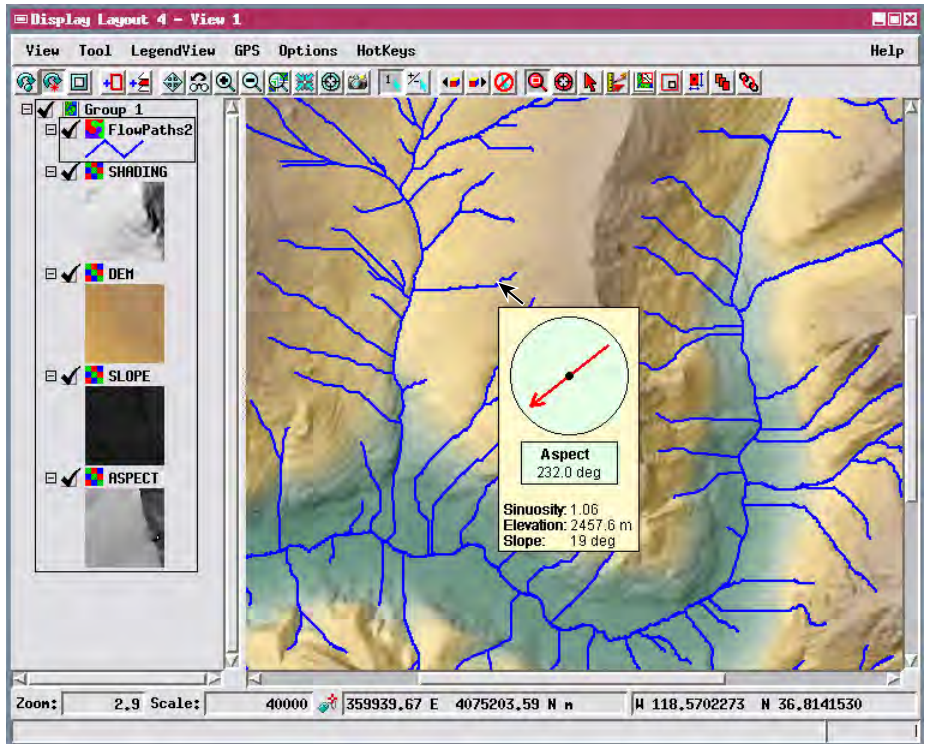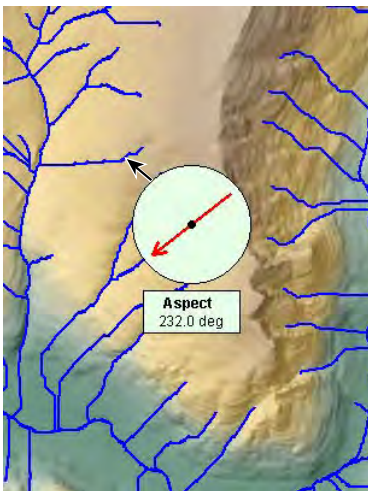# Using Graphics in Complex DataTips

Display control scripts allow you to add custom text and rich, context-sensitive graphical information to DataTips in groups, layouts, and atlases. A control script can read values from the raster cells and geometric elements at the cursor location, perform calculations if needed, and create charts, graphs, or other specialized graphics for presentation in the DataTip. You can configure the script to use the graphic alone (with transparent background) or embed the graphic in the default rectangular DataTip with background color and border. The script can also either allow or suppress the DataTip information that has been set outside the script for the various layers in the view.

The illustrations on this page show these DataTip context options for a sample control script graphic. Each of the three sample control scripts reads the cell value under the mouse cursor from an ASPECT raster layer, creates a graphic with a pointer showing this aspect direction, and creates formatted text to report the numerical value. AspectTip1.sml shows only the aspect graphic with no DataTip background.
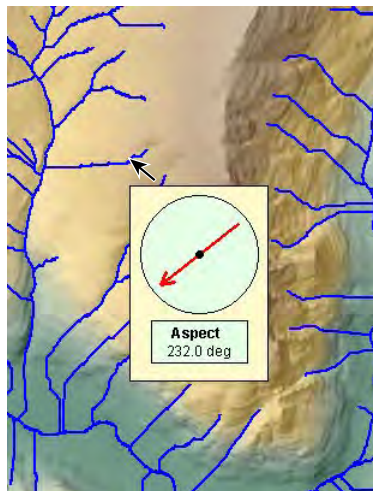
AspectTip2.sml embeds the graphic in the normal colored DataTip background and border, but shows no information from other layers in the layout. AspectTip3.sml embeds the graphic in the normal DataTip background but also allows attribute informa-

AspectTip3.sml: Display layout with complex DataTip incorporating a graphic created by this Display Control Script. The graphic is created from the cell value read from the ASPECT raster object. The script adds this graphic to the DataTip information drawn from the SLOPE, DEM, and FlowPaths2 layers.

AspectTip1.sml: Display Control Script creates DataTip with aspect graphic only; no DataTip background or border.

AspectTip2.sml: Display Control Script creates DataTip with aspect graphic and default background color and border.

tion from other layers in the layout to appear (as designated by the settings on the DataTip panel of the Layer Controls window for each layer; see the Technical Guide entitled *Spatial Display: Designing Complex DataTips*).

The bulk of the code for these three scripts is identical; only a few lines of code differ to produce these variations in DataTip context. The relevant excerpts of AspectTip1.sml are shown on the reverse side of this page, with the salient code section highlighted; the matching salient sections of the other two scripts are also shown for comparison, along with illustrations of the resulting DataTips.

Excerpts and Variations for AspectTip Scripts

Red-boxed code sections below in AspectTip1, AspectTip2, and AspectTip3 determine the DataTip form.

Global Variables

```
class RVC_RASTER ASPECT;                    the aspect raster object
class GRE_LAYER aspectLayer;                 the layer containing the aspect raster

class GRDEVICE_MEM_RGBA dev;                 RGB Alpha (opacity) graphics
                                            rendering device in memory
                                            for drawing in DataTip
class COLOR transparent(0, 0, 0, 0);        transparent color to clear device

numeric height, width;          height and width of graphics rendering device
```

procedure called when group or layout is initialized

```
proc OnInitialize () {
    width = 110;                    set width of rendering device in pixels
    height = width * 1.45;          set height of rendering device in pixels
    dev.Create(height, width);      create graphics rendering device
    }

func OnViewDataTipShowRequest (
    class GRE_VIEW view,            function called when
    class POINT2D point,           DataTip request is made
    class TOOLTIP datatip
    ) {                get transform from screen to layer coordinates and
                       convert cursor position to layer coordinates

trans = view.GetTransLayerToScreen(aspectLayer, 1);
point = trans.ConvertPoint2DFwd(point);
lin = floor(point.y);     col = floor(point.x);      line and column position
                                                      of cell under cursor

aspect = ASPECT[lin, col];      get cell value from ASPECT raster

if (aspect <> ASPECT.$Info.NullValue)    if cursor is not over a null cell
    {
    dev.Clear(transparent);      clear the graphics rendering device

    Draw static GraphTip elements
    gc = dev.CreateGC();      create graphics context

    draw background circle for aspect
    indicator and rectangle for aspect text
    center.x = width * 0.5;        center.y = center.x;
    radius = width * 0.45;         boxtop = center.x * 2;
    gc.SetColorRGB(240, 255, 240);
    gc.FillCircle(center.x, center.y, radius);
    gc.FillRect(15, boxtop, width-30, 35);

    draw black circle outline and black rectangle outline
    gc.SetColorName("black");
    gc.DrawCircle(center.x, center.y, radius);      circle outline
    gc.DrawRect(15, boxtop, width-30, 35);          rectangle outline

    set parameters for drawing Aspect text label in rectangle
    gc.DrawTextSetFont("ARIALBD.TTF");
    gc.DrawTextSetHeightPixels(12);
    color.Name = "black";
    gc.DrawTextSetColors(color);'

    draw centered Aspect label in rectangle
    start = center.x - ( gc.TextGetWidth("Aspect:") * 0.5 );
    gc.DrawTextSimple("Aspect", start, boxtop + 15);

    set aspect text and pointer parameters
    gc.DrawTextSetFont("ARIAL.TTF");      font for aspect value string

    if (aspect == -1) {      flat area with aspect undefined; only write string
        aspect$ = "Undefined";
        start = center.x - ( gc.TextGetWidth(aspect$) * 0.5 );
        gc.DrawTextSimple(aspect$, start, boxtop + 30);
        }
```

```
    else {          draw aspect value and pointer
        aspect$ = sprintf("%.1f deg", aspect);      create and draw
                                                    aspect label string
        start = center.x - ( gc.TextGetWidth(aspect$) * 0.5 );
        gc.DrawTextSimple(aspect$, start, boxtop + 30);

        aspectDrawAngle = (aspect - 90);      drawing angle for aspect pointer
        arrowLength = 40;
        gc.SetLineWidth(3);      set line width to 3 pixels for pointer

        find and store coordinates of end of pointer line
        arrowEnd.x = center.x + arrowLength * cosd(aspectDrawAngle);
        arrowEnd.y = center.y + arrowLength * sind(aspectDrawAngle);

        move to start of pointer line and draw to end
        gc.MoveTo( center.x - arrowLength *
            cosd(aspectDrawAngle), center.y - arrowLength *
            sind(aspectDrawAngle) );
        gc.SetColorName("red");
        gc.DrawTo(arrowEnd.x, arrowEnd.y);

    draw arrowhead and small black circle at center of rotation of pointer
        gc.DrawArrow(arrowEnd.x, arrowEnd.y, aspectDrawAngle, 9, 30, "Open");
        gc.SetColorName("black");
        gc.FillCircle(center.x, center.y, 3);
```
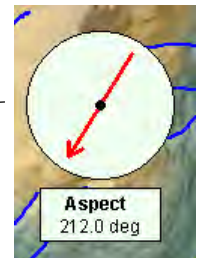
AspectTip1.sml

set graphics device as
an image in the DataTip

```
        datatip.SetImageTip(dev);
```

use only the image, suppress
the DataTip frame

```
        return -1;

        }
    }      end if not over null cell
```
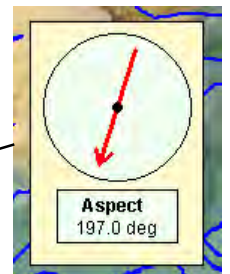
if outside ASPECT raster or on
null cell, don't show any DataTip

```
else return -1;

}      end OnViewDataTipShowRequest
```


Aspect
212.0 deg

Substituted in AspectTip2.sml

append the graphics
device to the DataTip

```
datatip.AppendImage(dev);
datatip.AppendText("\n");
```

render image in normal DataTip frame,
suppress entries from other layers

```
return 1;
```


Aspect
197.0 deg

Substituted in AspectTip3.sml

append the graphics
device to the DataTip

```
datatip.AppendImage(dev);
```

set font for following entries

```
datatip.AppendText("{~FARIAL.TTF}\n");
```

render image in normal DataTip frame,
allow entries from other layers

```
return 0;
```


Aspect
190.0 deg

Sinuosity: 1.08
Elevation: 2276.3 m
Slope: 30 deg