DrawArc

DrawArc [**/W=*winName*/X/Y**] ***xOrg*, *yOrg*, *arcRadius*, *startAngle*, *stopAngle***

The DrawArc operation draws a circular counterclockwise arc with center at *xOrg* and *yOrg*.

Parameters

(*xOrg*, *yOrg*) defines the center point for the arc in the currently active coordinate system.

Angles are measured in degrees increasing in a counterclockwise direction. The *startAngle* specifies the starting angle for the arc and *stopAngle* specifies the end. If *stopAngle* is equal to *startAngle*, 360° is added to *stopAngle*. Thus, a circle can be drawn using *startAngle* = *stopAngle*.

The *arcRadius* is the radial distance measured in points from (*xOrg*, *yOrg*).

Flags

/W=*winName*	Draws to the named window or subwindow. When omitted, action will affect the active window or subwindow. This must be the first flag specified when used in a Proc or Macro or on the command line. When identifying a subwindow with *winName*, see **Subwindow Syntax** on page III-92 for details on forming the window hierarchy.
/X	Measures *arcRadius* using the current X-coordinate system. If /Y is also used, the arc may be elliptical.
/Y	Measures *arcRadius* using the current Y-coordinate system. If /X is also used, the arc may be elliptical.

Details

Arcs honor the current dash pattern and arrowhead setting in the same way as polygons and Beziers. In fact, arcs are implemented using Bezier curves.

Normally, you would create arcs programmatically. If you need to sketch an arc-like object, you should probably use a Bezier curve because it is more flexible and easier to adjust. However, there is one handy feature of arcs that make them useful for manual drawing: the origin can be in any of the supported coordinate systems and the radius is in points.

To draw an arc interactively, see **Arcs and Circles** on page III-64 for instructions.

See Also

Chapter III-3, **Drawing**.

The **SetDrawEnv** and **SetDrawLayer** operations.

The **DrawBezier**, **DrawOval** and **DrawAction** operations.

DrawBezier

DrawBezier [**/W=*winName* /ABS**] ***xOrg*, *yOrg*, *hScaling*, *vScaling*, {*x*$_0$,*y*$_0$,*x*$_1$,*y*$_1$** ...**}**

DrawBezier [**/W=*winName* /ABS**] ***xOrg*, *yOrg*, *hScaling*, *vScaling*, *xWaveName*, *yWaveName***

DrawBezier/A [**/W=*winName***] **{*x*$_n$, *y*$_n$, *x*$_{n+1}$, *y*$_{n+1}$** ...**}**

The DrawBezier operation draws a Bezier curve with first point of the curve positioned at *xOrg* and *yOrg*.

Parameters

(*xOrg*, *yOrg*) defines the starting point for the Bezier curve in the currently active coordinate system.

hScaling and *vScaling* set the horizontal and vertical scale factors about the origin, with 1 meaning 100%.

The *xWaveName*, *yWaveName* version of DrawBezier gets data from the named X and Y waves. This connection is maintained so that any changes to either wave will result in updates to the Bezier curve.

To use the version of DrawBezier that takes a literal list of vertices, you place as many vertices as you like on the first line and then use as many /A versions as necessary to define all the vertices.