Sample SML Tool Script

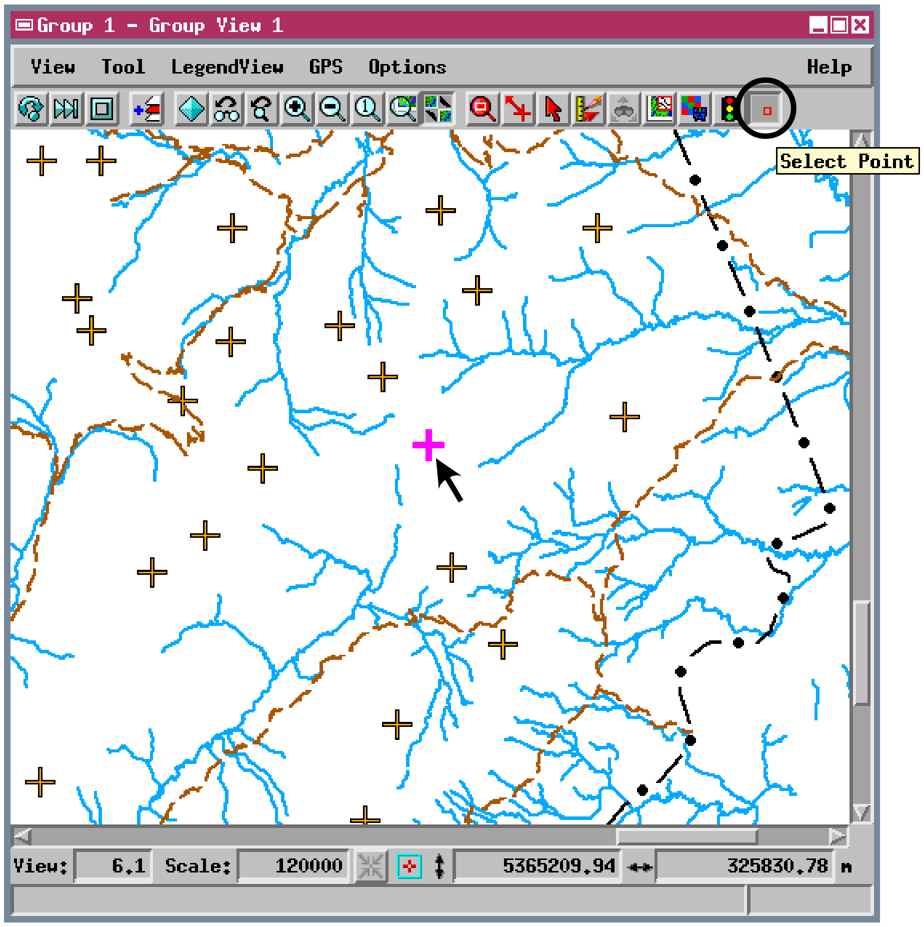
Select Point

The Select Point sample script illustrates how to write a tool script that lets the user interactively select elements from a vector object in the View window. In this simple script, which is shown on the other side of this page, point elements are the target.

Like other tool scripts, you install the script using the Customize / Tool Scripts... selection on the Options menu of a View window, after which you can activate it by pressing the resulting icon button on that (or any other) View window. You then just move the mouse pointer to the desired point in the window and click the left mouse button. The script finds the closest point element, records its element ID number, and highlights the point's symbol in the View window.

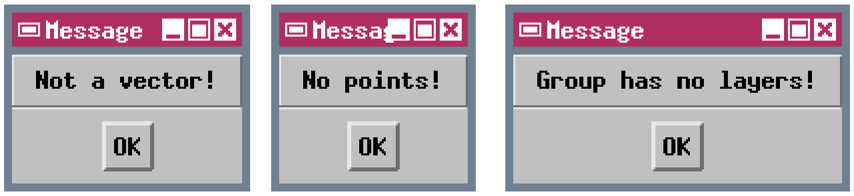
The sample Select Point script implements only the location and selection of point elements. MicroImages presents it as a prototype for many possible scripts that you could develop to carry out interactive selection and processing of individual vector elements. To extend the script to perform an action on or with the point element, you would add appropriate script statements to the end of the function that is called when the user presses the left mouse button. For example, the script could read the map coordinates of the selected point and save them to an external file, or pass them to another script function or external program for processing.

The Select Point script was written for the general case in which a View window can display either a single group or a display layout or print layout with several groups. If the View is a layout, the vector point object must be in the active group. If there are several data layers in the active group, the object containing the point elements must be the active layer. If these conditions are not met, appropriate error messages are displayed in popup dialog windows. (Note that tool scripts must be installed separately for normal single-group view windows, display layout windows, and print layout windows.)



When you run the Select Point tool script, point elements in the active layer (shown in this view by cross symbols) can be selected one at a time by clicking the left mouse button. The point element near the center of the view has been selected, and has been highlighted by the script (magenta cross symbol).

No interactive graphic tool is created by the Select Point script because in this case none is needed. Tool scripts are able to directly record the mouse pointer coordinates, which are stored in the predefined variables PointerX and PointerY (in screen coordinates). Other sample tool scripts provide examples of the use of interactive point, line, and polygon graphic tools to acquire information from the view window for processing.



Not a vector!

No points!

Group has no layers!

The Select Point script also illustrates how to design a tool script to handle potential error conditions and show error messages in popup dialog windows.

Macro and Tool Scripts can be created using SML in any TNTmips process that uses a View window (Options / Customize from the View window menu bar). These scripts are then available from an icon, which you select or design, on the toolbar. Sample scripts have been prepared to illustrate how you might use these features, which are available only in TNTmips 6.4 or later, to assist with specific tasks you perform on a regular basis. If possible, the full script is printed below for your quick perusal. When a script is too long to fit on one page, key sections are reproduced below. All sample Tool and Macro Scripts illustrated can be found in their entirety on your TNT products CD-ROM in the folder in which you installed TNTmips 6.6. These scripts, among others, can be downloaded from the SML script exchange at www.microimages.com/sml/ftpsmllink/TNT_Products_V6.4_CD.

Script for Select Point (pointsel.sml)

```
# The following symbols are predefined

#  class VIEW View
```

use to access the view the tool script is attached to

```
#  class GROUP Group
```

use to access the group being viewed if the script is run from a group view

```
#  class LAYOUT Layout
```

use to access the layout being viewed if the script is run from a layout view

```
# The following values are also predefined and
# are valid when the various On...()functions are
# called which deal with pointer and keyboard
# events.

# number PointerX
```

pointer X coordinate within view in pixels

```
# number PointerY
```

pointer Y coordinate within view in pixels

```
class VECTORLAYER vectorLayer;
class Vector targetVector;
class GROUP activegroup;
```

variable declarations

checks layer to see if it is valid

```
func checkLayer() {
   local boolean valid = true;

   if (activegroup.ActiveLayer.Type == “”) {
   PopupMessage(“Group has no layers!”);
   valid = false;
   }
```

get name of active layer if it is usable, if not output an error message

```
   else if (activegroup.ActiveLayer.Type ==
      “Vector”) {
   vectorLayer = activegroup.ActiveLayer;
   DispGetVectorFromLayer(targetVector,
      vectorLayer);
   if (targetVector.$Info.NumPoints < 1) {
      PopupMessage(“No points!”);
      valid = false;
      }
   }

   else {
      PopupMessage(“Not a vector!”);
      valid = false;
      }
   return valid;
   }
```

```
func OnLeftButtonPress () {
```

called when user presses ‘left’ pointer/mouse button

```
   if (checkLayer()) {
```

if the selected layer is not valid, don’t do anything

```
      local class POINT2D point;
```

set local variables

```
      point.x = PointerX;
      point.y = PointerY;
```

get point tool location

```
      point = TransPoint2D(point,
         ViewGetTransViewToScreen(View, 1));
      point = TransPoint2D(point,
         ViewGetTransMapToView(View,
         vectorLayer.Projection, 1));
```

transform from screen to map coordinates

find closest vector point element

```
      elementNum = FindClosestPoint(targetVector,
         point.x, point.y,
         GetLastUsedGeorefObject(targetVector));

      if (elementNum > 0)
   vectorLayer.Point.HighlightSingle(elementNum);
      }
```

highlight point element in view

Add code here to define an action to perform on or with the highlighted point element.

```
   }
```

end of OnLeftButtonPress

callback for when the active group changes in layout

```
proc cbGroup() {
   activegroup = Layout.ActiveGroup;
   }
```

```
func OnInitialize () {
```

called the first time the tool is activated

```
   if (Layout) {
```

if view is a layout, add a callback to the function to check the active group

```
   WidgetAddCallback(Layout.GroupSelectedCallback,
         cbGroup);
      activegroup = Layout.ActiveGroup;
      }

   else
      activegroup = Group;
   }
```

end of OnInitialize