' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'

' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dim swApp As SldWorks.SldWorks

Dim swModel As SldWorks.ModelDoc2

Dim swSelMgr As SldWorks.SelectionMgr

Dim swEnt As SldWorks.Entity

Dim swDraw As SldWorks.DrawingDoc

Dim swDrawModel As SldWorks.ModelDoc2

Dim swView As SldWorks.View

Dim swSheet As SldWorks.Sheet

Dim swEdge As SldWorks.Edge

Dim swVertex As SldWorks.Vertex

Dim swVertexStart As SldWorks.Vertex

Dim CurveParams As CurveParamData

Dim Compcount As Integer

Dim vComps As Variant

Dim vEdges As Variant

Dim swMathUtil As SldWorks.MathUtility

Dim swSelData As SldWorks.SelectData

Dim swDisplayDimension As SldWorks.DisplayDimension

Dim swModelExt As SldWorks.ModelDocExtension

Dim swComponent As SldWorks.Component2

Dim swDrawingComponent As SldWorks.DrawingComponent

Dim CurveType As swCurveTypes\_e

Dim SelectType As swSelectType\_e

Dim ComponentPath As Variant

Dim x\_val As Double

Dim y\_val As Double

Dim y\_to\_compare As Double

Dim Prev\_x\_vals() As Variant

Dim x\_check As Boolean

Dim array\_length As Integer

Dim coordinate\_max\_store As Object

Dim coordinate\_y\_store As Object

Dim swPt1 As Variant, swPt2 As Variant, swPt3 As Variant, swPt4 As Variant

Dim dwgMathCoordinate As SldWorks.MathPoint

Dim dwgStartMathCoordinate As SldWorks.MathPoint

Dim dwgoutput As SldWorks.MathPoint

Dim dwgStartOutput As SldWorks.MathPoint

Dim dwgCoordinate As Variant

Dim modelCoordinate(2) As Double

Dim startCoordinate As Variant

Dim i As Integer

Dim j As Integer

Dim k As Integer

Dim key As Integer

Dim x\_coor\_index As Integer

Dim y\_coor\_index As Integer

Dim test\_value\_xform As Variant

Dim Part As SldWorks.ModelDoc2

Dim boolstatus As Boolean

Dim status As Long, longwarnings As Long

Dim errors As Long

Dim ViewXForm As MathTransform

Dim end\_x\_vertex As SldWorks.Vertex

Dim end\_y\_vertex As SldWorks.Vertex

Dim x\_right As Boolean

Dim y\_down As Boolean

Dim x\_y\_checked As Boolean

Sub main()

Set swApp = Application.SldWorks

Set swModel = swApp.ActiveDoc

Set swMathUtil = swApp.GetMathUtility

array\_length = 1

ReDim Prev\_x\_vals(1)

Prev\_x\_vals(0) = -100

Set coordinate\_max\_store = CreateObject("Scripting.Dictionary")

Set coordinate\_y\_store = CreateObject("Scripting.Dictionary")

x\_coor\_index = 0

y\_coor\_index = 1

x\_right = True

y\_down = True

x\_y\_checked = False

If Not swModel Is Nothing Then

If Not swModel.GetType = 3 Then

MsgBox "Need to have a drawing open"

Exit Sub

End If

Set swDrawModel = swModel

Set swSelMgr = swModel.SelectionManager

Set swDraw = swModel

Set swView = swDraw.ActiveDrawingView

Set ViewXForm = swView.ModelToViewTransform

test\_value\_xform = swView.GetViewXform

'Set swView = swDrawModel.GetFirstView

'Set swView = swView.GetNextView

Set swSelData = swSelMgr.CreateSelectData

Set swModelExt = swModel.Extension

For k = 1 To swSelMgr.GetSelectedObjectCount

If swSelMgr.GetSelectedObjectType3(k, -1) = swSelectType\_e.swSelVERTICES Then

Set swVertex = swSelMgr.GetSelectedObject6(k, -1)

Set swVertexStart = swSelMgr.GetSelectedObject6(k, -1)

swPt1 = swVertex.GetPoint

End If

Next

'Do While Not swView Is Nothing

Compcount = swView.GetVisibleComponentCount

vComps = swView.GetVisibleComponents

'startCoordinate(0) = swPt1(0)

'startCoordinate(1) = swPt1(1)

'startCoordinate(2) = swPt1(2)

Set dwgStartMathCoordinate = swMathUtil.CreatePoint(swPt1)

Set dwgStartOutput = dwgStartMathCoordinate.IMultiplyTransform(ViewXForm)

startCoordinate = dwgStartOutput.ArrayData

'For i = 0 To UBound(vComps)

'swModel.ClearSelection2 True

'MsgBox "Component name is " & vComps(i).Name2

ComponentPath = Split(vComps(0).Name2, "/")

'MsgBox "Path file name is " & ComponentPath(UBound(ComponentPath))

vEdges = swView.GetVisibleEntities(vComps(0), swViewEntityType\_Edge)

For k = 0 To UBound(vEdges)

Set swEdge = vEdges(k)

Set CurveParams = swEdge.GetCurveParams3

CurveType = CurveParams.CurveType

modelCoordinate(0) = CurveParams.EndPoint(0)

modelCoordinate(1) = CurveParams.EndPoint(1)

modelCoordinate(2) = CurveParams.EndPoint(2)

Set dwgMathCoordinate = swMathUtil.CreatePoint(modelCoordinate)

Set dwgoutput = dwgMathCoordinate.IMultiplyTransform(ViewXForm)

dwgCoordinate = dwgoutput.ArrayData

If CurveType = CIRCLE\_TYPE Then

'MsgBox "the transformed coordinates are " & dwgCoordinate(0) & ", " & dwgCoordinate(1) & ", " & dwgCoordinate(2) & ") The model coordinates are (" & CurveParams.EndPoint(0) & ", " & CurveParams.EndPoint(1) & ", " & CurveParams.EndPoint(2) & ")"

Set swEnt = swEdge

If Not x\_y\_unchecked Then

If startCoordinate(0) > dwgCoordinate(0) Then

x\_right = False

End If

If startCoordinate(1) < dwgCoordinate(1) Then

y\_down = False

End If

x\_y\_unchecked = True

End If

'boolstatus = swEnt.Select4(True, swSelData)

'x\_val = CurveParams.EndPoint(x\_coor\_index)

x\_val = dwgCoordinate(x\_coor\_index)

If IsEmpty(coordinate\_max\_store(x\_val)) Then

'coordinate\_max\_store(x\_val) = Round(CurveParams.EndPoint(y\_coor\_index), 4)

coordinate\_max\_store(x\_val) = Round(dwgCoordinate(y\_coor\_index), 4)

Else

y\_to\_compare = coordinate\_max\_store(x\_val)

If y\_down Then

If y\_to\_compare < Round(dwgCoordinate(y\_coor\_index), 4) Then

coordinate\_max\_store(x\_val) = Round(dwgCoordinate(y\_coor\_index), 4)

End If

Else

If y\_to\_compare > Round(dwgCoordinate(y\_coor\_index), 4) Then

coordinate\_max\_store(x\_val) = Round(dwgCoordinate(y\_coor\_index), 4)

End If

End If

End If

End If

Next

For j = 0 To UBound(vEdges)

Set swEdge = vEdges(j)

Set CurveParams = swEdge.GetCurveParams3

CurveType = CurveParams.CurveType

modelCoordinate(0) = CurveParams.EndPoint(0)

modelCoordinate(1) = CurveParams.EndPoint(1)

modelCoordinate(2) = CurveParams.EndPoint(2)

Set dwgMathCoordinate = swMathUtil.CreatePoint(modelCoordinate)

Set dwgoutput = dwgMathCoordinate.IMultiplyTransform(ViewXForm)

dwgCoordinate = dwgoutput.ArrayData

If CurveType = CIRCLE\_TYPE Then

Set swEnt = swEdge

'x\_val = CurveParams.EndPoint(x\_coor\_index)

x\_val = dwgCoordinate(x\_coor\_index)

x\_check = False

'For Each x\_coor In Prev\_x\_vals

' If Round(x\_val, 4) = Round(x\_coor, 4) Then

' x\_check = True

' End If

'Next

If coordinate\_max\_store(x\_val) = Round(dwgCoordinate(y\_coor\_index), 4) Then

x\_check = False

Else

x\_check = True

End If

If Not x\_check Then

'ReDim Preserve Prev\_x\_vals(array\_length)

'Prev\_x\_vals(array\_length) = x\_val

'array\_length = array\_length + 1

boolstatus = swEnt.Select4(True, swSelData)

'Debug.Assert (boolstatus)

'MsgBox "The Curve is " & CurveParams.CurveType & " endpoint is (" & CurveParams.EndPoint(0) & ", " & CurveParams.EndPoint(1) & ", " & CurveParams.EndPoint(2)

End If

End If

Next

'Next

'MsgBox "visible components in this view are... " & Compcount

errors = swModelExt.AddOrdinateDimension(swAddOrdinateDims\_e.swHorizontalOrdinate, 0.094688827625117, 0.272968021978022, 0)

swDrawModel.ClearSelection2 True

boolstatus = swSelMgr.AddSelectionListObject(swVertexStart, swSelData)

For k = 0 To UBound(vEdges)

Set swEdge = vEdges(k)

Set CurveParams = swEdge.GetCurveParams3

CurveType = CurveParams.CurveType

modelCoordinate(0) = CurveParams.EndPoint(0)

modelCoordinate(1) = CurveParams.EndPoint(1)

modelCoordinate(2) = CurveParams.EndPoint(2)

Set dwgMathCoordinate = swMathUtil.CreatePoint(modelCoordinate)

Set dwgoutput = dwgMathCoordinate.IMultiplyTransform(ViewXForm)

dwgCoordinate = dwgoutput.ArrayData

If CurveType = CIRCLE\_TYPE Then

Set swEnt = swEdge

y\_val = dwgCoordinate(y\_coor\_index)

If IsEmpty(coordinate\_y\_store(y\_val)) Then

coordinate\_y\_store(y\_val) = Round(dwgCoordinate(x\_coor\_index), 4)

Else

y\_to\_compare = coordinate\_y\_store(y\_val)

If x\_right Then

If y\_to\_compare > Round(dwgCoordinate(x\_coor\_index), 4) Then

coordinate\_y\_store(y\_val) = Round(dwgCoordinate(x\_coor\_index), 4)

End If

Else

If y\_to\_compare < Round(dwgCoordinate(x\_coor\_index), 4) Then

coordinate\_y\_store(y\_val) = Round(dwgCoordinate(x\_coor\_index), 4)

End If

End If

End If

End If

Next

For j = 0 To UBound(vEdges)

Set swEdge = vEdges(j)

Set CurveParams = swEdge.GetCurveParams3

CurveType = CurveParams.CurveType

modelCoordinate(0) = CurveParams.EndPoint(0)

modelCoordinate(1) = CurveParams.EndPoint(1)

modelCoordinate(2) = CurveParams.EndPoint(2)

Set dwgMathCoordinate = swMathUtil.CreatePoint(modelCoordinate)

Set dwgoutput = dwgMathCoordinate.IMultiplyTransform(ViewXForm)

dwgCoordinate = dwgoutput.ArrayData

If CurveType = CIRCLE\_TYPE Then

Set swEnt = swEdge

y\_val = dwgCoordinate(y\_coor\_index)

x\_check = False

If coordinate\_y\_store(y\_val) = Round(dwgCoordinate(x\_coor\_index), 4) Then

x\_check = False

Else

x\_check = True

End If

If Not x\_check Then

'ReDim Preserve Prev\_x\_vals(array\_length)

'Prev\_x\_vals(array\_length) = x\_val

'array\_length = array\_length + 1

boolstatus = swEnt.Select4(True, swSelData)

End If

End If

Next

errors = swModelExt.AddOrdinateDimension(swAddOrdinateDims\_e.swVerticalOrdinate, 0.094688827625117, 0.272968021978022, 0)

Set swDisplayDimension = swSelMgr.GetSelectedObject6(1, -1)

swDisplayDimension.SetOrdinateDimensionArrowSize False, 0.00288

'Set swView = swView.GetNextView

'Loop

Else

MsgBox "Need to have a drawing open and active"

Exit Sub

End If

End Sub