DrawWindow (Window)

Properties

```
f2Transform As folderItem
AH(-1) As Double
APhi(-1) As Double
p As Picture
sutPts(-1) As Point
interPts(-1) As Point
stats(5) As Double
```

DrawWindow.getPhiData:

```
Sub getPhiData()
  Dim Y As Integer
  Dim doStream  as TextInputStream
  Dim CharTemp, LineIn, LineTemp   as String
  Dim I, J, K, LineLen  As Integer
  Dim AllIn( 12000 ) As Double
  Dim moreToDo  As Boolean

  //open the input file
  doStream = f2Transform.OpenAsTextFile

  //parse one line at a time
  J = 1
  while not doStream.EOF
    // get a line
    LineIn = doStream.ReadLine()
    LineIn = LTrim(RTrim(LineIn))
    if len(LineIn) <  5 then
      moreToDo =  False
      // the shortest line is "0,0,0"
    else
      moreToDo =  True

      //another line to parse
      LineTemp =  ""
      for I = 1 to len(LineIn)
        //step through the line
        CharTemp = Mid(LineIn, I,   1)
        if ((CharTemp<> ",") and (CharTemp<> " ")) then
          // then we must be in the midst of an entry
          LineTemp = LineTemp + CharTemp
        else
          // We just finished reading a number
          if (len(LineTemp)> 0) then
            //    values are separated by commas and/or spaces
            //    if you reach one of these, process the number
            AllIn(J) = CDbl(LineTemp)
```

```
          // increment the index count and reset the buffer
          J = J + 1
          LineTemp =  ""
        end if
      end if
    next //I
    if (len(LineTemp)> 0) then
      // catch the last entry on the line
      AllIn(J) = CDbl(LineTemp)
      J = J + 1
      LineTemp =  ""
    end if
  end if
WEND //end parsing
doStream.Close

// put the data into APhi and APhi
For I = 1 To J Step 3
  K = AllIn(I)
  if K>UBound(APhi)  then
    Redim APhi(K)
  end if
  APhi(K) = AllIn(I +  1)
       //stats(2) = Power(Phi)
  if K<> 0 then
    stats( 2) = stats( 2) + APhi(K)*APhi(K)
  end if
  if (K+1)>UBound(APhi)  then
    Redim APhi(K +  1)
  end if
next
End Sub
```

DrawWindow.getHData:

```
Sub getHData()
  Dim Y As Integer
  Dim doStream  as TextInputStream
  Dim CharTemp, LineIn, LineTemp   as String
  Dim I, J, K, LineLen  As Integer
  Dim AllIn( 12000 ) As Double
  Dim moreToDo  As Boolean

  //open the input file
  doStream = f2Transform.OpenAsTextFile

  //parse one line at a time
  J = 1
  while not doStream.EOF
    // get a line
```

```
    LineIn = doStream.ReadLine()
    LineIn = LTrim(RTrim(LineIn))
    if len(LineIn) <  5 then
      moreToDo =  False
      // the shortest line is "0,0,0"
    else
      moreToDo =  True

      //another line to parse
      LineTemp =  ""
      for I = 1 to len(LineIn)
        //step through the line
        CharTemp = Mid(LineIn, I,   1)
        if ((CharTemp<> ",") and (CharTemp<> " ")) then
          // then we must be in the midst of an entry
          LineTemp = LineTemp + CharTemp
        else
          // We just finished reading a number
          if (len(LineTemp)> 0) then
            //    values are separated by commas and/or spaces
            //    if you reach one of these, process the number
            AllIn(J) = CDbl(LineTemp)
            // increment the index count and reset the buffer
            J = J + 1
            LineTemp =  ""
          end if
        end if
      next //I
      if (len(LineTemp)> 0) then
        // catch the last entry on the line
        AllIn(J) = CDbl(LineTemp)
        J = J + 1
        LineTemp =  ""
      end if
    end if
  WEND //end parsing
  doStream.Close

  // put the data into AH and AH
  For I = 1 To J Step 3
    K = AllIn(I)
    if K>UBound(AH)  then
      Redim AH(K)
    end if
    AH(K) = AllIn(I +  1)
         //stats(1) = Power(H)
    if K<> 0 then
      stats( 1) = stats( 1) + AH(K)*AH(K)
```

```
    end if
    if (K+1)>UBound(AH)  then
      Redim AH(K + 1)
    end if
  next
End Sub
```

DrawWindow.fCalcTheta:

```
Function fCalcTheta(angle  As Double) As Double
  Dim Phi As Double
  Dim i, nAmps  As Integer

  nAmps = UBound(APhi)
  Phi = 0.0
  FOR i = 0 TO nAmps
    Phi = Phi + APhi(i) * Sin( angle * i )
  NEXT //i
  return (angle - Phi)
End Function
```

DrawWindow.fCalcH:

```
Function fCalcH(angle  As Double) As Double
  Dim H As Double
  Dim i, nAmps  As Integer

  nAmps = UBound(AH)
  H = 0.0
  FOR i = 0 TO nAmps
    H = H + AH(i) * Cos( angle * i )
  NEXT //i
  return H
End Function
```

DrawWindow.Open:

```
Sub Open()
  p = NewPicture( 1200, 800, 16)
End Sub
```

DrawWindow.OpenBtn.Action:

```
Sub Action()
  Dim f As folderItem
  Dim dlg As openDialog

  dlg = new openDialog
  dlg.Filter =  "*/*"
  dlg.title =  "Find the X-amplitudes (*.ampx)"
  f = dlg.showModal()
  f2Transform = f
  statsFld.Enabled =  True
```

```
  if f<>Nil and f.exists  then // valid file was selected
    DrawBtn.Enabled =  True
    infoLbl.text =  "File to draw: " + dlg.result.Name
    getPhiData()
  else    // no file selected
    // there's nothing to do
  end if

  dlg.title =  "Find the Y-amplitudes (*.ampy)"
  f = dlg.showModal()
  f2Transform = f
  if f<>Nil and f.exists  then // valid file was selected
    DrawBtn.Enabled =  True
    infoLbl.text =  "File to draw: " + dlg.result.Name
    getHData()
    stats( 3) = stats( 1) + stats( 2)
  else    // no file selected
    // there's nothing to do
  end if
  statsFld.Text =  "P(H) = " + Str(stats( 1)) + Chr( 13) +"P(Phi) = " + Str(stats( 2)) + Chr( 13) + "P(All) =
              "+ Str(stats( 3))
End Sub
```

DrawWindow.QuitBtn.Action:

```
Sub Action()
  Quit
End Sub
```

DrawWindow.SutCanvas.Close:

```
Sub Close()
  Quit
End Sub
```

DrawWindow.SutCanvas.Paint:

```
Sub Paint(g As Graphics)
  SutCanvas.Graphics.ForeColor = RGB(   255, 255, 255)
  SutCanvas.Graphics.FillRect   0, 0, Me.Width- 1, Me.Height- 1
  SutCanvas.Graphics.ForeColor = RGB(   0, 0, 0)
  SutCanvas.Graphics.DrawRect   0, 0, Me.Width- 1, Me.Height- 1
End Sub
```

DrawWindow.DrawBtn.Action:

```
Sub Action()
  Dim nPts, winHt, winWid, margin   As Integer
  Dim drawingWid, xZero, yZero   As Integer
  Dim Theta, H  As Double
  Dim PlotTheta, PlotH  As Integer
  Dim lastTheta, lastH  as Double
  Dim scale  As Double
  Dim iPt As Integer
```

```
  Dim angle As Double
  Const Pi=3.14159265358979323846264338327950

  winHt      = SutCanvas.Height
  winWid     = SutCanvas.Width
  margin     =  30
  drawingWid = winWid -   2*margin
  scale      = drawingWid/Pi
  yZero      = winHt\ 2
  xZero      = margin
  nPts =  4096
  Redim sutPts( 0)

  angle=0.0
  lastTheta = xZero + scale * fCalcTheta(angle)
  lastH = yZero - scale * fCalcH(angle)
  sutPts( 0) = new Point(lastTheta, lastH)

  DrawWindow.Title = f2Transform.Name
  SutCanvas.Graphics.PenHeight =   2
  SutCanvas.Graphics.PenWidth =   2
  p.Graphics.PenHeight =  2
  p.Graphics.PenWidth =  2
  lastTheta = xZero + scale * fCalcTheta(angle)
  lastH = yZero - scale * fCalcH(angle)
  For iPt =  1 to nPts
    angle = (Pi * iPt) / nPts

    PlotTheta = xZero + scale * fCalcTheta(angle)
    PlotH = yZero - scale * fCalcH(angle)
    Redim sutPts(iPt)
    sutPts(iPt) =  new Point(PlotTheta, PlotH)
    p.Graphics.DrawLine(lastTheta, lastH, PlotTheta, PlotH)
    SutCanvas.Graphics.DrawLine(lastTheta, lastH, PlotTheta, PlotH)
    lastTheta = PlotTheta
    lastH = PlotH

    infoLbl.Text =  "Drawing point "  + Str(iPt)
    infoLbl.Refresh
  Next //iPt

  infoLbl.Text =  "Done drawing " +Str(nPts)+ " points"
  SaveImgBtn.Enabled =   True
  InterpBtn.Enabled =   True
End Sub
```

DrawWindow.infoLbl.Open:

```
Sub Open()
  Me.Text = Str(Screen( 0).Width)+ ", "+Str(Screen( 0).Height)
End Sub
```

DrawWindow.SaveImgBtn.Action:

```
Sub Action()
  if p.Graphics<> nil then
    if ExportPicture(p)  then
      'if ExportPicture(p) then
    else
      MsgBox "Not saved"
    end if
  else
    MsgBox "Can't... no picture"
  end if
End Sub
```

DrawWindow.InterpBtn.Action:

```
Sub Action()
  Dim thisPoint, lastPoint  As Point
  Dim sutLen As Double
  Dim ptLen(- 1) As Double
  Dim scale As Double
  Dim targetDist As Double
  Dim intLen, lenRatio  As Double
  Dim notBracketed  As Boolean
  Dim margin, drawingWid, yZero, xZero   As Integer
  Dim i, j, x, y, nInPts  As Integer
  Dim nOutPts As Integer
  Const Pi=3.14159265358979323846264338327950

  nInPts = UBound(sutPts)
  Redim ptLen(nInPts)
  nOutPts =  512
  Redim interPts(nOutPts)
  for i = 0 to nOutPts
    interPts(i) =  new Point
  next //i
  thisPoint =  new Point
  lastPoint =  new Point
  margin     =  30
  drawingWid = SutCanvas.Width -   2*margin
  yZero      = SutCanvas.Height\  2
  xZero      = margin
  scale      = drawingWid/Pi
  SutCanvas.Graphics.ForeColor = RGB(   0, 0, 0) ' black
  p.Graphics.ForeColor = RGB(  0, 0, 0) ' black
```

```
ptLen( 0) = 0.0
for i = 1 to nInPts
  ptLen(i) = ptLen(i- 1) + sutPts(i- 1).distanceTo( sutPts(i) )
  sutLen = ptLen(i)
next //i
sutLen = ptLen(nInPts)
stats( 5) = sutLen/Abs(sutPts(nInPts).x - sutPts(   0).x)
infoLbl.Text =  "Suture length = "   + Str(sutLen)
statsFld.Text =  "S.I. = "  + Str(stats( 5))
statsFld.Refresh

interPts( 0).set(sutPts( 0))
SutCanvas.Graphics.FillRect(interPts(   0).x-1, interPts( 0).y-1,3,3)
p.Graphics.FillRect(interPts(  0).x-1, interPts( 0).y-1,3,3)

p.Graphics.ForeColor = RGB(  0, 0, 255) ' blue
SutCanvas.Graphics.ForeColor = RGB(   0, 0, 255) ' blue
i=1
for j = 1 to (nOutPts- 1)
  targetDist = ( sutLen * j ) / nOutPts
  notBracketed =  True
  While notBracketed
    If ( ptLen(i) >= targetDist )   Then
      notBracketed =  False
    else
      i = i + 1
    End If
  Wend  'lastPoint < target < thisPoint
  thisPoint.set(sutPts(i))
  lastPoint.set(sutPts(i- 1))
  lenRatio = (targetDist - ptLen(i-  1))/(ptLen(i) - ptLen(i- 1))
  interPts(j).x = lastPoint.x + lenRatio*(thisPoint.x - lastPoint.x)
  interPts(j).y = lastPoint.y + lenRatio*(thisPoint.y - lastPoint.y)
  intLen = interPts(j).distanceTo(interPts(j-   1))

  infoLbl.Text = Str(j)
  if interPts(j).x<SutCanvas.Graphics.Width    and interPts(j).y<SutCanvas.Graphics.Height    then
    infoLbl.Text = Str(interPts(j).x)+   ", "+Str(interPts(j).y)
    infoLbl.Refresh
    p.Graphics.FillRect(interPts(j).x-  1, interPts(j).y- 1,3,3)
    SutCanvas.Graphics.FillRect(interPts(j).x-   1, interPts(j).y- 1,3,3)
  end if
Next //j

interPts(nOutPts).set(sutPts(nInPts))
SutCanvas.Graphics.FillRect(interPts(nOutPts).x-    1, interPts(nOutPts).y-  1,3,3)
p.Graphics.FillRect(interPts(nOutPts).x-   1, interPts(nOutPts).y-  1,3,3)
SavePtsBtn.Enabled =   True
```

```
End Sub
```

DrawWindow.SavePtsBtn.Action:

```
Sub Action()
  Dim i, nPts  As Integer
  Dim y0 As Double
  Dim f As FolderItem
  Dim fileStream  As TextOutputStream
  Dim fileName  As String
  Dim tempDbl, convert  As Double
  Const Pi=3.14159265358979323846264338327950

  nPts = UBound(interPts)
  convert = Pi / (interPts(nPts).x - interPts(    0).x)
  fileName = Left(f2Transform.Name,   4) + ".512.txt"
  //Open the output file - type is left blanki for text files
  f=GetSaveFolderItem(  "",fileName)
  If f<> Nil Then
    fileStream=f.CreateTextFile
    for i=0 to nPts
      fileStream.Write Str(i)
      fileStream.Write  ", "
      tempDbl = convert * ( interPts(i).x - interPts(    0).x)
      fileStream.Write Str(tempDbl)
      fileStream.Write  ", "
      tempDbl = convert * (interPts(   0).y - interPts(i).y)
      fileStream.Write Str(tempDbl)
      fileStream.WriteLine
    next
    fileStream.Close
  End if
End Sub
```

Menu (Item of Unknown Type)

Point (Class)

Properties

```
x As Double
y As Double
```

Point.Point:

```
Sub Point()

End Sub
```

Point.Point:

```
Sub Point(x As Double, y As Double)
  Self.x = x
  Self.y = y
End Sub
```

Point.add:

```
Function add(pt1 As Point, pt2 As Point) As Point
  dim p As Point
  p = new Point()
  p.x = pt1.x + pt2.x
  p.y = pt1.y + pt2.y
  return p
End Function
```

Point.subtract:

```
Function subtract(pt1 As Point, pt2 As Point) As Point
  dim p As Point
  p = new Point
  p.x = pt1.x - pt2.x
  p.y = pt1.y - pt2.y
  return p
End Function
```

Point.distanceTo:

```
Function distanceTo(pt2 As Point) As Double
  Dim p As Point
  Dim d As Double
  p = subtract( Self, pt2)
  d = Sqrt(p.x*p.x + p.y*p.y)
  return d
End Function
```

Point.toString:

```
Function toString() As String
  return "(" + Str(Self.x) + ", " + Str(Self.y) + ")"
End Function
```

Point.set:

```
Sub set(x As Double, y As Double)
  Self.x = x
  Self.y = y
End Sub
```

Point.add:

```
Function add(P As Point) As Point
  dim p2 As Point
  p2 = new Point()
  p2.x = Self.x + P.x
  p2.y = Self.y + P.y
```

```
  return p2
End Function
```

Point.add:

```
Function add(x As Integer, y As Integer) As Point
  Dim p As Point
  p = new Point(x,y)
  return Self.add(p)
End Function
```

Point.set:

```
Sub set(p As Point)
  Self.x = p.x
  Self.y = p.y
End Sub
```