Related Applications or Terminology

- **Corner rounding**

Overview

The MintMT `VECTORR/VECTORA` keywords provide interpolated motion on two or more axes and the `CIRCLER/CIRCLEA` keywords provide circular interpolated motion on two orthogonal axes, for example an X-Y table.

In interpolated moves, the axis speed and acceleration refer not to individual axis speed, but to the path along which motion is in progress. Interpolated moves are positional moves and so use a trapezoidal motion profile.

The MintMT `CONTOURMODE` keyword allows a series of interpolated moves to be blended together to give a continuous path velocity. When contouring is turned on, consecutive moves are blended together so that the axes do not decelerate to a stop at the vector boundaries. `CONTOURMODE` accepts the sum of a bit pattern. Bit 0 has overall control of when contouring is applied. The contour mode can be set for each move as it is loaded into the buffer.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit dec value</th>
<th>Constant(s)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>_c tmCONTOUR_OFF _c tmCONTOUR_ON</td>
<td>Turns contouring on (1) or off (0).</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>_c tmVECTOR_ANGLE_ON</td>
<td>Enable inter-vector angle control</td>
</tr>
</tbody>
</table>

The diagram below illustrates an example pair of linearly interpolated moves in the X/Y plane:

![Diagram of linearly interpolated moves](image-url)
Assuming that the two axes X,Y (mapped to controller axes 0 and 1) started at position 0,0 and contouring is required to ensure constant motion, then these two moves could have been coded in MintMT as...

```plaintext
' Set vectored speed...
SPEED([0,1]) = 100,100
ACCEL([0,1]) = 600,600
DECEL([0,1]) = 600,600

' Set contouring mode...
CONTOURMODE([0,1]) = _ctmCONTOUR_ON

' Load vectors and start move
VECTORR(0,1) = 100,150
VECTORR(0,1) = 10,-70
GO(0,1)
```

This would result in velocity profiles for the two axes as shown below:

It can be clearly seen that, as the second vector is implemented, the speeds of the axes are modified instantaneously (i.e. step changes in velocity). This can result in audible noise from the drive mechanisms and potentially lead to early mechanical problems or failures.

`CONTOURMODE` also allows inter-vector angle control to be turned on (bit 1). If the angle between two consecutive vectors exceeds a threshold set with `CONTOURPARAMETER.axis.0`, then the axes will decelerate at that point. This would produce a single stop point in the velocity profile. This
way, it is possible to specify which inter-vector boundaries result in a stop, which must not stop, and which use inter-vector angle control. However, this not only increases the total move time, but also can lead to application problems resulting from non-constant velocity.

A good solution to this problem, if the application allows, is to fillet the two vectors. This process allows a circle type move of a specified radius to be inserted between the two vectors.

Although the exact corner point may not be reached, the time to complete the move will be decreased and the step changes in velocity will be eliminated (as shown by the velocity traces below):
The cyan and magenta lines show the original move (without any filleting of the corner) and the dark blue and green lines show the velocity profile with a 10-unit fillet applied.

To apply a fillet to a 2D sequence of vectored moves we need to calculate several important values:

- Resulting change in vector lengths
- Centre point of partial circle or arc (relative to end of first vector)
- Angular increment for arc (in degrees)

Once these values are known the original vector displacements may be modified and a MintMT CIRCLER command can be inserted between the two original moves via a MintMT user function.

**Geometric Analysis**

Consider the following diagram that indicates how an arc of a specified radius may be used to ‘blend’ two vectors together:

The original relative vector move is from point A to point B followed by a move from point B to point C. To avoid step changes in velocity while these moves are contoured we apply a fillet of radius R to the moves. This results in three elements to the total move:

- A relative vector from point A to point D
- A circle move of angle $\theta$ with center point at F (point D to point E)
- A relative vector from point E to point C
We therefore need to calculate the vector changes DB and BE, the relative position of point F (with respect to point D) and angle θ.

\[
\alpha = \text{atan2} \left( \frac{y_B - y_A}{x_B - x_A} \right)
\]
\[
\beta = \text{atan2} \left( \frac{y_C - y_B}{x_C - x_B} \right)
\]
\[
\Theta = \alpha - \beta
\]
\[
|AB| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}
\]
\[
|BC| = \sqrt{(x_C - x_B)^2 + (y_C - y_B)^2}
\]
\[
|DB| = |BE| = R * \tan \left( \frac{\Theta}{2} \right)
\]

Therefore, taking into consideration the quadrant of the angular change Θ:

\[
x_D = x_B - \left( \text{sgn} \left( \Theta \right) \times \left( |DB| \times \cos \alpha \right) \right)
\]
\[
y_D = y_B - \left( \text{sgn} \left( \Theta \right) \times \left( |DB| \times \sin \alpha \right) \right)
\]
\[
x_E = x_B + \left( \text{sgn} \left( \Theta \right) \times \left( |BE| \times \cos \beta \right) \right)
\]
\[
y_E = y_B + \left( \text{sgn} \left( \Theta \right) \times \left( |BE| \times \sin \beta \right) \right)
\]
\[
X_F = x_D + \left( \text{sgn} \left( \Theta \right) \times R \times \sin \alpha \right)
\]
\[
Y_F = y_D - \left( \text{sgn} \left( \Theta \right) \times R \times \cos \alpha \right)
\]

There are some important considerations when deriving a fillet solution:

1. Vector lengths AB and BC need to exceed a minimum size (if either are zero then there will be no solution) – this can be tested by our MintMT function
2. If ( |DB| > |AB| ) or ( |BE| > |BC| ) then there is no solution
3. Angle Θ must be restricted to the range +/- 180 degrees

Having derived generalized formulae for a fillet of radius R we can now construct our MintMT user function.
MintMT User Function

Our MintMT function will accept x and y displacements for each of the original two relative vector moves. These will be passed **By Reference** so that modified values to account for the fillet can be passed back to the calling code (if there is no solution then the original values remain unchanged).

The required fillet radius will also be passed as a parameter.

Our function will also return (again via ‘By Reference’ parameters) the relative displacement of the center point of the arc and its angle (to allow a MintMT CIRCLER command to be constructed).

The overall function return is an integer value (True or False) to indicate to the calling code whether a solution was found for the specified vectors and radius.
'--------------------------------------------------------------
' doFillet
'
' A MintMT function to fillet 2D vectors at a specified radius
'
' Argument list
' fDeltaX1 - change in first axis for first vector
' fDeltaY1 - change in second axis for first vector
' fDeltaX2 - change in first axis for second vector
' fDeltaY2 - change in second axis for second vector
' fRadius - required fillet radius
' fAngle - calculated (circle) angle for fillet
' fCentreX - relative displacement of circle from end of first vector
' fCentreY - relative displacement of circle from end of first vector
'
' Return value
' True or False depending on whether there is a solution
'--------------------------------------------------------------

Function doFillet (fDeltaX1, fDeltaY1, fDeltaX2, fDeltaY2, fRadius, fAngle, fCentreX, fCentreY) As Integer

    Const _fSmallestRad As Float = 0.1
    Const _fSmallestVec As Float = 0.1
    Dim fAlpha As Float
    Dim fBeta As Float
    Dim fVectorCrop As Float

    ' Check whether a solution is immediately possible...
    If (fRadius < _fSmallestRad) Or ((Abs(fDeltaX1) < _fSmallestVec) And _
        (Abs(fDeltaY1) < _fSmallestVec)) Or ((Abs(fDeltaX2) < _fSmallestVec) And _
        (Abs(fDeltaY2) < _fSmallestVec)) Then
        doFillet = _false
        Exit Function
    End If

    ' Calculate angular displacements
    fAlpha = Atan2(fDeltaY1, fDeltaX1)
    fBeta = Atan2(fDeltaY2, fDeltaX2)

    ' Normalise to +/- 180 degrees
    fAngle = fAlpha - fBeta
    If fAngle < -180 Then
        fAngle = fAngle + 360
    Else If fAngle > 180 Then
        fAngle = fAngle - 360
    End If
' Calculate modified vectors and check fillet radius is possible...

\[
f_{\text{VectorCrop}} = f_{\text{Radius}} \times \tan\left(\frac{f_{\text{Angle}}}{2}\right)
\]

\[
\text{If } (\text{Abs}(f_{\Delta X1} - (f_{\text{VectorCrop}} \times \text{Sgn}(f_{\text{Angle}}) \times \cos(f_{\text{Alpha}}))) > \_
\text{Abs}(f_{\Delta X1})) \text{ Or } (\text{Abs}(f_{\Delta Y1} - (f_{\text{VectorCrop}} \times \text{Sgn}(f_{\text{Angle}}) \times \sin(f_{\text{Alpha}}))) > \_
\text{Abs}(f_{\Delta Y1})) \text{ Or } (\text{Abs}(f_{\Delta X2} - (f_{\text{VectorCrop}} \times \text{Sgn}(f_{\text{Angle}}) \times \cos(f_{\text{Beta}}))) > \_
\text{Abs}(f_{\Delta X2})) \text{ Or } (\text{Abs}(f_{\Delta Y2} - (f_{\text{VectorCrop}} \times \text{Sgn}(f_{\text{Angle}}) \times \sin(f_{\text{Beta}}))) > \_
\text{Abs}(f_{\Delta Y2}) ) \text{ Then}
\]

' Fillet radius extends vectors beyond original bounds

\[
\text{doFillett} = _{false}
\]

\[
\text{Exit Function}
\]

' Fillet radius is valid

\[
f_{\Delta X1} = f_{\Delta X1} - (f_{\text{VectorCrop}} \times \text{Sgn}(f_{\text{Angle}}) \times \cos(f_{\text{Alpha}}))
\]

\[
f_{\Delta Y1} = f_{\Delta Y1} - (f_{\text{VectorCrop}} \times \text{Sgn}(f_{\text{Angle}}) \times \sin(f_{\text{Alpha}}))
\]

\[
f_{\Delta X2} = f_{\Delta X2} - (f_{\text{VectorCrop}} \times \text{Sgn}(f_{\text{Angle}}) \times \cos(f_{\text{Beta}}))
\]

\[
f_{\Delta Y2} = f_{\Delta Y2} - (f_{\text{VectorCrop}} \times \text{Sgn}(f_{\text{Angle}}) \times \sin(f_{\text{Beta}}))
\]

' Calculate circle relative centre

\[
f_{\text{CentreX}} = f_{\text{Radius}} \times \text{Sgn}(f_{\text{Angle}}) \times \sin(f_{\text{Alpha}})
\]

\[
f_{\text{CentreY}} = -f_{\text{Radius}} \times \text{Sgn}(f_{\text{Angle}}) \times \cos(f_{\text{Alpha}})
\]

' Ensure angle direction suits MintMT

\[
f_{\text{Angle}} = -f_{\text{Angle}}
\]

\[
\text{doFillett} = _{true}
\]

End Function

Example Usage

The following code snippet illustrates how this function may be used:

\[
\text{Dim } fX1,fX2,fY1,fY2,fFillet,fCircAngle,fCircX,fCircY
\]

\[
\text{Dim } f\text{Return As Integer}
\]

\[
\text{CONTOURMODE}([0,1]) = _{\text{ctmCONTOUR\_ON}};
\]

\[
\text{MOVEBUFFERSIZE}([0,1]) = 3;
\]

\[
\text{RESET}([0,1])
\]

\[
f_{\text{Return}} = \text{doFillett}(fX1,fY1,fX2,fY2,fFillet,fCircAngle,fCircX,fCircY)
\]

\[
\text{VECTORR}(0,1) = fX1,fY1
\]

' Only add the arc if a solution was found...

\[
\text{If } f_{\text{Return}} \text{ Then } \text{CIRCLE}(0,1) = fCircX,fCircY,fCircAngle
\]

\[
\text{VECTORR}(0,1) = fX2,fY2
\]

\[
\text{GO}(0,1)
\]