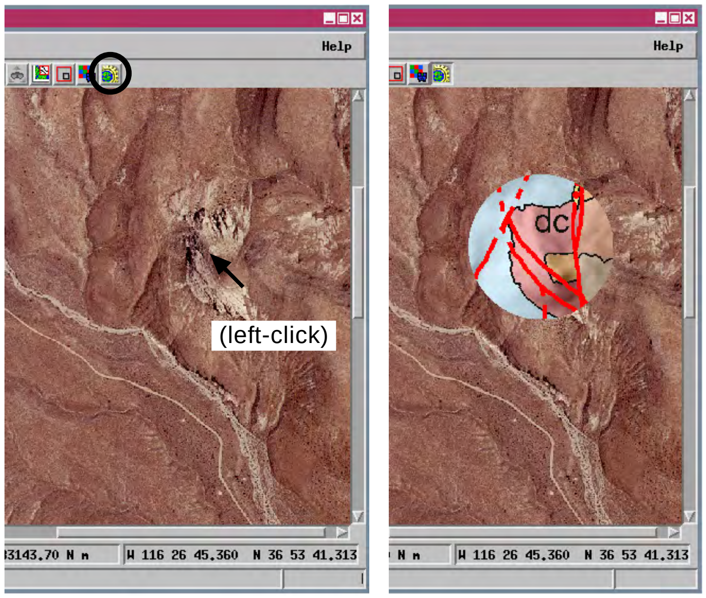
Sample Tool Script

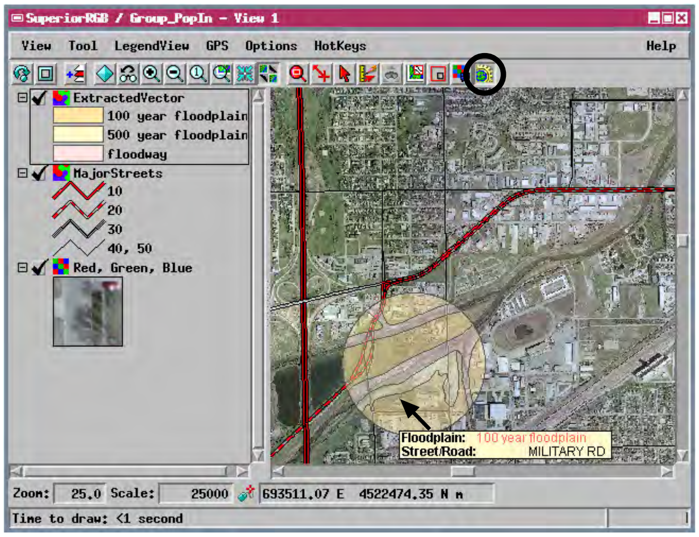
Pop-In View

A Tool Script can be used to present location-specific graphical information in a View in similar fashion to a GraphTip created by a Display Control Script. But while a GraphTip appears automatically with preset content whenever the cursor pauses over the View, with a Tool Script the user can have control over when and where the information appears and even over the specific content presented. A custom tool created by a Tool Script can be selected or deselected at any time from the View window's toolbar, and the user triggers the tool action (information presentation) explicitly with a mouse click. The Tool Script can also provide a dialog window(s) to allow selection of the geospatial object to be used as the source for the data presentation or to set other parameters. As with a GraphTip, this source object need not be a layer in the group or layout, and the ToolScript can also adjust for any differences in Coordinate Reference System, datum, or cell size between the source object and the geospatial data in the View. A Tool Script also can be saved with a group or layout for use in an atlas and can be set as the default tool when the atlas is opened. The View can also include multiple versions of the same tool tied to different data, or multiple custom tools with different purposes, each accessible from its own icon button or Tool menu entry (as many as needed) on the View. Like the standard tools on the View's toolbar, these custom tools can be added or removed from the window as needed to tailor the View to your intended audience.


Help
(left-click)
dc
3143.70 N m
W 116 26 45.360 N 36 53 41.313
N m
W 116 26 45.360 N 36 53 41.313

When you select the custom PopInView tool from the View window's tool bar, a dialog window automatically opens temporarily so you can specify the type of object to use for the pop-in view (raster or vector) and then select the object. In this example a raster object was selected to provide the circular pop-in view when and where the left mouse-button is pressed. This pop-in raster is a shaded-relief geologic map showing labeled rock units and faults (red lines). The background raster object is a color orthoimage.

The sample Tool Script PopInViewTool.sml was created by MicroImages to demonstrate these capabilities; excerpts of the script are shown on the reverse side of this page and the complete script is available for download from MicroImages.com. The tool created by the script has capabilities similar to the Spyglass GraphTip scripts described in the color plate entitled *Sample GraphTip Script: Spyglass View*. The tool creates a circular GraphTip view at the cursor location when the left mouse-button is pressed. When you first select the tool, a dialog opens to allow you to select the source of the geodata to present in the circular view (raster or vector object in a Project File or any linkable external file such as GeoTIFF, shapefile, MrSID, ...). The pop-in GraphTip then shows the matching circular portion of this object centered on the left-click position and matching the current scale of the View. When a raster object has been selected as a source, the pop-in view is simply rendered into the drawing area of the view window as needed. When a vector object has been selected as the source, the tool extracts the matching vector data to a temporary object covering only the GraphTip area. This temporary vector object is then added to the group as a separate display layer. In either case, the previous results are erased as each new pop-in view is presented or when another graphical tool is selected from the View's toolbar.



When you have selected a vector object as the source for the pop-in view, each left-mouse click extracts a circular area from the source vector and adds it to the group temporarily as a separate display layer. Legend entries for the pop-in vector therefore appear automatically in LegendView, and any previously-designated DataTips are active, including those for the pop-in vector object. In this example the base group includes a color orthoimage mosaic with an overlay of major streets and the pop-in view shows partially-transparent floodplain polygons. The DataTip exposed when the cursor pauses within the area of the GraphTip includes any location/element-specific information set up for the GraphTip object and for any other layers; in this case the DataTip identifies the 100-year floodplain from the GraphTip layer and the street name from the MajorStreets layer. The DataTip for areas outside the GraphTip will show only information from the original display layers (in this case the MajorStreets layer).

Many sample scripts have been prepared to illustrate how you might use the features of the TNT products’ scripting language for scripts and queries. These scripts can be downloaded from www.microimages.com/freestuf/scripts.htm.

Script Excerpts for Pop-In View Tool Script

(PopInViewTool.sml)

Procedure to extract circular area from source vector to show as tool result in View. Draws circle in MaskVector to use as extraction boundary; extracted vector is added to group as a layer

```
proc VectorTool () {
```

create temporary vector objects:

temp vector to draw circle in for extraction

```
    CreateTempVector(MaskVector,"VectorToolkit,Polygonal", "NoStatusText" );
    CopySubobjects(ToolVector, MaskVector, "GEOREF");
```

extracted vector to display as tool result

```
    CreateTempVector(ExtractedVector,"", "NoStatusText");
```

get projection information for coordinate transformations

```
    ToolGeo.OpenLastUsed(ToolVector);
    ToolGeo.GetTransparm(ToolMapToObj, 1, ToolGeo.GetCalibModel() );
    FMapToToolMap.OutputCoordRefSys = ToolGeo.GetCoordRefSys();
    FGeo.OpenLastUsed(F);
    FGeo.GetTransparm(FObjToMap, 0, FGeo.GetCalibModel() );
    FMapToToolMap.InputCoordRefSys = FGeo.GetCoordRefSys();
```

apply transformations

find position in lin and col of displayed raster

```
    SourcePoint = point;
    LayerPoint  = ScreenToLayer.ConvertPoint2DFwd(SourcePoint);
```

convert to map coordinates of displayed raster

```
    MapPoint = TransPoint2D(LayerPoint, FObjToMap);
```

find map coordinates in georeference used by tool vector

```
    ToolMapPoint = FMapToToolMap.ConvertPoint2dFwd(MapPoint);
```

convert map coordinates to object coordinates of tool vector

```
    ToolObjPoint = TransPoint2D(ToolMapPoint, ToolMapToObj);
```

draw circular polygon in mask vector to use as extraction boundary

```
    point =ToolObjPoint;
    shape.Clear();
```

clear previous polyline

class instance for computed vertex position on circumference of circle

```
    local class POINT2D circlepoint;
    local numeric x, y;
```

radius of circle in source object coordinates

```
    local numeric radius = 500;
    local numeric r2 = radius * radius;
    point.y += radius/2;
```

create top half of circle and add to polyline

```
    for (x = -radius; x <= radius; x++) {
        y = sqrt((r2 - x*x) +0.5);
        circlepoint.x = point.x + x;
        circlepoint.y = point.y + y;
        shape.AppendVertex(circlepoint);
        }
```

create bottom half of circle and add to polyline

```
    for (x = radius; x >= -radius; x--) {
        y = sqrt((r2 - x*x) +0.5);
        circlepoint.x = point.x +x;
        circlepoint.y = point.y -y;
        shape.AppendVertex(circlepoint);
        }
```

close polyline and add it as an element to the MaskVector to use for extraction

```
    shape.SetClosed(1);
    VectorAddPolyLine(MaskVector, shape);

    ExtractedVector = VectorExtract(MaskVector, ToolVector, "InsideClip");
    VectLayer = GroupQuickAddVectorVar(Group, ExtractedVector);
    View.RedrawDirect("NoBlankScreen");
    }
```

Procedure called when user presses 'left' pointer/mouse button

```
proc OnLeftButtonPress () {
    point.x = ceil(PointerX);
    point.y = ceil(PointerY);
```

get transparm from screen coordinates to raster layer coordinates

```
    ScreenToLayer = View.GetTransLayerToScreen(F_layer, 1);
```

transform cursor screen coordinates to object coordinates of raster that is first layer in group

```
    LayerPoint  = ScreenToLayer.ConvertPoint2DFwd(point);
```

check if cursor is on null cell

```
    local numeric onNull = IsNull( F[LayerPoint.y, LayerPoint.x] );

    if (isRaster) {
        View.RestoreAll();
        if (onNull == false) then
            RasterTool();
    }
```

clear previous tool image from View

if on non-null cell, show raster tool result

```
    else if (isVector) {
        local numeric removed = RemoveVectorLayer();
        if (removed == 2) then
            View.RedrawDirect("NoBlankScreen");
        if (onNull == false) then
            VectorTool();
    }
```

only redraw if a vector layer was actually removed

if on non-null cell, show vector tool result

```
    else {
        PopupMessage("Please select an input object");
        dialog.DoModal();
        }
    }
```

Procedure called when tool is activated

```
proc OnActivate () {
    F_layer = Group.FirstLayer;
    DispGetRasterFromLayer(F, F_layer);
    }
```

get handle for raster that is first layer in group

Procedure called when user presses 'left' pointer/mouse button

```
proc OnLeftButtonPress () {
    point.x = ceil(PointerX);
    point.y = ceil(PointerY);
```

get transparm from screen coordinates to raster layer coordinates

```
    ScreenToLayer = View.GetTransLayerToScreen(F_layer, 1);
```

transform cursor screen coordinates to object coordinates of raster that is first layer in group

```
    LayerPoint  = ScreenToLayer.ConvertPoint2DFwd(point);
    local numeric onNull = IsNull( F[LayerPoint.y, LayerPoint.x] );
```

check if cursor is on null cell

```
    if (isRaster) {
        View.RestoreAll();
        if (onNull == false) then
            RasterTool();
        }   else if (isVector) {
        local numeric removed = RemoveVectorLayer();
        if (removed == 2) then
            View.RedrawDirect("NoBlankScreen");
        if (onNull == false) then
            VectorTool();
        } else {
        PopupMessage("Please select an input object");
        dialog.DoModal();
        }
    }
```

clear previous tool image from View

only redraw if a vector layer was actually removed