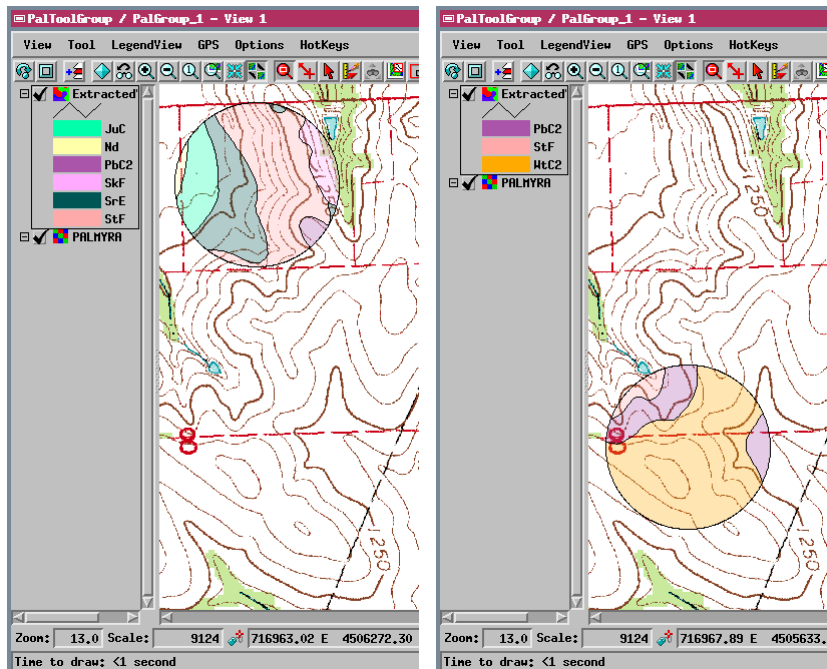# Sample GraphTip Script

Spyglass View

A GraphTip created by a Display Control Script automatically presents location-specific graphical information when the cursor pauses over a position in the View. Taking this concept a step further, a GraphTip can even pop-in a geographically-matching portion of another geospatial object at the cursor location. The effect is similar to the View-In-View tool provided with all TNT View windows, but the pop-in "spyglass" view can have any shape, appears automatically as determined by the Display Control Script, and requires no set-up by the end user. And unlike the View-In-View tool, a GraphTip can show a geospatial object that is not assigned to any view layer, but simply resides in an available Project File. The script that creates the GraphTip can also automatically adjust for any differences in Coordinate Reference System, datum, or cell size between the GraphTip source object and the geospatial data in the View. The script can also rescale the geodata within the GraphTip to match the current View scale.

MicroImages has prepared several sample Spyglass GraphTip scripts that are illustrated here and available for download from microimages.com. These scripts create a circular GraphTip whose diameter is set to be a fixed fraction of the size of the View window.

The SpyGlassRaster script (shown on the opposite side of this plate) uses standard GraphTip methods and structures
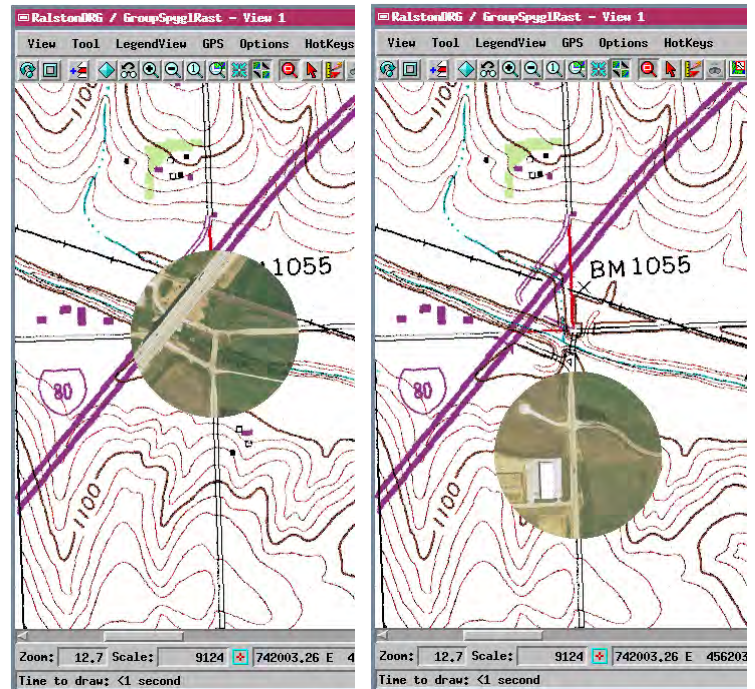
The SpyGlassRaster Display Control Script pops-in a circular GraphTip with an image of the cursor's geographic position. This image is centered on the cursor position and read from a specific raster object that is not a layer in the display group. In the illustration, the group displays a scanned topographic map, while the GraphTip pops-in the corresponding image from a color digital orthoimage.

to copy the geographically-matching portion of a raster object into the GraphTip at each GraphTip event. The SpyGlassVector script constructs a GraphTip that shows a matching vector object. This vector script cannot use conventional GraphTip methods, but instead extracts the relevant vector data to a temporary vector object covering only the area of the GraphTip. This temporary vector object is added / removed as a separate layer in the display group at each GraphTip event. Because the vector GraphTip is added to the display group as a layer, its legend appears automatically in LegendView, providing a context for the GraphTip content. This legend includes samples for only those data attributes present in the GraphTip extract from the parent vector, and it is updated automatically as the vector GraphTip layer is replaced at each new GraphTip event.

The SpyGlassVector Display Control Script pops-in a circular GraphTip with geographically matching data from a vector object. In this case the required portion of the vector data is extracted to a temporary vector object that is added / removed from the display group automatically as needed. In this illustration, the GraphTip shows the matching part of a soil map with transparent polygon fills, so the underlying topographic map image is visible through the GraphTip.

Script for Spyglass Raster GraphTip
(SpyGlassRaster.sml Display Control Script)

```
class GRE_LAYER_RASTER  F_layer;
class RASTER  F, Zoom, SourceRaster;
class POINT2D  rasterUL, offset, center;
```
global variable declarations

Graphics rendering device in memory for drawing 24-bit raster image as GraphTip. Use _RGB16 version for 16-bit color GraphTip image or _GRAY8 version for 8-bit grayscale GraphTip image.

```
class GRDEVICE_RAST_RGB24  rasterdev;
```

graphics rendering device in memory for binary image buffer (for transparency mask for GraphTip)

```
class GRDEVICE_MEM_BINARY  maskdev;

class GC gc;
```
graphics context for drawing to GraphTip mask

```
numeric height, width;
numeric count = 0;
class POINT2D SourcePoint, LayerPoint, MapPoint,
        SourceMapPoint, SourceObjPoint;
class RVC_GEOREFERENCE SourceGeo, FGeo;
class TRANSPARM ScreenToLayer, FMapToSourceMap;
class TRANSPARM FObjToMap, SourceMapToObj;
```

Procedure called when any view for the group is created. Gets the raster from the group's first layer. This raster (F) is used to establish a translation between the screen coordinates of the mouse cursor and the map coordinates of the desired location to be displayed in the GraphTip's view circle.

```
proc OnGroupCreateView (
    class GRE_GROUP group
    ) {
    F_layer = group.FirstLayer;
    DispGetRasterFromLayer(F, F_layer);
```
open raster that is source for spyglass image; file should be in same directory as control script
```
    string filename$ = _context.ScriptDir + "\\RalstonDOQQ.rvc";
    OpenRaster(SourceRaster, filename$, "RalstonDOQQ");
}
```

Function called when the mouse is left over a position. This is the primary function for creating and displaying the GraphTip.

```
func OnViewDataTipShowRequest (
    class GRE_VIEW view,
    class POINT2D point,     cursor position in screen coordinates
    class TOOLTIP datatip
    ) {

    height =view.height/4;
    width = view.width/4;
```
set height and width of GraphTip area (in screen pixels, as a fraction of View dimensions). Height and width should match. Height must be > 32
```
    if (height < 32 && width < 32) {
        height = width = 32;
    } else {
        if (height > width) {
            width = height;
        } else {
            height = width;
        }
    }

    maskdev.Create(height,width);
    maskdev.ClearAll();
```
create memory device for mask and set all to 0

```
    center.x = height/2;
    center.y = height/2;
```
position in mask for center of circle
```
    offset.x = -height/2;
    offset.y = -height/2;
```
offset GraphTip so it is centered on cursor position
```
    point.x = point.x + offset.x;
    point.y = point.y + offset.y;
```
reset point to upper left corner of GraphTip raster area
create temporary raster to construct spyglass image
```
    CreateTempRaster(Zoom, height, width, "24-bit color RGB");
```
set up projection transformations
```
    SourceGeo.OpenLastUsed(SourceRaster);
    SourceGeo.GetTransparm(SourceMapToObj, 1, SourceGeo.GetCalibModel() );
    FMapToSourceMap.OutputCoordRefSys = SourceGeo.GetCoordRefSys();
    FGeo.OpenLastUsed(F);
    FGeo.GetTransparm(FObjToMap, 0, FGeo.GetCalibModel() );
    FMapToSourceMap.InputCoordRefSys = FGeo.GetCoordRefSys();
    ScreenToLayer = view.GetTransLayerToScreen(F_layer, 1);
```
copy cells from the spyglass image raster into the temp raster, repeating (if zoomed in) or excluding (if zoomed out) cells as needed
```
    local numeric x, y;

    for (x = 0;  x < width; x++) {
        for (y = 0; y < height; y++) {     increment screen position
            SourcePoint.x = point.x + x;
            SourcePoint.y = point.y + y;
```
find position in lin and col of displayed raster
```
    LayerPoint = ScreenToLayer.ConvertPoint2DFwd(SourcePoint);
```
convert to map coordinates of displayed raster
```
    MapPoint = TransPoint2D(LayerPoint, FObjToMap);
```
find map coordinates in georeference used by spyglass image raster
```
    SourceMapPoint = FMapToSourceMap.ConvertPoint2dFwd(MapPoint);
```
convert map coordinates to lin and col of spyglass image raster
```
    SourceObjPoint = TransPoint2D(SourceMapPoint, SourceMapToObj);
```
copy cell from spyglass source raster into the temp raster
```
    Zoom[y+1,x+1] = SourceRaster[SourceObjPoint.y+1, SourceObjPoint.x+1];
        }
    }
```
create the raster device for the GraphTip and check for errors
```
    local numeric error = rasterdev.Create(Zoom);
    if (error < 0) {
        PopupError(error);
        return (error);
    }
```
create graphics context for mask and draw circular image area (mask = 1)
```
    gc = maskdev.CreateGC();
    gc.SetColorPixel(1);
    gc.FillCircle(center.y, center.x, width/2 - 2);
```
set up the GraphTip
```
    datatip.PixelDelta = 0;
    datatip.Delay = 300 ;
    datatip.MarginHeight = 100;
```
set the temp raster and binary mask as the source for the GraphTip image
```
    datatip.SetImageTip(rasterdev, maskdev, offset);

    DeleteTempRaster(Zoom);
```
in GraphTip use only what is created by script
```
    return (1);
}
```