11. APPENDICES

Appendix 1

Look-up tables for the fault database

Bos2

FAULT_NAME	NAME_ID
	0
Belt of Hills/Gatkop	1
Bobbejaanwater splay	2
Boschpoort	3
Brits	4
Brits Graben	5
Buffelshoek/Belt of Hill	6
Bultfontein	7
Bushveld	8
Crocodile Bridge/Bobbeja	9
Crocodile River	10
Crocodile River fragment	11
Derdepoort	12
Donkerpoort	13
Droogekloof/Sandrivier	14
Elandsberg	15
Dwarsberg	16
Gevonden	17
Groot Marico	18
Johannesburg dome	19
Kwarriehoek	20
Liliput	21
Magaliesberge	22
Mahobieskraal	23
Mankwe	24
Mannyelanong	25
Nieuwpoort	26
Pilaansberg	27
Rietfontein fault system	28
Rooiberg	29
Rooiberg fragment	30
Rosseauspoort Range	31
Ruighoek	32
Rustenburg	33
Sandrivier	34
South parallel/Nieuwpoor	35
Springbok flats	36
Swartruggens	37
Syferfontein	38
Vaalwater	39
Vaalwater fault zone	40
Vlakfontein fault	41
Zeerust	42

Bos3

STRUCTURE	USER_ID
Northern Transvaal	11
Zebediela fault	23
Swaershoek	17
Northern Bushveld	9
Warmbaths	19
Nederland	8
De Hoop	4
Northern Melinda	10
Sunnyside	16
Blouberg	2
Makgabeng	6
Zoetfonein/Melinda	24
Uitkomst	18
Abbotspoort	1
Southern Melinda	15
Melinda	7
Potgietersrus	13
Pruizen	14
Grootvlei	5
Welgevonden	20
Northern Waterberg	12
Zebediela	22
Ysterberg	21

Bos5

NAME	USER_ID
unknown	0
Acre	1
Archaean	2
Barberton Greenstonebelt	3
Eastern Transvaal	4
Penge-Sabie Gravity	5
Groblersdal	6
Jarrabad	7
Laersdrift	8
Loskopdam	9
Magnet Heights	10
Malta	11
Marble Hall	12
Mhlapitsi	13
Murchison Greenstonebelt	14
Northeastern Transvaal	15
Pietersburg Greenstonebelt	16
Sekhukhune	17
Sekhukhuneberge	18
Serala	19
Stavoren	20
Steelpoort	21
Steelpoort extension	22
Strydpoort	23
Welkommyn	24
Wilgerrivier	25
Wolkberg	26
Wonderkop	27
Zebediela	28

USER_ID	DISPLACEMENT
dn	down north
de	down east
ds	down south
dw	down west
dnw	down northwest
dse	down southeast
dsw	down southwest
dnw	down northwest
un	up north
ue	up east
us	up south
uw	up west
une	up northeast
use	up southeast
usw	up southwest
unw	up northwest
ll	left-lateral
rl	right-lateral

USER_ID	FAULT_TYPE
1	normal
2	thrust
3	strike-slip
4	reverse
5	dip-slip
10	fault observed
15	fault inferred
16	inferred normal
17	inferred reverse
18	inferred thrust
19	inferred strike-slip

USER_ID	AGE
500	Pre-Transvaal
400	Post-Tvl/Pre-Bush
300	Post-Bush/Pre-Waterberg
200	Post-Water/Pre-Karoo
150	Post-Pilanesberg/Pre-Karoo
100	Post-Karoo

Look-up Tables for fold database

Bos2

NAME_ID	FOLD_NAME
1	Blaaubank
2	Crocodile River dome
3	Elandsberg
4	Johannesburg dome
5	Kalkfontein - Piet Zyn Kop
6	Kwa-Lobatleng - Ruitjesvlakte
7	Kwarriehoek
8	Loubad
9	Magaliesberg
10	Makoppa dome
11	Marico River
12	Nebo
13	Nylstroom
14	Pilanesberg
15	Rooiberg
16	Rooiberg fragment
17	Rustenburg
18	Swaershoek
19	Waterberg
20	Witkleigat
21	Zwartkloof

Bos3

STRUCTURE	USER_ID
	0
Loubad	1
Swaershoek	7
Nylstroom	3
Pruizen	4
Zebediela	9
Spanje	5
Makapans	2
de Hoop	10
Tweefontein	8
Sterkrivierdam	6

Bos4

NAME	USER_ID
Moltke	11
Malta	8
The Downs	18
Mhlapitsi	10
Serala	15
Wolkberg	20
Acre	1
Welkommyn	19
Schwerin	14
Adriaanskop	2
Katkloof	6
Fortdraai	5
Steelpoort Pericline	17
Magnet Heights	7
Paradys Structure	13
Marble Hall	9
Nebo	12
Dennilton	3
Dwars River	4
Stavoren	16

USER_ID	AXIAL_PLANE
N	north
E	east
S	south
W	west
NE	northeast
SE	southeast
SW	southwest
NW	northwest
u	upright
c	curved

USER_ID	FOLD_TYPE
0	undifferentiate
1	anticline
2	syncline
3	pericline

USER_ID	FOLD_AXIS_
N	north
E	east
S	south
W	west
NE	northeast
SE	southeast
SW	southwest
NW	northwest
NS	doubly plunging NS
EW	doubly plunging EW
NE-SW	doubly plunging NE-SW
NW-SE	doubly plunging NW-SE

Appendix 2

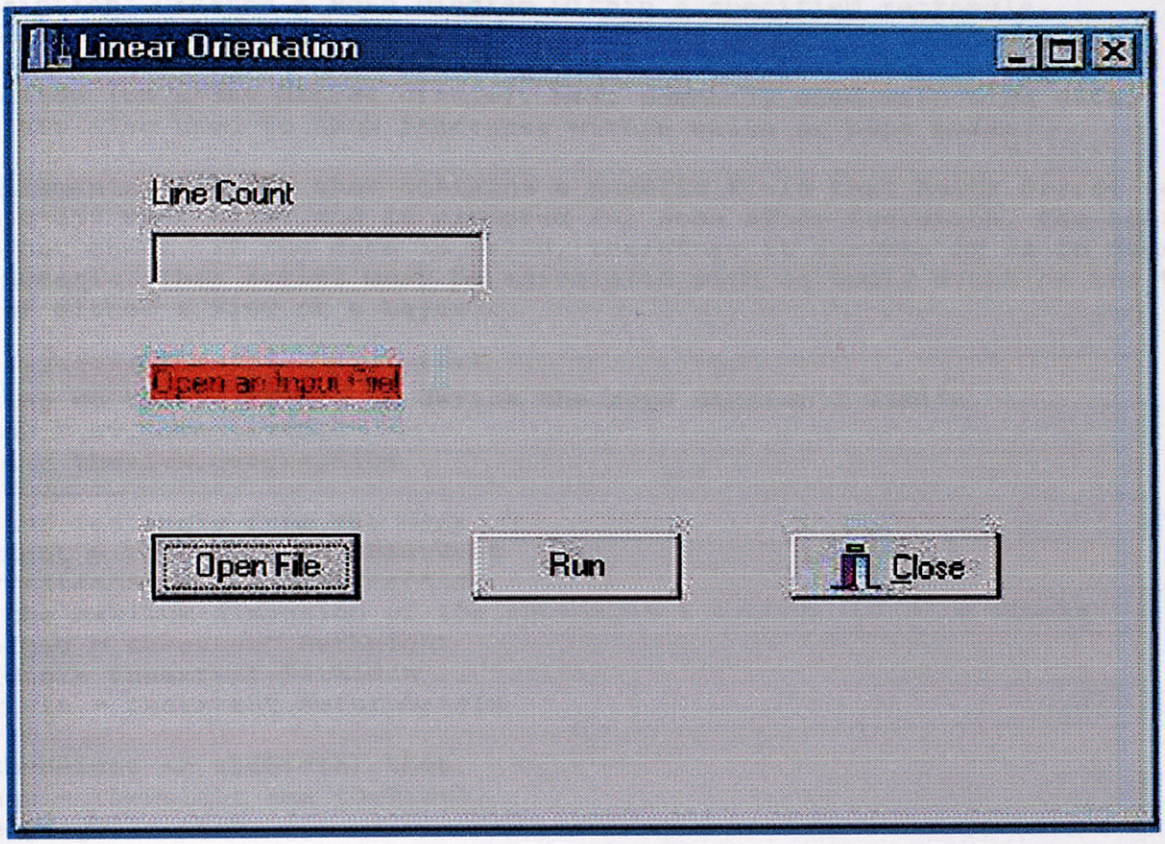
Scripts used in rose diagram production

```
' Name:  View.ShapeToGenerate
'
' Title:  Exports active theme to ARC/INFO export format
'
' Topics:  GeoData
'
' Description:  Exports the active themes to ARC/INFO Export (Generate)
' format files.  Each active theme will require the user to specify an
' output file name.  The format of the export file with be either POINT,
' LINE, or POLYGON depending upon the type of Shape Field found in the
' Theme FTab (the Enum Type FieldEnum).
'
' The user will be prompted for the field from which to create IDs in
' the generate file.  If <None> is selected the ID for the feature will
' be set to the current FTab record number.
'
' In the case of overlapping polygons, the data should be further
' converted into an ARC/INFO region coverage with the REGIONCLASS
' command.  ArcView polygon shapes may need to be modeled in ARC/INFO as
' regions to handle the case of polygons with multiple parts.
'
' Following conversion of the generate file to an ARC/INFO coverage
' the Theme's FTab can be exported to an INFO file that can be joined to
' the coverage Feature Attribute Table (FAT) to restore all attributes as
' found in the original Theme.  Consult ARC/INFO command documentation for
' JOINITEM and related commands.
'
' When creating polygon format Generate files the AUTO keyword is used
' along with the feature ID to indicate that labels will be automatically
' created when coverages are imported using ARC/INFO Generate.  Consult
' the ARC/INFO Generate documentation for additional options and discussion.
'
' Requires:  A View must be the active document.
'
' Self:
'
' Returns:
'


' View should be the Active Document
'
theView = av.GetActiveDoc

' Establish output precision...
'
Script.The.SetNumberFormat( "d.dddddd" )

for each t in theView.GetActiveThemes
  defaultName    = FN.Make("$HOME").MakeTmp(t.GetName.LCase,"gen")
  ungenFileName  = FileDialog.Put( defaultName,"*.gen","Export"++t.GetName )
  if (ungenFileName = nil) then
    exit
  else
    exportFile = LineFile.Make(ungenFileName, #FILE_PERM_WRITE)
```

```
end

' The ARC/INFO generate format uses the ID number of the feature
' in the export file to provide a means for joining attributes.
' When the generate file is converted to an ARC/INFO coverage the ID
' as found in the generate file will be used as the ID in the coverage
' feature attribute table (the .AAT or .PAT).  Allow the user to specify
' the ID field to use or default to the record number....
'
fieldList = t.GetFTab.GetFields.Clone
fieldList.Insert( "<None>" )
idField = MsgBox.ChoiceAsString( fieldList,
  "Select ID field for export file:","Choose ID Field" )
if ( idField = nil ) then
  exit
elseif ( idField = "<None>" ) then
  useID = false
else
  useID = true
end

theFTab     = t.GetFTab
shapeField  = theFTab.FindField( "Shape" )
shapeType   = shapeField.GetType
numRecs     = theFTab.GetNumRecords

av.ShowStopButton
av.ShowMsg( "Exporting"++t.GetName+"..." )

for each recNum in theFTab
  currentShape = theFTab.ReturnValue( shapeField, recNum )

  if ( useID ) then
    id =theFTab.ReturnValueString( idField, recNum )
  else
    id = (recNum + 1).SetFormat("d").AsString
  end

  if (shapeType = #FIELD_SHAPEPOINT) then
    ' ARC/INFO POINT format export file...
    '
    Xvalue = currentShape.GetX.AsString
    Yvalue = currentShape.GetY.AsString
    outputLine = id+", "+Xvalue+", "+Yvalue
    exportFile.WriteElt( outputLine )

  elseif (shapeType = #FIELD_SHAPEMULTIPOINT) then
    ' ARC/INFO POINT format export file...
    '
    pointList = currentShape.AsList
    for each pt in pointList
      Xvalue = pt.GetX.AsString
      Yvalue = pt.GetY.AsString
      outputLine = id+", "+Xvalue+", "+Yvalue
      exportFile.WriteElt( outputLine )
    end

  elseif (shapeType = #FIELD_SHAPELINE) then

    ' ARC/INFO LINE format export file...
    '
```

```
      ' exportFile.WriteElt( id )
      shapeList = currentShape.AsList
      for each shapePart in shapeList
        for each xyPoint in shapePart
          outputLine = id+", "
            +xyPoint.GetY.AsString+", "+xyPoint.GetX.AsString
          exportFile.WriteElt( outputLine )
        end
        exportFile.WriteElt( "END" )
      end

    elseif (shapeType = #FIELD_SHAPEPOLY) then

      ' ARC/INFO POLYGON format export file...
      '
      ' exportFile.WriteElt( id++"AUTO" )  'automatic label generation flag...
      shapeList = currentShape.AsList
      for each shapePart in shapeList
        for each xyPoint in shapePart
          outputLine = id++"AUTO"+", "
            +xyPoint.GetY.AsString+", "+xyPoint.GetX.AsString
          exportFile.WriteElt( outputLine )
        end
        'exportFile.WriteElt( "END" )
      end
    end

    progress = (recNum / numRecs) * 100
    proceed = av.SetStatus( progress )
    if ( proceed.Not ) then
      av.ClearStatus
      av.ShowMsg( "Stopped" )
      exit
    end

  end

  'exportFile.WriteElt( "END" )
  exportFile.Close

  av.ClearStatus
  av.ClearMsg
end
```

Linears.exe



```
'Name: rose.ave
'
'Description: Creates a Rose Diagram within a specified rectangle.
'The diagram is based on the percentage of the number of degrees within
'a specified interval. Rose diagrams are used to show major trends of
'direction (on a 360 degree circle). Most commonly used with wind data,
'they are also used to show fractures within wells or bore holes.
'
'Requirements: A table that contains a numeric field that holds degree values
'from 0-359 must exist and is prompted for soon after execution. The script
'does not check  if the data is valid, therefore it assumes it is in degrees
'and numeric. This script must be associated with an apply event on the tool
'bar of either a View or a Layout.
'
'*********************************
'Setting up the rectangle to define the rose diagram's limits
theView = av.GetActiveDoc
thegra = theview.getgraphics

'get the rectangle from the user
theextent = theView.ReturnUserRect
'*********************************
'Get the maximum dimension of the rectangle & convert it to a square
theHeight = theextent.GetHeight
theWidth = theextent.GetWidth
theOrigin = theextent.ReturnOrigin
'
 if(theHeight <> theWidth) then
   test = theHeight Max theWidth
   newHeight = test
   newOffSet = Point.Make(newHeight,newHeight)
 end
aBox = Rect.Make(theOrigin, newOffSet)
theExtent = aBox
'*********************************
'Rose Programme starts here
'*********************************
' get user stuff
thefn = filedialog.show ("*.*","*.*","Please Select a table.")
if (thefn = NIL) then
exit
else
thevtab = vtab.make (thefn,FALSE,FALSE)
end

thefld = msgbox.choice (thevtab.getfields,"Please Select a trend field.","Trend
Selection")
if (thefld = NIL) then
exit
end
' ***********************************
'Optional routine to use cumulative length instead of frequency to draw rose
petals
'This program section reads the "Length" field in the table

huh = msgbox.yesnocancel ("Use cumulative length?","Length",TRUE)
if (huh = NIL) then
exit
end
 if (huh = TRUE) then
   cum_length = TRUE
```

```
  thelength = msgbox.choice (thevtab.getfields,"Please Select a Length
field.","Length Selection")
 else
  cum_length = FALSE
 end
if (thefld = NIL) then
exit
end
' **********************************
shrink = 0.005
'
huh = msgbox.input ("Enter number of intervals. Max: 30","Intervals","5")
if (huh = NIL) then
exit
end
if (huh.isnumber.not) then
exit
else
deg = huh.asnumber
end
'
' **********************************
'Optional routine to draw break lines within rose
'huh = msgbox.yesnocancel ("Show interval break lines within rose?","Interval
Lines",TRUE)
'if (huh = NIL) then
'exit
'end
'if (huh = TRUE) then
 interval_lines = TRUE
'end
'if (huh = FALSE) then
 'interval_lines = FALSE
 'end
' **********************************
'Optional routine to show diagram labels within rose
'************************************************
'huh = msgbox.yesnocancel ("Show diagram labels?","Circle labels",TRUE)
'if (huh = NIL) then
'exit
'end
'if (huh = TRUE) then
 labels = TRUE
'end
'if (huh = FALSE) then
' labels = FALSE
'end
' ********************************
' some constants (change if nessesary)
txtspc = 0.01  ' text mover to place on the line (for single digits)
txtspc2 = 0.019 ' same as above for double digits
thefont = font.make ("Helvetica","Normal")
textsize = 5
rings = 5
sumdeg = 360 / deg
' ********************************
' initalize dict to all zeros 0..359
dict = {}
for each num in 0..(sumdeg - 1)
dict.add (0.0)
end
```

```
' ***********************************
' break up values in intervals from user
' if a value falls within a interval, it adds to a certain bit in the list
for each rec in thevtab
val = thevtab.returnvaluenumber (thefld,rec)
 if (cum_length = TRUE) then
  len = thevtab.returnvaluenumber (thelength,rec)
 end
mover = deg
tester1 = 0
tester2 = (deg - 0.001).abs
for each num in 0..dict.count
 if ((val >= tester1) AND (val <= (tester2))) then
' ** change this line to add cumulative length **
  if (cum_length = TRUE) then
   dict.set (num,(dict.get(num) + len))
  else
   dict.set (num,(dict.get(num) + 1))
  end
 end
 mover = mover + deg
tester1 = tester2 + 0.001
tester2 = (mover - 0.001)
end
end
'***********************************
' get percentages of values in the list dict (was a dictionary, but lists are
sometimes easier)
newdict = {}
for each num in dict
newdict.add (((num / thevtab.getnumrecords)))
end
' ***********************************
' determine largest value in list
count = 0
for each val in newdict
if (count = 0) then
 val2 = val
 themax = 0
 else
  test = val Max val2
   if (test > themax) then
    themax = test
    end
  end
count = count = 1
val2 = val
end
mulval =  (1 / themax)
fmax = mulval * themax
' ********************************
' multipy all values by milval to scale (largest % becomes the largest line )
testdict = {}
for each val in newdict
testdict.add (mulval * val)
end
newdict = testdict
' ***********************************
' determine user rect characteristics for drawing
theheight =  theextent.getheight
cntr = theextent.returncenter
```

```
inc = ((theheight / 2) / rings) ' - shrink)
' ************************************
' determine scale labels
lablst = {}
'lab_end = themax * 100
lab_end = fmax * 100
lab_range = lab_end / rings
hold = lab_range.asstring.left (3)
lab_range = hold.asnumber
count = lab_range
for each num in 1..rings
if (count.asstring.count > 3) then
hold = count.asstring.left (3)
count = (hold.asnumber + 0.5).round
end
lablst.add (count)
count = count + lab_range
end
' ************************************
' create a white shaded box around the rose diagram for use in a layout or view
etc..
gra = graphicshape.make (theextent)
gra.getsymbol.setstyle (#RASTERFILL_STYLE_SOLID)
gra.getsymbol.setcolor (color.getwhite)
gra.setselected (TRUE)
theview.getgraphics.add (gra)
theview.getgraphics.moveselectedtoback
' ************************************
'circles and labels
cnt = 0
i = inc
for each num in 0..(rings - 1)
if (cnt = 0) then
cir = circle.make (cntr, (inc))
gra = graphicshape.make (cir)
gra.setselected (TRUE)
theview.getgraphics.add (gra)
if (labels = TRUE) then
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx -
txtspc)@((cntr.gety + inc - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx -
txtspc)@((cntr.gety - inc - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx + inc -
txtspc)@((cntr.gety - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx - inc -
txtspc)@((cntr.gety - txtspc)))
gratxt.setselected (TRUE)
```

```
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
end
else
cir = circle.make (cntr, (inc + i))
gra = graphicshape.make (cir)
gra.setselected (TRUE)
theview.getgraphics.add (gra)
if ((lablst.get(num)).asstring.trim.count < 2) then
if (labels = TRUE) then
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx -
txtspc)@((cntr.gety + inc + i - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx -
txtspc)@((cntr.gety - inc - i - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx + inc + i -
txtspc)@((cntr.gety - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx - inc - i -
txtspc)@((cntr.gety - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
end
else
if (labels = TRUE) then
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx -
txtspc2)@((cntr.gety + inc + i - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx -
txtspc2)@((cntr.gety - inc - i - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx + inc + i -
txtspc2)@((cntr.gety - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
theview.getgraphics.add (gratxt)
gratxt = graphictext.make (lablst.get(num).asstring, (cntr.getx - inc - i -
txtspc2)@((cntr.gety - txtspc)))
gratxt.setselected (TRUE)
gratxt.getsymbol.setfont (thefont)
gratxt.getsymbol.setsize (textsize)
```

```
theview.getgraphics.add (gratxt)
end
end
inc = inc + i
end
cnt = cnt + 1
end
lastcir = cir
pntlst = lastcir.asmultipoint.aslist
' ***************************
' adjust the point list from the largest circle to make 0 start at the top
' get different results on ???
count = 0
deglst = {}
for each pnt in pntlst
if (count >= 270) then
deglst.add (pnt)
end
count = count + 1
end
count = 0
for each pnt in pntlst
if (count < 270) then
deglst.add (pnt)
end
count = count + 1
end
newlst = {}
count = 0
for each rec in deglst
newlst.add (count)
count = count + 1
end
'*****************************
' reverse list so 90 is on the right side
count = (deglst.count - 1)
for each pnt in deglst
newlst.set (count,pnt)
count = count - 1
end
deglst = newlst
' ***********************************
' interval lines
count = 0
cnt = deg
linelst = {}
for each pnt in deglst
if (cnt = deg) then
aline = line.make (cntr,pnt)
if (interval_lines = TRUE) then
gra = graphicshape.make (aline)
'gra.getsymbol.setpattern ({5,5})
gra.setselected (TRUE)
theview.getgraphics.add (gra)
end
count = count + 1
cnt = 0
linelst.add (aline)
end
cnt = cnt + 1
```

```
end
thedist = aline.returnlength
' ********************************************
' create petal polygons
count = 0
for each val in newdict
if (val <> 0) then
line1 = linelst.get (count)
if (count >= (linelst.count - 1)) then
line2 = linelst.get(0)
else
line2 = linelst.get (count + 1)
end
center1 = line1.returncenter
center2 = line2.returncenter
diffx1 = ((line1.returnstart.getx - center1.getx)  * 2) * (val)
diffy1 = ((line1.returnstart.gety - center1.gety)  * 2) * (val)
diffx2 = ((line2.returnstart.getx - center2.getx)  * 2) * (val)
diffy2 = ((line2.returnstart.gety - center2.gety)  * 2) * (val)
newx1 = (diffx1 + line1.returnstart.getx )
newy1 = (diffy1 + line1.returnstart.gety )
newx2 = (diffx2 + line2.returnstart.getx )
newy2 = (diffy2 + line2.returnstart.gety )
pntlst = {cntr,newx1@newy1,newx2@newy2,cntr}
apoly = polygon.make ({pntlst})
if (apoly.returnlength >= thedist) then
end
gra = graphicshape.make (apoly)
gra.getsymbol.setstyle (#RASTERFILL_STYLE_SOLID)
if (val = fmax) then
gra.getsymbol.setcolor (color.getred)
else
gra.getsymbol.setcolor (color.getblue)
end
gra.setselected (TRUE)
theview.getgraphics.add (gra)
av.showmsg ("Building petals...")
'av.setstatus ((count / newdict.count) * 100)
end
count = count + 1
end
' ******************************
' places the file name at the origin
'gratxt = graphictext.make (thefn.getbasename,(gra.getorigin.getx +
0.01)@(gra.getorigin.gety + 0.01))
'gratxt.setselected (TRUE)
'gratxt.getsymbol.setfont (thefont)
'gratxt.getsymbol.setsize (textsize + 1)
'theview.getgraphics.add (gratxt)

' ******************************
'Select graphic text and move to front

theview.getgraphics.unselectall
count = thegra.count-1

for each rec in 0..count by 1
theshape = thegra.Get(rec)
if(theshape.Is(GraphicText) = TRUE) then
theshape.SetSelected(TRUE)
end
```

```
end
thegra.MoveSelectedToFront
'thegra.unselectall
' ******************************
theview.getgraphics.selectall
theview.getgraphics.groupselected
theview.getgraphics.unselectall
theview.getgraphics.endbatch
av.clearmsg
av.clearstatus
```