL.Add(ListofPT)
    ListofListofPL.Add(ListofPL)
  end

  'Ein Feature-Shape-File für die Linien (Polyline) wird hergestellt.
  aWD=av.GetProject.GetWorkDir.AsString
  fnStr=FileName.Make(aWD).MakeTmp("Radien11","shp")
  fName=FileDialog.Put(fnStr, "*.shp",
      "Output Shape File (Linien bis zum Rand)")
  if (fName=nil) then exit end
  fName.SetExtension("shp")
  PLFTab=FTab.MakeNew(fName, Polyline)
  PLIDFld=Field.Make("ID", #FIELD_SHORT, 2, 0)
```

```
PLWKFld=Field.Make("Winkel", #FIELD_SHORT, 4, 0)
PLRWklFld=Field.Make("RW1", #FIELD_Float, 10, 2)
PLHWklFld=Field.Make("HW1", #FIELD_Float, 10, 2)
PLRWgrFld=Field.Make("RW2", #FIELD_Float, 10, 2)
PLHWgrFld=Field.Make("HW2", #FIELD_Float, 10, 2)

ListofPLFlds={PLIDFld,PLWKFld,PLRWklFld,PLHWklFld,PLRWgrFld,PLHWgrFld}
PLFTab.AddFields(ListofPLFlds)
PLShpFld=PLFTab.FindField("shape")

AnzofList=ListofListofPL.Count
IdxofList=AnzofList-1

PLFTab.SetEditable(false)
PLFTab.SetEditable(true)

for each aL in 0..IdxofList
   ListofLinien=ListofListofPL.Get(aL)
   myPolyL=PolyLine.Make(ListofLinien)
   PLFTab.AddRecord
   PLFTab.SetValue(PLShpFld, aL, myPolyL)
   PLFTab.SetValue(PLIDFld, aL, aL)
   theWK=aL*10
   PLFTab.SetValue(PLWKFld, aL, theWK)
   ListofPT=ListofgPT.Get(aL)
   thePTkl=ListofPT.Get(0)
   thePTgr=ListofPT.Get(1)
   aRWkl=thePTkl.Getx
   aHWkl=thePTkl.Gety
   aRWgr=thePTgr.Getx
   aHWgr=thePTgr.Gety
   PLFTab.SetValue(PLRWklFld, aL, aRWkl)
   PLFTab.SetValue(PLHWklFld, aL, aHWkl)
   PLFTab.SetValue(PLRWgrFld, aL, aRWgr)
   PLFTab.SetValue(PLHWgrFld, aL, aHWgr)
end

PLFTab.SetEditable(false)
ThmNew=FTheme.Make(PLFTab)
theView.AddTheme(ThmNew)

'Ein Feature-Shape-File für Punkte auf dem Rand des Modellierungsgebietes
fnStr=FileName.Make(aWD).MakeTmp("Randpt11","shp")
fName=FileDialog.Put(fnStr, "*.shp",
      "Output shape File (Punkte auf dem Rand)")
if (fName=nil) then exit end
fName.SetExtension("shp")
RPtFTab=FTab.MakeNew(fName, Point)
RPtShpFld=RPtFTab.FindField("shape")
RPtIDFld=Field.Make("ID", #Field_Short, 4, 0)
RPtXFld=Field.Make("RW", #Field_FLOAT, 10, 2)
RPtYFld=Field.Make("HW", #Field_FLOAT, 10, 2)

ListofRPtFlds={RPtIDFld, RPtXFld, RPtYFld}
RPtFTab.AddFields(ListofRPtFlds)

RPtFTab.SetEditable(false)
RPtFTab.SetEditable(true)

recNr=-1
for each aP in 0..IdxofList
   ListofPT=ListofgPT.Get(aP)
```

```
   thePTkl=ListofPT.Get(0)
   aRWkl=thePTkl.Getx
   aHWkl=thePTkl.Gety
   RPtFTab.AddRecord
   recNr=recNr+1
   RPtFTab.SetValue(RPtShpFld, recNr, thePTkl)
   RPtFTab.SetValue(RPtIDFld, recNr, recNr)
   RPtFTab.SetValue(RPtxFld, recNr, aRWkl)
   RPtFTab.SetValue(RPtyFld, recNr, aHWkl)

   thePTgr=ListofPT.Get(1)
   aRWgr=thePTgr.Getx
   aHWgr=thePTgr.Gety
   RPtFTab.AddRecord
   recNr=recNr+1
   RPtFTab.SetValue(RPtShpFld, recNr, thePTgr)
   RPtFTab.SetValue(RPtIDFld, recNr, recNr)
   RPtFTab.SetValue(RPtxFld, recNr, aRWgr)
   RPtFTab.SetValue(RPtyFld, recNr, aHWgr)
end

RPtFTab.SetEditable(false)
RthmNew=FTheme.Make(RPtFTab)
theView.AddTheme(RthmNew)

'Ein Feature-Shape-File für Punkte auf dem Kreis
fnStr=FileName.Make(aWD).MakeTmp("Kreispt1","shp")
fName=FileDialog.Put(fnStr, "*.shp",
      "Output shape File (Punkte auf dem Kreis)")
if (fName=nil) then exit end
fName.SetExtension("shp")
KPtFTab=FTab.MakeNew(fName, Point)
KPtShpFld=KPtFTab.FindField("shape")
KPtIDFld=Field.Make("ID", #Field_Short, 4, 0)
KPtXFld=Field.Make("RW", #Field_FLOAT, 10, 2)
KPtYFld=Field.Make("HW", #Field_FLOAT, 10, 2)

ListofKPtFlds={KPtIDFld, KPtXFld, KPtYFld}
KPtFTab.AddFields(ListofKPtFlds)

KPtFTab.SetEditable(false)
KPtFTab.SetEditable(true)

AnzKPt=ListofKPt.Count
IdxKPt=AnzKPt-1

for each aT in 0..IdxKPt
   theKPt=ListofKPT.Get(aT)
   aRW=theKPT.Getx
   aHW=theKPT.Gety
   KPtFTab.AddRecord
   KPtFTab.SetValue(KPtShpFld, aT, theKPT)
   KPtFTab.SetValue(KPtIDFld, aT, aT)
   KPtFTab.SetValue(KPtxFld, aT, aRW)
   KPtFTab.SetValue(KPtyFld, aT, aHW)
end

KPtFTab.SetEditable(false)
KthmNew=FTheme.Make(KPtFTab)
theView.AddTheme(KthmNew)
```

```
'wandpg1.ave
'Eine Wand wird für eine offene Stelle in einer 3D-Blockzeichnung
'als PolygonZ hergestellt. Als Daten werden eine Tabelle des Schichtenmodells
'des gesamten Gebietes und Schichtenmodelle der einzelnen Schichten
'in dem aktiven 3D-Szene benutzt. Bei der Sortierung wird die Reihenfolge
'der Punkte so ausgwählt, dass die Punkte der Ober- und Unterkante der
'Schichten in PolygonZ im Uhrzeigersinn liegen. Dieses Script
'wird als ein Menü zum Anklicken in einer aktiven 3D-Szene benutzt.

theProject=av.GetProject
theScene=av.GetActiveDoc       'eine aktive Szene für 3D-Zeichnungen
ListofThms=theScene.GetThemes

'Eingabe einer Tabelle für ein Schichtenmodell des Gebietes
av.ShowMsg("Eingabe einer Schicht, um ein Schichtenmodell auszuwählen ...")
aTable=MsgBox.Input("für ein Schichtenmodell des Gebietes",
      "Eingabe einer Tabelle im Projekt", "Schmgebt.dbf")
aVTab = theProject.FindDoc(aTable).GetVTab

'Bestimmung der Anzahl der Schichten im Schichtenmodell des Gebietes
AnzSch=0
for each rec in aVTab
   AnzSch=AnzSch+1
end
IdxSch=AnzSch-1
MsgBox.Report("im Schichtenmodell"++aTable.AsString+":"++AnzSch.AsString,
      "Anzahl der Schichten")

ListofFlds=aVTab.GetFields
aNmAbkFld=MsgBox.ChoiceAsString(ListofFlds, "das die Abkürzung des Namens"
        +NL+"der Schichten enthält",
        "Auswahl eines Feldes im:"++aTable.AsString)
aNamenFld=MsgBox.ChoiceAsString(ListofFlds, "das den Namen der Schichten enthält",
        "Auswahl eines Feldes im:"++aTable.AsString)
aSchmFld=MsgBox.ChoiceAsString(ListofFlds,
       "welches das Schichtenmodell der Schichten enthält",
       "Auswahl eines Feldes im:"++aTable.AsString)

ListofSchNm={}
for each i in 0..IdxSch
  aSchNm=aVTab.ReturnValue(aNamenFld, i)
  ListofSchNm.Add(aSchNm)
end

theScht=MsgBox.ChoiceAsString(ListofSchNm, "um ein PolygonZ für Fläche zu bilden",
     "Auswahl einer Schicht im"++aTable.AsString)

theIdxNr=ListofSchNm.FindByValue(theScht)
aAbk=aVTab.ReturnValue(aNmAbkFld, theIdxNr)
aSchmStr=aVTab.ReturnValue(aSchmFld, theIdxNr)
aSchmThm=theScene.FindTheme(aSchmStr)

MsgBox.Report("Namen der Schicht:"++theScht.AsString+NL+
           "Abkürzung des Namens:"++aAbk+NL+
           "Schitenmodell der Schicht:"++aSchmStr, "Kontrolle")

av.ShowMsg("Eingabe der Felder für Ober- und Unterkante"++
         "im Schichtenmodell der einzelnen Schicht")

aSchmFTab=aSchmThm.GetFTab
```

```
ListofSchmFlds=aSchmFTab.GetFields
aSchmShpFld=MsgBox.ChoiceAsString(ListofSchmFlds,
        "welches die Punkte der Schichten enthält",
        "Auswahl eines Feldes im:"++aSchmStr)

aSchmTOKHFld=MsgBox.ChoiceAsString(ListofSchmFlds,
        "welches die Oberkante der Schichten enthält",
        "Auswahl eines Feldes im:"++aSchmStr)

aSchmBasHFld=MsgBox.ChoiceAsString(ListofSchmFlds,
        "welches die Unterkante der Schichten enthält",
        "Auswahl eines Feldes im:"++aSchmStr)

av.ShowMsg("Auswahl der Punkte in"++aAbk+","
         ++"um PolygonZ für eine Fläche zu bilden ...")

XminStr=MsgBox.Input("der kleinste RW in"
       ++aAbk+NL+"für die Wand",
      "Eingabe der Koordinate", "2558600.00")
Xmin=XminStr.AsNumber
XmaxStr=MsgBox.Input("der größte RW in"
      ++aAbk+NL+"für die Wand",
     "Eingabe der Koordinate", "2582450.00")
Xmax=XmaxStr.AsNumber

YminStr=MsgBox.Input("der kleinste HW in"
      ++aAbk+NL+"für die Wand",
     "Eingabe der Koordinate", "5618450.00")
Ymin=YminStr.AsNumber
YmaxStr=MsgBox.Input("der größte HW in"
      ++aAbk+NL+"für die Wand",
     "Eingabe der Koordinate", "5618450.00")
Ymax=YmaxStr.AsNumber

av.ShowStopButton

AnzFThmRec=0
for each j in aSchmFTab
  AnzFThmRec=AnzFThmRec+1
end
IdxFThmR=AnzFThmRec-1
MsgBox.Info(AnzFThmRec.AsString, "Anzahl der ganzen Punkte")

Ng=0
ListofListofPt={}

for each aI in 0..IdxFThmR
  Ng=Ng+1
  aPt=aSchmFTab.ReturnValue(aSchmShpFld, aI)
  aX=aPt.Getx.SetFormat("d.dd")
  aY=aPt.Gety.SetFormat("d.dd")
  ListofPt={}
  if ((aX >= Xmin) and (aX <= Xmax)) then
    if ((aY >= Ymin) and (aY <= Ymax)) then
      ListofPt.Add(aPt)
      aTokH=aSchmFTab.ReturnValue(aSchmTOKHFld, aI)
      aBasH=aSchmFTab.ReturnValue(aSchmBasHFld, aI)
      ListofPt.Add(aTokH)
      ListofPt.Add(aBasH)
      ListofListofPt.Add(ListofPt)
    end
  end
```

```
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzFThmRec*100)
  if (not more) then
    exit
  end
end

av.ShowMsg("Sortierung der Punkte in"++aAbk
        ++"für die Schichtenoberkante ...")
av.ShowStopButton

'Sortieren der ausgewählten Punkte
ListofSort={"aufwährts", "abwärhrts"}

aRichtTok=MsgBox.ChoiceAsString(ListofSort,
    "die Richtung der Reihenfolge"+NL+"der Schichtenoberkante",
    "Sortierung der Punkte von"++theScht.AsString)

AnzSchmPt=ListofListofPt.Count
IdxSchmPt=AnzSchmPt-1
IdxSchmPt2=IdxSchmPt-1
ListofListofSPt={}
MsgBox.Info(AnzSchmPt.AsString, "Anzahl der ausgewäghlten Punkte")

Ng=0
if (aRichtTok = "aufwährts") then
  for each i in 0..IdxSchmPt2
    Ng=Ng+1
    aListofPt1=ListofListofPt.Get(i)
    aPt1=aListofPt1.Get(0)
    aPtx1=aPt1.Getx
    jIdx=i+1
    for each j in jIdx..IdxSchmPt
      aListofPt2=ListofListofPt.Get(j)
      aPt2=aListofPt2.Get(0)
      aPtx2=aPt2.Getx
      if (aPtx2 < aPtx1) then
        ListofListofPt.Set(j, aListofPt1)
        ListofListofPt.Set(i, aListofPt2)
        aListofPt1=aListofPt2
        aPtx1=aPtx2
      end
    end
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/IdxSchmPt*100)
    if (not more) then
      exit
    end
  end
elseif (aRichtTok = "abwärhrts") then
  for each i in 0..IdxSchmPt2
    Ng=Ng+1
    aListofPt1=ListofListofPt.Get(i)
    aPt1=aListofPt1.Get(0)
    aPtx1=aPt1.Getx
    jIdx=i+1
    for each j in jIdx..IdxSchmPt
      aListofPt2=ListofListofPt.Get(j)
      aPt2=aListofPt2.Get(0)
      aPtx2=aPt2.Getx
      if (aPtx2 > aPtx1) then
        ListofListofPt.Set(j, aListofPt1)
```

```
          ListofListofPt.Set(i, aListofPt2)
          aListofPt1=aListofPt2
          aPtx1=aPtx2
        end
      end
      'Show percentage complete with enabled stop button
      more=av.SetStatus(Ng/IdxSchmPt*100)
      if (not more) then
        exit
      end
    end
end

av.ShowMsg("Bestimmung der 3D-Punkte"
          ++"für die Schichtenoberkante ...")
av.ShowStopButton

ListofPt3D={}
Ng=0

for each i in 0..IdxSchmPt
  Ng=Ng+1
  ListofPt=ListofListofPt.Get(i)
  aPt=ListofPt.Get(0)
  aTokH=ListofPt.Get(1)
  ListofPt3D.Add(aPt@aTokH)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzSchmPt*100)
  if (not more) then
    exit
  end
end

av.ShowMsg("Bestimmung der 3D-Punkte"
          ++"für die Schichtenunterkante ...")
av.ShowStopButton

Qt11 = MsgBox.YesNo("Soll die Unterkante eine bestimmte"
    +NL+"einheitliche Höhe erhalten?",
    "Veränderungsmöglichkeit der Unterkante", true)
if (Qt11) then
  aBHStr = MsgBox.Input("eine bestimmte einheitliche"
        +NL+"Höhe der Unterkante",
        "Eingabe eines bestimmten Wertes", "0.00")
  aBH = aBHStr.AsNumber
end

Ng=0
for each i in 0..IdxSchmPt
  Ng=Ng+1
  GegenIdx=IdxSchmPt-i
  ListofPt=ListofListofPt.Get(GegenIdx)
  aPt=ListofPt.Get(0)
  if (Qt11) then
    aBasH = aBH
  elseif (not Qt11) then
    aBasH = ListofPt.Get(2)
  end
  ListofPt3D.Add(aPt@aBasH)
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzSchmPt*100)
  if (not more) then
```

```
      exit
    end
  end

  ListofListofPt3D={}
  ListofListofPt3D.Add(ListofPt3D)
  thePg=PolygonZ.Make(ListofListofPt3D)

  av.ShowMsg("Speicherung der Fläche als ein Polygon ...")
  av.ShowStopButton

  ListofAW={"ein vorhandenes Thema", "ein neues Thema"}
  AW11=MsgBox.ChoiceAsString(ListofAW, "zur Speicherung von PolygonZ",
    "Auswahl eines PolygonZ-Themas")

  if (AW11 = "ein vorhandenes Thema") then
    ListofPgZ={}
    for each aT in ListofThms
      if (aT.Is(FTheme)) then
        aFTab=aT.GetFTab
        if(aFTab.GetShapeClass.IsSubClassof(PolygonZ)) then
          ListofPgZ.Add(aT)
        end
      end
    end
    thePgZThm=MsgBox.ChoiceAsString(ListofPgZ,
          "zur Speicherung von PolygonZ",
          "Auswahl eines PolygonZ-Themas")
    PgFTab=thePgZThm.GetFTab
    PgShpFld=PgFTab.FindField("Shape")
    PgSIDFld=PgFTab.FindField("PgZ_ID")
    PgNmAbkFld=PgFTab.FindField("Namen_Abk")

    AnzPgZR=0
    for each arec in PgFTab
      AnzPgZR=AnzPgZR+1
    end
    recNr=AnzPgZR

  elseif (AW11 = "ein neues Thema") then
    WDStr=theProject.GetWorkDir.AsString
    fnStr=FileName.Make(WDStr).MakeTmp("Pgnts11","shp")
    fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PolygonZ)")
    if (fName=nil) then exit end
    fName.SetExtension("shp")
    PgFTab=FTab.MakeNew(fName, PolygonZ)
    PgShpFld=PgFTab.FindField("shape")
    PgSIDFld=Field.Make("PgZ_ID", #Field_Short, 2, 0)
    PgNmAbkFld=Field.Make("Namen_Abk", #Field_Char, 6, 0)

    ListofPgFlds={PgSIDFld, PgNmAbkFld}
    PgFTab.AddFields(ListofPgFlds)

    recNr=0
  end

  PgFTab.SetEditable(false)
  PgFTab.SetEditable(true)

  PgFTab.AddRecord
  PgFTab.SetValue(PgShpFld, recNr, thePg)
  PgFTab.SetValue(PgSIDFld, recNr, recNr)
```

```
PgFTab.SetValue(PgNmAbkFld, recNr, aAbk)

PgFTab.SetEditable(false)

if (AW11 = "ein vorhandenes Thema") then
  theTheme=thePgZThm
elseif (AW11 = "ein neues Thema") then
  NewFThm=FTheme.Make(PgFTab)
  theScene.AddTheme(NewFThm)
  theTheme=NewFThm
end

theTheme.UpdateLegend




'wandpt1.ave
'Eine Wand wird für eine offene Stelle in einer 3D-Blockzeichnung
'als PointZ hergestellt. Als Daten werden eine Tabelle des Schichtenmodells
'des gesamten Gebietes und Schichtenmodelle der einzelnen Schichten
'in dem aktiven 3D-Szene benutzt. Dieses Script wird als ein Menü
'zum Anklicken in einer aktiven 3D-Szene benutzt.

theProject=av.GetProject
theScene=av.GetActiveDoc    'eine aktive Szene für 3D-Zeichnungen
ListofThms=theScene.GetThemes

'Eingabe einer Tabelle für ein Schichtenmodell des Gebietes
av.ShowMsg("Eingabe einer Schicht, um ein Schichtenmodell auszuwählen ...")
aTable=MsgBox.Input("für ein Schichtenmodell des Gebietes",
      "Eingabe einer Tabelle im Projekt", "Schmgebt.dbf")
aVTab = theProject.FindDoc(aTable).GetVTab

'Bestimmung der Anzahl der Schichten im Schichtenmodell des Gebietes
AnzSch=0
for each rec in aVTab
   AnzSch=AnzSch+1
end
IdxSch=AnzSch-1
MsgBox.Report("im Schichtenmodell"++aTable.AsString+":"++AnzSch.AsString,
      "Anzahl der Schichten")

ListofFlds=aVTab.GetFields
aNmAbkFld=MsgBox.ChoiceAsString(ListofFlds, "das die Abkürzung des Namens"
       +NL+"der Schichten enthält",
       "Auswahl eines Feldes im:"++aTable.AsString)
aNamenFld=MsgBox.ChoiceAsString(ListofFlds, "das den Namen der Schichten enthält",
       "Auswahl eines Feldes im:"++aTable.AsString)
aSchmFld=MsgBox.ChoiceAsString(ListofFlds,
      "welches das Schichtenmodell der Schichten enthält",
      "Auswahl eines Feldes im:"++aTable.AsString)

ListofSchNm={}
for each i in 0..IdxSch
  aSchNm=aVTab.ReturnValue(aNamenFld, i)
  ListofSchNm.Add(aSchNm)
end

theScht=MsgBox.ChoiceAsString(ListofSchNm, "um neue Punkte für Fläche zu bilden",
     "Auswahl einer Schicht im"++aTable.AsString)
```

```
theIdxNr=ListofSchNm.Find(theScht)
aAbk=aVTab.ReturnValue(aNmAbkFld, theIdxNr)
aSchmStr=aVTab.ReturnValue(aSchmFld, theIdxNr)
aSchmThm=theScene.FindTheme(aSchmStr)

MsgBox.Report("Namen der Schicht:"++theScht.AsString+NL+
            "Abkürzung des Namens:"++aAbk+NL+
            "Schichtenmodell der Schicht:"++aSchmStr,
            "Kontrolle der eingegebenen Daten")

av.ShowMsg("Eingabe der Felder für Ober- und Unterkante"++
          "im Schichtenmodell der einzelnen Schicht")

aSchmFTab=aSchmThm.GetFTab
ListofSchmFlds=aSchmFTab.GetFields
aSchmShpFld=aSchmFTab.FindField("Shape")
aSchmTOKHFld=MsgBox.ChoiceAsString(ListofSchmFlds,
         "welches die Oberkante der Schichten enthält",
         "Auswahl eines Feldes im:"++aSchmStr)

aSchmBasHFld=MsgBox.ChoiceAsString(ListofSchmFlds,
         "welches die Unterkante der Schichten enthält",
         "Auswahl eines Feldes im:"++aSchmStr)

av.ShowMsg("Auswahl der Punkte in"++aAbk+","
          ++"um PointZ für eine Fläche zu bilden ...")

XminStr=MsgBox.Input("der kleinste RW in"
       ++aAbk+NL+"für die Wand",
       "Eingabe der Koordinate", "2579121.12")
Xmin=XminStr.AsNumber
XmaxStr=MsgBox.Input("der größte RW in"
       ++aAbk+NL+"für die Wand",
      "Eingabe der Koordinate", "2582450.00")
Xmax=XmaxStr.AsNumber

YminStr=MsgBox.Input("der kleinste HW in"
       ++aAbk+NL+"für die Wand",
      "Eingabe der Koordinate", "5618450.00")
Ymin=YminStr.AsNumber
YmaxStr=MsgBox.Input("der größte HW in"
       ++aAbk+NL+"für die Wand",
      "Eingabe der Koordinate", "5618450.00")
Ymax=YmaxStr.AsNumber

av.ShowStopButton

AnzFThmRec=0
for each j in aSchmFTab
  AnzFThmRec=AnzFThmRec+1
end
IdxFThmR=AnzFThmRec-1
MsgBox.Info(AnzFThmRec.AsString, "Anzahl der ganzen Punkte")

Qt11 = MsgBox.YesNo("Soll die Unterkante eine bestimmte"
    +NL+"einheitliche Höhe erhalten?",
    "Veränderungsmöglichkeit der Unterkante", true)
if (Qt11) then
  aBHStr = MsgBox.Input("eine bestimmte einheitliche"
        +NL+"Höhe der Unterkante",
        "Eingabe eines bestimmten Wertes", "0.00")
```

```
  aBH = aBHStr.AsNumber
end

Ng=0
ListofListofPt={}

for each aI in 0..IdxFThmR
  Ng=Ng+1
  aPt=aSchmFTab.ReturnValue(aSchmShpFld, aI)
  aX=aPt.Getx.SetFormat("d.dd")
  aY=aPt.Gety.SetFormat("d.dd")
  ListofPt={}
  if ((aX >= Xmin) and (aX <= Xmax)) then
    if ((aY >= Ymin) and (aY <= Ymax)) then
      ListofPt.Add(aPt)
      aTokH=aSchmFTab.ReturnValue(aSchmTOKHFld, aI)
      if (Qt11) then
        aBasH=aBH
      elseif (not Qt11) then
        aBasH=aSchmFTab.ReturnValue(aSchmBasHFld, aI)
      end
      ListofPt.Add(aTokH)
      ListofPt.Add(aBasH)
      ListofListofPt.Add(ListofPt)
    end
  end
  'Show percentage complete with enabled stop button
  more=av.SetStatus(Ng/AnzFThmRec*100)
  if (not more) then
    exit
  end
end


av.ShowMsg("Berechnung der neuen PointZ ...")
av.ShowStopButton

AnzaltPt=ListofListofPt.Count
IdxaltPt=AnzaltPt-1

ListofNPtZ={}
Ng=0
for each aLst in ListofListofPt
   Ng=Ng+1
   aPt=aLst.Get(0)
   aTokH=aLst.Get(1)
   aBasH=aLst.Get(2)
   ListofNPtZ.Add(aPt@aTokH)
   ListofNPtZ.Add(aPt@aBasH)
   aDicke=aTokH-aBasH
   if (aDicke > 0) then
     aTokFlH=aTokH.Floor
     aBasCH=aBasH.Ceiling
     aDiff=aTokFlH-aBasCH
     if (aDiff >= 0) then
       for each aD in 0..aDiff
         aNH=aBasCH+aD
         ListofNPtZ.Add(aPt@aNH)
       end
     end
   end
  'Show percentage complete with enabled stop button
```

```
  more=av.SetStatus(Ng/AnzaltPt*100)
  if (not more) then
    exit
  end
end

av.ShowMsg("Speicherung der Fläche als PointZ ...")

ListofAW={"ein vorhandenes Thema", "ein neues Thema"}
AW11=MsgBox.ChoiceAsString(ListofAW, "zur Speicherung von PointZ",
  "Auswahl eines PointZ-Themas")

if (AW11 = "ein vorhandenes Thema") then
  ListofPtZ={}
  for each aT in ListofThms
    if (aT.Is(FTheme)) then
      aFTab=aT.GetFTab
      if(aFTab.GetShapeClass.IsSubClassof(PointZ)) then
        ListofPtZ.Add(aT)
      end
    end
  end
  thePtZThm=MsgBox.ChoiceAsString(ListofPtZ,
        "zur Speicherung von PoinzZ",
        "Auswahl eines PointZ-Themas")
  PtFTab=thePtZThm.GetFTab
  PtShpFld=PtFTab.FindField("Shape")
  PtSIDFld=PtFTab.FindField("ID")
  PtSHmFld=PtFTab.FindField("Hoehe (m)")
  PtNmAbkFld=PtFTab.FindField("Namen_Abk")

  AnzPtZR=0
  for each arec in PtFTab
    AnzPtZR=AnzPtZR+1
  end
  recNr=AnzPtZR-1

elseif (AW11 = "ein neues Thema") then
  WDStr=theProject.GetWorkDir.AsString
  fnStr=FileName.Make(WDStr).MakeTmp("Ptnts11","shp")
  fName=FileDialog.Put(fnStr, "*.shp", "Output shape File (PointZ)")
  if (fName=nil) then exit end
  fName.SetExtension("shp")
  PtFTab=FTab.MakeNew(fName, PointZ)
  PtShpFld=PtFTab.FindField("shape")
  PtSIDFld=Field.Make("ID", #Field_Short, 6, 0)
  PtSHmFld=Field.Make("Hoehe (m)", #Field_Float, 8, 2)
  PtNmAbkFld=Field.Make("Namen_Abk", #Field_Char, 6, 0)

  ListofPtFlds={PtSIDFld, PtSHmFld, PtNmAbkFld}
  PtFTab.AddFields(ListofPtFlds)

  recNr=-1
end

AnzNPt=ListofNPtZ.Count
IdxNPt=AnzNPt-1

PtFTab.SetEditable(false)
PtFTab.SetEditable(true)
for each aNPt in 0..IdxNPt
   theNPt=ListofNPtZ.Get(aNPt)
```

```
    theZ=theNPt.Getz
    recNr=recNr+1
    PtFTab.AddRecord
    PtFTab.SetValue(PtShpFld, recNr, theNPt)
    PtFTab.SetValue(PtSIDFld, recNr, recNr)
    PtFTab.SetValue(PtSHmFld, recNr, theZ)
    PtFTab.SetValue(PtNmAbkFld, recNr, aAbk)
end
PtFTab.SetEditable(false)
if (AW11 = "ein vorhandenes Thema") then
  theTheme=thePtZThm
elseif (AW11 = "ein neues Thema") then
  NewFThm=FTheme.Make(PtFTab)
  theScene.AddTheme(NewFThm)
  theTheme=NewFThm
end
theTheme.UpdateLegend
```

```
'wdrmod1.ave
'Eine theoretische Windrose wird für eine geneigte, ebene Fläche
'durch eine Modellrechnung berechnet.
'Dieses Script wird als ein Menü zum Anklicken
'in einem aktiven Karten-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc    'ein aktives Karten-View

myScript=theProject.FindScript("wdrmod11")
myScript.SetNumberFormat( "d.dddddddd") ' script default

'Eingabe der Daten, um theoretische Neigungen zu berechnen
'Eingabe der Daten eines Kreises
ListofRWs={"2565000.00", "2574000.00", "2579000.00"}
ListofHWs={"5632000.00", "5639000.00"}
ListofRds={"2000.00000000", "8000.00000000"}
qt1="Eingabe der Daten eines Kreises"

theZRWStr=MsgBox.ChoiceAsString(ListofRWs,
      "Rechtswert des Zentrums im View"++theView.AsString, qt1)
theZHWStr=MsgBox.ChoiceAsString(ListofHWs,
      "Hochwert des Zentrums im View"++theView.AsString, qt1)
theKRDStr=MsgBox.ChoiceAsString(ListofRds,
      "Radius des Kreises im View"++theView.AsString, qt1)

'Veränderungsmöglichkeit
the2RWStr=MsgBox.Input("Rechtswert des Zentrums (einen anderen Wert?)",
       qt1, theZRWStr)
the2HWStr=MsgBox.Input("Hochwert des Zentrums (einen anderen Wert?)",
       qt1, theZHWStr)
the2RDStr=MsgBox.Input("Radius des Kreises (einen anderen Wert?)",
       qt1, theKRDStr)

theZRW=the2RWStr.AsNumber
theZHW=the2HWStr.AsNumber
theKRD=the2RDStr.AsNumber
```

```
'Eingabe der maximalen Neigung einer Fläche zur Berechnung
'der Neigungen in den unterschiedlichen Himmelsrichtungen
aMaxNeigStr=0.02574417.AsString
maxNeigStr=MsgBox.Input("maximale Neigung der Fläche"
        +NL+"(einen anderen Wert?)",
        "Berechnung der Neigungen in allen Himmelsrichtungen",
        aMaxNeigStr)

maxNeig=maxNeigStr.AsNumber
maxRad=maxNeig.AsRadians
amaxCos=maxRad.Cos
Rk=theKRD
A=Rk/amaxCos
ZK2="Eine Ellipse zur Brechnung der Neigungen"
MsgBox.Report("Die Länge der langen Achse:"++A.AsString+NL+
            "Die Länge der kurzen Achse:"++Rk.AsString, ZK2)

B=Rk
ep=(((A*A)-(B*B)).Sqrt)/A
ListofaRe={}
ListofaRe2={}
ListofaNWk={}
ListofaNWk2={}

'Berechnung der Neigungen einer Fläche in allen Himmelsrichtungen
'mit den Polarkoordinaten einer Ellipse
av.ShowMsg("Berechnung der theoretischen Neigungen"++"...")
av.ShowStopButton
Anz=0

for each aW in 0..179
   Anz=aW+1
   aWkRad=aW.AsRadians
   aWkCos=aWkRad.Cos
   aRe=B/((1-((ep*ep)*(aWkCos*aWkCos))).Sqrt)
   ListofaRe.Add(aRe)
   ListofaRe2.Add(aRe)
   aNgWRadCos=Rk/aRe
   aNgWRad=aNgWRadCos.ACos
   aNgWGr=aNgWRad.AsDegrees
   ListofaNWk.Add(aNgWGr)
   ListofaNWk2.Add(aNgWGr)
   if (aw = 0) then
     aRe0=aRe
     aNgWGr0=aNgWGr
   end
   'Show percentage complete with enabled stop button
   more=av.SetStatus(Anz/180*100)
   if (not more) then
     break
   end
end

'Ergänzung der Daten für die andere Hälfte
for each aW in 0..179
   aRe=ListofaRe.Get(aW)
   ListofaRe2.Add(aRe)
   aNWk=ListofaNWk.Get(aW)
   ListofaNWk2.Add(aNWk)
end

ListofaRe2.Add(aRe0)
```

```
ListofaNWk2.Add(aNgWGr0)

MsgBox.ListAsString(ListofaNWk2,
     "Die berechneten Neigungswinkel in Grad", ZK2)

'Herstellung einer Windrose
Frag1=MsgBox.YesNo("Zeigt die maximale Neigung der Fläche"
    +NL+"die Himmelsrichtung von N-S ?",
    "Drehung der berechneten Windrose", false)
if (Frag1) then
  aDRW=0
elseif (Not Frag1) then
  MsgBox.Info("Die Windrose muß gedreht werden.", "Zeichnung der Windrose")
  aDRWStr=MsgBox.Input("der Drehung der Himmelsrichtung"
       +NL+"der maximalen Neigung der Fläche",
       "Eingabe des Betrags (Grad)", "5")
  aDRW=aDRWStr.AsNumber.Abs
  ListofDR={"Gegenuhrzeigersinn", "Uhrzeigersinn"}
  aDR=MsgBox.ChoiceAsString(ListofDR, "Auswahl der Drehrichtung",
    "Drehung der berechneten Windrose")
end

AnzNWk=ListofaNWk2.Count
IdxNWk=AnzNWk-1
ListofNWkKr={}
ListofPt={}

av.ShowMsg("Herstellung der Windrose in einem Kreis"++"...")
av.ShowStopButton
Anz=0

for each aE in 0..IdxNWk
   Anz=aE+1
   aKrIdx=IdxNWk-aE
   aNgWk=ListofaNWk2.Get(aKrIdx)
   ListofNWkKr.Add(aNgWk)
   aKrWk=aE
   if (Not Frag1) then
     if (aDR = "Gegenuhrzeigersinn") then
       aKrWk=aKrWk-aDRW
       if (aKrWk < 0) then
         aKrWk=360+aKrWk
       end
       aHR=360-aDRW
       agHR=aHR-180
     elseif (aDR = "Uhrzeigersinn") then
       aKrWk=aKrWk+aDRW
       if (aKrWk > 360) then
         aKrWk=aKrWk-360
       end
       aHR=aDRW
       agHR=aHR+180
     end
   end
   aKrWkRad=aKrWk.AsRadians
   theNRD=(aNgWk/aNgWGr0)*Rk
   aX=theNRD*(aKrWkRad.Sin)+theZRW
   aY=theNRD*(aKrWkRad.Cos)+theZHW
   ListofPt.Add(aX@aY)
   'Show percentage complete with enabled stop button
   more=av.SetStatus(Anz/AnzNWk*100)
   if (not more) then
```

```
      break
    end
end

MsgBox.ListAsString(ListofNWkKr,
     "Die berechneten Neigungswinkel in Grad", "Eine Elipse (Umgekehrt)")

'Ein Feature-Shape-File für eine Windrose (Polygon) wird hergestellt.
aWD=av.GetProject.GetWorkDir.AsString
fnStr=FileName.Make(aWD).MakeTmp("Wdrntmd1","shp")
fName=FileDialog.Put(fnStr, "*.shp",
      "Output Shape File (für eine Windrose)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PgFTab=FTab.MakeNew(fName, Polygon)
PgIDFld=Field.Make("ID", #FIELD_SHORT, 2, 0)
PgRwFld=Field.Make("Kreisz_RW", #Field_FLOAT, 10, 2)
PgHwFld=Field.Make("Kreisz_HW", #Field_FLOAT, 10, 2)
PgRdFld=Field.Make("Kreis_Rd", #Field_FLOAT, 10, 2)
PgNWkFld=Field.Make("max_Neig",  #Field_FLOAT, 12, 8)
PgHRFld=Field.Make("Himmelsr",  #Field_CHAR, 20, 0)

ListofPgFlds={PgIDFld,PgRwFld,PgHwFld,PgRdFld,PgNWkFld,PgHRFld}
PgFTab.AddFields(ListofPgFlds)
PgShpFld=PgFTab.FindField("shape")

ListofListofPt={}
ListofListofPt.Add(ListofPt)
thePolyg=Polygon.Make(ListofListofPt)
if (Frag1) then
  aHR=360
  agHR=180
elseif (Not Frag1) then
  if (aDR = "Gegenuhrzeigersinn") then
     aHR=360-aDRW
     agHR=aHR-180
  elseif (aDR = "Uhrzeigersinn") then
     aHR=aDRW
     agHR=aHR+180
  end
end
if (aHR < agHR) then
  aHRkl=aHR.SetFormat("").SetFormat("d")
  aHRgr=agHR.SetFormat("").SetFormat("d")
elseif (aHR > agHR) then
  aHRkl=agHR.SetFormat("").SetFormat("d")
  aHRgr=aHR.SetFormat("").SetFormat("d")
end
theHRStr=aHRkl.AsString++"-"++aHRgr.AsString++"(Grad)"

PgFTab.SetEditable(false)
PgFTab.SetEditable(true)

PgFTab.AddRecord
PgFTab.SetValue(PgShpFld, 0, thePolyg)
PgFTab.SetValue(PgIDFld, 0, 0)
PgFTab.SetValue(PgRwFld, 0, theZRW)
PgFTab.SetValue(PgHwFld, 0, theZHW)
PgFTab.SetValue(PgRdFld, 0, theKRD)
PgFTab.SetValue(PgNWkFld, 0, maxNeig)
PgFTab.SetValue(PgHRFld, 0, theHRStr)
```

```
PgFTab.SetEditable(false)
ThmNew=FTheme.Make(PgFTab)
theView.AddTheme(ThmNew)



'windrpgl.ave
'Die auf den Profilen der Terrassen bestimmten Neigungen
'werden im aktiven Karten-View als eine Windrose gezeichnet.
'Dieses Script wird als ein Menü zum Anklicken
'in einem aktiven Karten-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc   'ein aktives Karten-View
myScript=theProject.FindScript("windrpgl")
myScript.SetNumberFormat( "d.dddddddd") ' script default

'Die Neigungen, die auf den Profilen bestimmt worden sind,
'werden von einer Tabelle abgelesen.
ListofDoc=theProject.GetDocs
aDoc=MsgBox.ChoiceAsString(ListofDoc,
   "Eingabe einer Tabelle für bestimmte Neigungen",
   "Auswahl eines Dokumentes im Projekt"++theProject.AsString)

theVTab=aDoc.GetVTab
ListofVTab=theVTab.GetFields
WKFld=MsgBox.ChoiceAsString(ListofVTab,
    "das die Winkel der Himmelsrichtungen enthält",
    "Auswahl eines Feldes in der Tabelle"++aDoc.AsString)

NgFld=MsgBox.ChoiceAsString(ListofVTab,
    "das die Winkel der Neigungen enthält",
    "Auswahl eines Feldes in der Tabelle"++aDoc.AsString)

'Anzahl der Datensätze in der Tabelle
AnzWk=0
for each rec in theVTab
   AnzWk=AnzWk+1
end
IdxWk=AnzWk-1

'Eingabe der Daten eines Kreises
ListofRWs={"2565000.00", "2574000.00", "2579000.00"}
ListofHWs={"5632000.00", "5639000.00"}
ListofRds={"2000.00", "8000.00"}
qt1="Zentrum eines Kreises"

theRWStr=MsgBox.ChoiceAsString(ListofRWs,
      "Rechtswert des Zentrums im View"++theView.AsString, qt1)
theHWStr=MsgBox.ChoiceAsString(ListofHWs,
      "Hochwert des Zentrums im View"++theView.AsString, qt1)
theRDStr=MsgBox.ChoiceAsString(ListofRds,
      "Radius des Kreises im View"++theView.AsString, qt1)

'Veränderungsmöglichkeit
the2RWStr=MsgBox.Input("Rechtswert des Zentrums (einen anderen Wert?)",
       qt1, theRWStr)
the2HWStr=MsgBox.Input("Hochwert des Zentrums (einen anderen Wert?)",
       qt1, theHWStr)
the2RDStr=MsgBox.Input("Radius des Kreises (einen anderen Wert?)",
```

```
        qt1, theRDStr)

theRW=the2RWStr.AsNumber
theHW=the2HWStr.AsNumber
theRD=the2RDStr.AsNumber

'Bestimmung eines Neigungswinkels,
'um die Radien der Windrose zu bestimmen
aNW="0.02574417"
aNWStr=MsgBox.Input("um die Radien der Windrose zu bestimmen",
      "Eingabe eines einheitlichen Winkels", aNW)
aNormWk=aNWStr.AsNumber

ListofPt={}
for each aW in 0..IdxWk
   aKrWk=theVTab.ReturnValue(WKFld, aW)
   aRad=(aKrWk).AsRadians
   aNgW=theVTab.ReturnValue(NgFld, aW)
   theNRD=(aNgW/aNormWk)*theRD
   aX=theNRD*(aRad.Sin)+theRW
   aY=theNRD*(aRad.Cos)+theHW
   ListofPt.Add(aX@aY)
   if (aw = 0) then
     aX0=aX
     aY0=aY
   end

end
ListofPt.Add(aX0@aY0)

'Ein Feature-Shape-File für eine Windrose (Polygon) wird hergestellt.
aWD=av.GetProject.GetWorkDir.AsString
fnStr=FileName.Make(aWD).MakeTmp("Wdrngml1","shp")
fName=FileDialog.Put(fnStr, "*.shp",
      "Output Shape File (für eine Windrose)")
if (fName=nil) then exit end
fName.SetExtension("shp")
PgFTab=FTab.MakeNew(fName, Polygon)
PgIDFld=Field.Make("ID", #FIELD_SHORT, 2, 0)
PgRwFld=Field.Make("Kreisz_RW", #Field_FLOAT, 10, 2)
PgHwFld=Field.Make("Kreisz_HW", #Field_FLOAT, 10, 2)
PgRdFld=Field.Make("Kreis_Rd", #Field_FLOAT, 10, 2)
PgNWkFld=Field.Make("Norm_Wk",  #Field_FLOAT, 12, 8)

ListofPgFlds={PgIDFld,PgRwFld,PgHwFld,PgRdFld,PgNWkFld}
PgFTab.AddFields(ListofPgFlds)
PgShpFld=PgFTab.FindField("shape")

ListofListofPg={}
ListofListofPg.Add(ListofPt)

PgFTab.SetEditable(false)
PgFTab.SetEditable(true)

myPolyg=Polygon.Make(ListofListofPg)
PgFTab.AddRecord
PgFTab.SetValue(PgShpFld, 0, myPolyg)
PgFTab.SetValue(PgIDFld, 0, 0)
PgFTab.SetValue(PgRwFld, 0, theRW)
PgFTab.SetValue(PgHwFld, 0, theHW)
PgFTab.SetValue(PgRdFld, 0, theRd)
PgFTab.SetValue(PgNWkFld, 0, aNormWk)
```

```
PgFTab.SetEditable(false)
ThmNew=FTheme.Make(PgFTab)
theView.AddTheme(ThmNew)



'xlsarcvw.ave
'Ein Punkt-Thema in ArcView wird aus Daten von Excel hergestellt.
'Der Dezimalpunkt in Excel ist ",".
'Dieses Script wird als ein Menü zum Anklicken
'in einem aktiven Karten-View benutzt.

theProject=av.GetProject
theView=av.GetActiveDoc   'ein aktives Karten-View
'myScript=theProject.FindScript("xlsarcvw")
'myScript.SetNumberFormat("d.dd")

EgD=MsgBox.Input("Eingabe der im Hintergrund laufenden Excel-Datei",
 "Dynamic-Data-Exchange", "Ghsfgr11.xls")
theClient=DDEClient.Make("Excel", EgD)

MsgBox.Report("Die erste Spalte in der Exel-Datei: x-Koordinate"+NL+
            "Die zweite Spalte in der Exel-Datei: y-Koordinate",
            "Die notwendige Form der Daten")

Anzstr = MsgBox.Input("in der Excel-Datei"+NL+"inclusiv Überschrift",
                  "Eingabe der Anzahl der Zeilen", "657")
Anzdat = Anzstr.AsNumber
EndIndex = Anzdat - 1

AnzColStr = MsgBox.Input("Anzahl der Spalten",
          "Eingabe der Excel-Daten", "8")
AnzCol = AnzColStr.AsNumber
IdxCol = AnzCol-1

av.ShowMsg("Die Excel-Daten werden gelesen ...")
av.ShowStopButton

ListofDatenArt = {}
ListofUebersch = {}
ListofListofDt = {}
Ng = 0
for each i in 0..EndIndex
   Ng = Ng + 1
   ListofxlsDt = {}
   zNr = i + 1
   zstring = zNr.AsString
   for each j in 0..IdxCol
      sNr = j + 1
      sstring = sNr.AsString
      Daten = theClient.Request("z"+zstring+"s"+sstring)
      RString = Daten.Right(2)
      DC = Daten.Count
      ListofDt = Daten.AsTokens(RString)
      DZ = ListofDt.Count
      if (DZ <> 0) then
         DatenStr = ListofDt.Get(0)
         if (DatenStr.Contains(",")) then
            DCStr = DatenStr.AsTokens(",")
```

```
            DtNr1 = DCStr.Get(0)
            DtNr2 = DCStr.Get(1)
          else
            DtNr1=DatenStr
            DtNr2="00"
          end
          DtNr12 = DtNr1+"."+DtNr2
          Kontr = DtNr12.IsNumber
          if (Kontr) then
              DatenNr=DtNr12.AsNumber.SetFormat("d.dd")
          elseif (Not Kontr) then
              DatenNr=DtNr1
          end
        else
          DatenNr = DZ
          Kontr = true
        end
        ListofxlsDt.Add(DatenNr)
        if (i = 0) then
          ListofUebersch.Add(DatenNr)
        end
        if (i = 1) then
          if (Kontr) then
            ListofDatenArt.Add("N")
          elseif (Not Kontr) then
            ListofDatenArt.Add("Z")
          end
        end
    end
    ListofListofDt.Add(ListofxlsDt)
    'Show percentage complete with enabled stop button
    more=av.SetStatus(Ng/Anzdat*100)
    if (not more) then
      break
    end
end
theClient.Close

av.ShowMsg("Punkte werden aus Excel-Daten in ArcView hergestellt ...")
av.ShowStopButton

'Ein Feature-Shape-File für Punkte wird hergestellt.
aWD=av.GetProject.GetWorkDir.AsString
fnStr=FileName.Make(aWD).MakeTmp("Ghsfgr11","shp")
fnName=FileDialog.Put(fnStr, "*.shp", "Output shape File (Point)")
if (fnName=nil) then exit end
fnName.SetExtension("shp")
PtFTab = FTab.MakeNew(fnName, point)
ShpFld = PtFTab.FindField("shape")

ListofFld = {}

for each i in 0..IdxCol
    IdxStr = i.SetFormat("d").AsString
    aDtArt = ListofDatenArt.Get(i)
    aFldBz = ListofUebersch.Get(i)
    aFldNm = IdxStr
    if (aDtArt = "N") then
      aFldNm = Field.Make(aFldBz, #Field_Float, 10, 2)
    elseif (aDtArt = "Z") then
      aFldNm = Field.Make(aFldBz, #Field_Char, 10, 0)
    end
```

```
    ListofFld.Add(aFldNm)
end
PtFTab.AddFields(ListofFld)

recNr=-1

PtFTab.SetEditable(false)
PtFTab.SetEditable(true)

for each j in 0..EndIndex
  aListofDt = ListofListofDt.Get(j)
  if (j > 0) then
    recNr = recNr + 1
    PtFTab.AddRecord
    for each i in 0..IdxCol
      aDt = aListofDt.Get(i)
      if (i = 0) then
        aX = aDt
      elseif (i = 1) then
        aY = aDt
      end
      aFld = ListofFld.Get(i)
      PtFTab.SetValue(aFld, recNr, aDt)
    end
    aShp = Point.Make(aX, aY)
    PtFTab.SetValue(ShpFld, recNr, aShp)
  end
end

PtFTab.SetEditable(false)
thmNew = FTheme.Make(PtFTab)
theView.AddTheme(thmNew)
```