

# Feature

## Smart Labels

by Mark Stefanchuk, Cadgurus.com

If you have used label line to add a length and direction label to a line then maybe you have noticed that this command is smart beyond its ability to measure the line. I'm talking about label line's ability to orient the label so that no matter which direction the line is drawn in the length of the line always draws "above" the line. Figure 1 demonstrates this with the small arrows indicating the drawing direction.

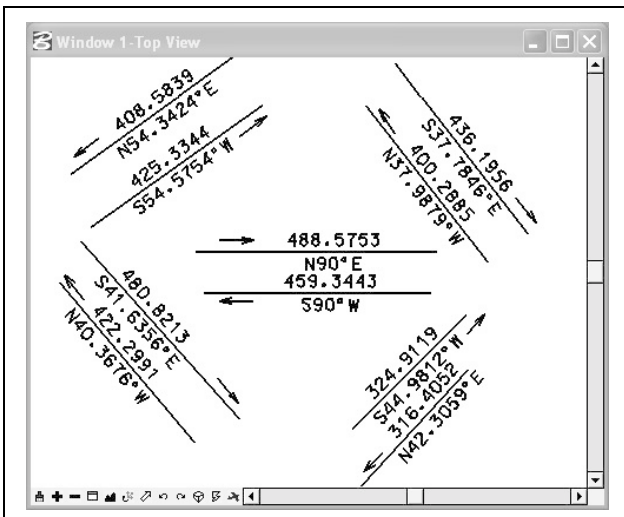


Figure 1 Label Line

To the casual observer defining the rotation of the text may not appear to be difficult. But, let's consider the problem more closely. Let's say that we want to create a command that adds the length of the line at the line's midpoint. We want the text to place dynamically as we rotate about the line origin.

On first analysis we might decide to simply take the rotation of the line and apply it to the text. Let's do that first to see what we get. The following draw routine shows how to draw the text using the angle of the line to define the text rotation.

**TIPS IN HERE**

### Code 1

```
' modified V8 VBA Example Draw Line VB.NET sub in VB.NET all variables are
' passed ByVal by default so when passing points in VB.NET be sure to use ByRef
'
' Also - at the beginning of the class msApp needs to be declared as
' MicroStationDGN.Application.
'
' at the top of the form code add the line
' Implements MicroStationDGN

Private Sub lineDynamics(ByRef Point As Point3d,
    ByVal DrawMode As MicroStationDGN.MsdDrawingMode)
    Dim oEl As LineElement, oText As TextElement
    Dim midPt As Point3d
    Dim angle As Double, txtStr As String
    Dim rMatrix As Matrix3d
```

(Continue)

## Code 1 (Continue)

```
' set the end point of the line to be point m_atPoints is
' is defined globally in the class DrawLine
m_atPoints(1) = Point
oEl = msApp.CreateLineElement1(Nothing, m_atPoints)

' find the angle using the line endpoints
angle = getAngleFromEndPts(m_atPoints(0), m_atPoints(1))

' get the rotation for the text using the z-axis and angle
rMatrix = msApp.Matrix3dFromAxisAndRotationAngle(2, angle)

' find the point in the middle of the line
midPt = oEl.PointAtDistance(oEl.Length / 2)

' format the length property to only show two decimal places
txtStr = Format(oEl.Length, "###0.00")

oText = msApp.CreateTextElement1(Nothing, _
    txtStr, midPt, rMatrix)

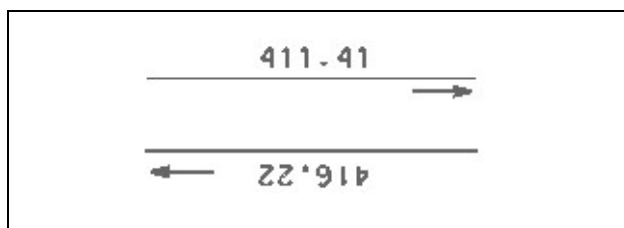
If DrawMode = MsdDrawingMode.msdDrawingModeNormal Then
    msApp.ActiveModelReference.AddElement(oEl)
    msApp.ActiveModelReference.AddElement(oText)
    ' set the start of the line to be the end of previous
    m_atPoints(0) = m_atPoints(1)
End If

oEl.Redraw(DrawMode)
oText.Redraw(DrawMode)
End Sub

' calculate the angle from two points - usually line end points
Public Function getAngleFromEndPts(ByRef pt0 As Point3d, _
    ByRef pt1 As Point3d) As Double
    Dim oppDist, adjDist As Double
    oppDist = pt1.Y - pt0.Y
    adjDist = pt1.X - pt0.X

    ' atan2 takes care of divide by 0
    getAngleFromEndPts = Math.Atan2(oppDist, adjDist)
End Function
```

This almost works. Drawing the line from right to left draws the text above the line the way we had expected. But look what happens when the line is draw from left to right. The text is upside down.



**Figure 2** Draw Line with Label

## Fixing the Problem

As you rotate the line about its origin the text follows the line but as you move into the quadrants defined by  $\pi/2$  to  $\pi$  and from  $\pi$  to  $-\pi/2$  you notice that the text is upside down and unreadable. Here's one way of working around the problem.

Using a few of the vector math tools provided by Bentley we can get the rotation of the line and apply it to the text. But we can also monitor the line orientation specifically the y direction vector to indicate that we need to define the rotation differently.

## Tips

### Finding The Origin

What is the easiest way to find origin in a design file (0,0)?

-----  
The many ways to find the origin of the design file, including:

1) Select the Place Line command and instead of picking a data point for your first point, key-in  $xy=0,0$  and you'll have a line from the origin.

2) Simply, key-in  $XY=$ .

3) Go to *Settings>View Attributes &* switch on ACS triad. Then use Fit All to view all elements in design file. The ACS triad marks the origin.

4) Or use Accudraw Shortcut 'P' and type in 0,0 (in a 2D file) or **0,0,0** (in a 3D file).

**CAD**

The first part of lineDynamics2 code is essentially the same as the previous version. A few more variables have been defined but the line create and point assignments are the same.

```
Private Sub lineDynamics2(ByRef Point As Point3d, _
    ByVal View As MicroStationDGN.View, _
    ByVal DrawMode As MicroStationDGN.MsdDrawingMode)
    Dim oEl, oTmpLine As LineElement, oText As TextElement
    Dim midPt, xVector, yVector, zVector, zeroPt As Point3d
    Dim txtStr As String, actHeight As Double
    Dim rMatrix As Matrix3d
    m_atPoints(1) = Point
    oEl = msApp.CreateLineElement1(Nothing, m_atPoints)

    ' get the midpoint of this line
    midPt = oEl.PointAtDistance(oEl.Length / 2)
```

Here's where the code begins to diverge. The next three lines set the direction vectors for the rotation matrix. There's a couple of ways to visualize this

- 1) Think of each vector as a line drawn from **0,0,0** to some end point. Each vector is just an X,Y,Z coordinate.
- 2 Think of MicroStation's ACS Triad. This is the Red, Green, and White coordinate system lines with its origin at **0,0,0** with each vector point along the positive X,Y, and Z axis.

That's all we have here is a little Triad, but rotated. We want the x-axis of the triad to be along the line. The point3dNormalize will create a unit vector (a vector whose length is 1) along the line. The x, y, and z components will describe a vector whose length is 1. Time to study Pythagoras' and the right hand triangle again.

```
' x axis is along the line
xVector = msApp.Point3dNormalize( _
    msApp.Point3dSubtract(m_atPoints(1), _
    m_atPoints(0)))
```

To get the z vector we use the view rotation. For our level of understanding we can think of view rotation as the ACS Triad, but remember that the view can be rotated. Generally though the zVector will have x, and y components equal to zero and a z component equal to 1. Still with me?

```
' get the zVector from the view
zVector = msApp.Point3dFromMatrix3dRow(View.Rotation, 2)
```

We get the y vector by calculating the cross product between x vector and the z vector. The definition of a cross product between two vectors requires that the direction of the resulting vector (y vector in this case) be perpendicular to both the x vector and the z vector. Take my word for it, this is very convenient.

```
' calculate cross product to get a y axis perpendicular to x and z
yVector = msApp.Point3dCrossProduct(xVector, zVector)
```

Next we will calculate the rotation the text element. To get the rotation we need three points. We have two the line start and end points. To get the third we need a point along the y axis with its origin at one of the end points. In this example I chose to work from the line endpoint. This was arbitrary, I could just as easily chosen to use the startpoint.

```
' create a line from 0,0,0 to yVector
zeroPt.X = 0 : zeroPt.Y = 0 : zeroPt.Z = 0
oTmpLine = msApp.CreateLineElement2(Nothing, zeroPt, yVector)
' put on the end of the line
oTmpLine.Transform(msApp.Transform3dFromPoint3d(m_atPoints(1)))
```

In case you are wondering, we need points to define the x and y axis because our plan view will be in the ihtople view. If we wanted the view to be in a side view we might have chosen to build the rotation from points along the x and z axis.

```
' Create point on Y axis by projecting distance along Y vector.
' Pass this point and the line points to mdlRMatrix_from3Points.
```

This is the reason to use vector math. The y component of the y direction vector will be less than zero whenever the end point of the line is greater than **0**. It doesn't matter what direction the line is drawn (think left to right versus right to left) as is the case using the angle calculation.

```

If yVector.Y < 0 Then
    rMatrix = msApp.Matrix3dRotationFromPoint3dOriginXY( _
        m_atPoints(0), oTmpLine.EndPoint, m_atPoints(1))
    ' make the vector really long so we can project along the
    ' resulting line
    yVector = msApp.Point3dScale(yVector, -1000.0)
Else
    rMatrix = msApp.Matrix3dRotationFromPoint3dOriginXY( _
        m_atPoints(1), m_atPoints(0), oTmpLine.EndPoint)
    yVector = msApp.Point3dScale(yVector, 1000.0)
End If

```

The rest of the code takes care of the offset and draws the line.

```

' offset the text so we can see it and define oTmpLine at midPt
' create a new dummy line so we have something to project along
oTmpLine = msApp.CreateLineElement2(Nothing, zeroPt, yVector)
oTmpLine.Transform(msApp.Transform3dFromPoint3d(midPt))

actHeight = msApp.ActiveSettings.TextStyle.Height
midPt = oTmpLine.PointAtDistance(actHeight)

txtStr = Format(oEl.Length, "###0.00")
oText = msApp.CreateTextElement1(Nothing, txtStr, origin, rMatrix)

If DrawMode = MsdDrawingMode.msdDrawingModeNormal Then
    msApp.ActiveModelReference.AddElement(oEl)
    msApp.ActiveModelReference.AddElement(oText)
    m_atPoints(0) = m_atPoints(1)
End If

oEl.Redraw(DrawMode)
oText.Redraw(DrawMode)
End Sub

```

Figure 3 demonstrates the result of the new code. No matter which direction the lines are drawn the length is always placed above the line.

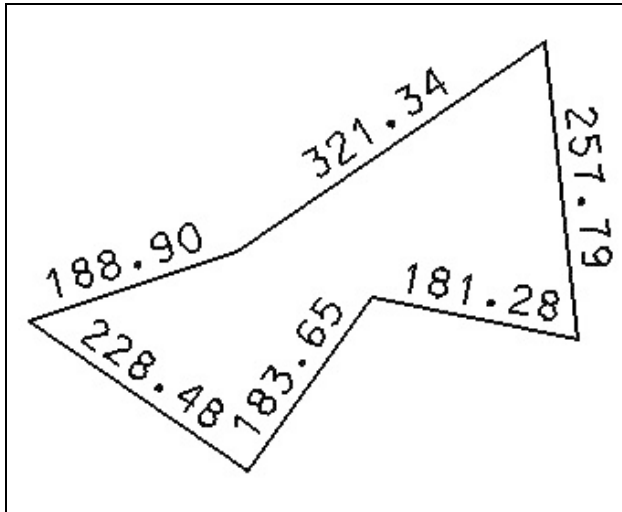


Figure 3 Draw Lines any Direction Label Stays On Top

## Comments Regarding The Examples

My main goal in using VB.Net in these examples was to show you that it could be done. Both examples could be accomplished using MDL/C++, VB6, and MicroStation VBA.

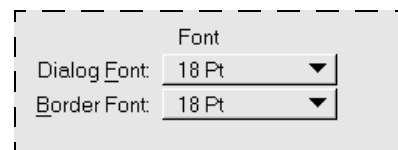
And realize too that it is possible to use more than one toolset to solve a workflow problem. For instance use MDL to solve graphic problems and VB.Net to transfer data from MicroStation to databases.

## Tips

### Dialog Border Font

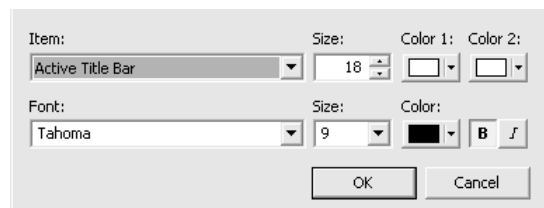
What happened to the *GUI Options>Border Font* settings? They are in V7 preferences box. However, under the *Look and Feel* option of the V8 preferences box, they are missing.

MicroStation now looks at the Windows settings that you get when you change *Control Panel>Display>Appearance>Advanced>ActiveTitleBar>Font*. Adjust to desire sizes.



This option is not available inside v8.

Settings in the Control Panel for change dialog font size in v8.



CAD

If you choose to implement VB.Net there are only a few things you need to do.

1) **Add a reference to your solution** - right click on the project name in the solution explorer window and choose Add Reference. In the dialog that displays click on the COM tab. Scroll down to find the Bentley V8 Object Model. Highlight it and then click on the Select button. Click **Ok**.

2) **At the top of your Form.vb code add the following line:**

Imports MicroStationDGN

3) **At the top of you class module add the following declaration:**

Dim msApp as MicroStationDGN.Application

That's it. Now you can reference MicroStation's object model to draw lines, scan file, or edit tags all from within VB.Net. But, if you're not ready for .Net that's OK because MicroStation VBA, or VB6 will allow you to build the same technology.

There are many possible applications of this technology. Labeling lines is just one example. You could use these same tools to automatically align cells to lines like windows or doors, draw labels perpendicular to lines to place elevations on contours, or define multiple place points offset from the line as might be the case with electrical outlets.

### About The Author

*Mark Stefanchuk is a partner with Ramsey Systems, Inc., the developers of cadgurus.com. Mark can be contacted by email on mark@cadgurus.com.*

*Please email Mark with any feedback or suggestions for future articles.*

**CAD**

If you have MicroStation tips,  
please email details to  
penbrush@ozemail.com.au

### Tips

## Exporting shape data into Excel

Is there a way to get data from a shape into Excel, for example the area?

-----  
You can export data from a shape if you make use of the Measure Area tool. If you turn **On** the 'Mass Properties' checkbox, select the shape, this will display various data from the shape, 'Perimeter', 'Surface Area' etc.

Save this information as a text file using the 'File' - 'Save...' option from the menu of the Measure Area tool. Open with Excel. You will need to go through Excels format wizard to get it in but it works.

**CAD**

### Did You Know?

## Level Creation Disabled

When i copy elements from a reference file they are placed on default level, why? also when i do an element information on them the level is displayed as default? But the level is identified correctly in level manager?

-----  
we are working in v7 workmode. This happens when the level creation capability is disabled.

**CAD**

## TIPS IN HERE



**MicroStation Training --- Software Development :: MDL MACROS VB --- Your CAD Gurus**