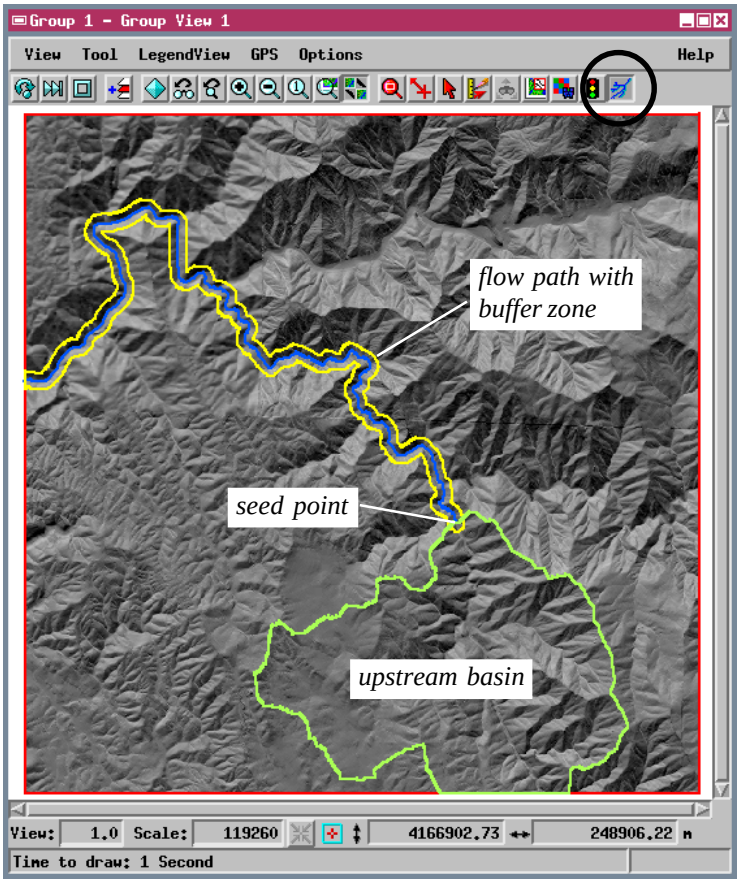
Sample SML Tool Script

Flow Path

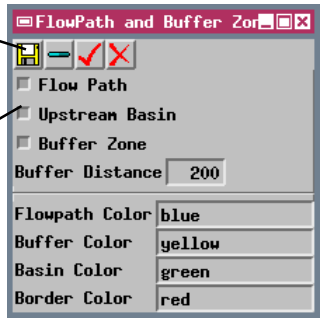
The Flow Path sample script shows how powerful custom analysis procedures can be performed on layers in the current view using an SML Tool Script. The script uses new SML watershed functions that operate on an elevation raster (DEM) in the View window. When you launch the script, it opens a FlowPath and Buffer Zone window and creates a graphic tool that allows you to place a watershed seed point on the DEM or on an overlying image layer. Depending on the options you have selected in the FlowPath window, the script computes and displays the upstream basin (area with flow toward the seed point), the flow path downstream, and a buffer zone around the flow path. You can move the seed point and repeat the analysis as many times as you like and save the computed elements at any time.

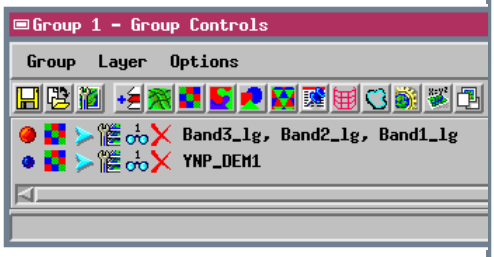
One application of this script is the evaluation of surface water pollution hazards. If the seed point represents a location where contamination has been detected, the upstream basin is the area of potential sources. If the seed point represents a contaminant spill, the flow path and buffer zone indicate the downstream area that is at risk.



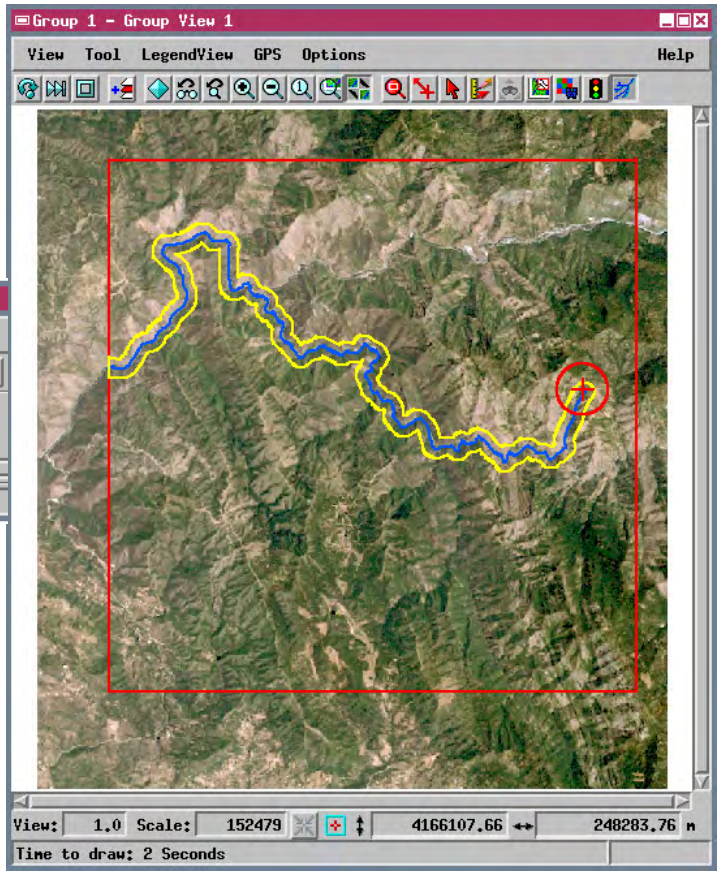
Use the Save button to save the computed watershed features as vector objects in a Project File.

Use the toggle buttons to choose which watershed features you want computed and displayed.





If the Display group has more than one layer, make sure that the elevation raster is the first layer in the group when you run the Flow Path Tool Script. In the example to the right, an RGB raster layer showing a natural-color satellite image is displayed on top of the DEM. The extents of the DEM layer are computed by the script and displayed as a red rectangle. Use this rectangle as a guide to placing seed points when the overlying image extends beyond the DEM layer in one or more directions.



Development of this and other sample Tool Scripts continues at MicroImages. Check the MicroImages web site for an updated version incorporating additional features.

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 6.4. 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 Flow Path (FlowPath.sml)

variable declarations

```
array seedx[10];                array seedy[10];
class WATERSHED w;              numeric numpts;
class RASTER DEM;               numeric firstpass;
class POINT2D pt;               class PointTool point_tool;
class VECTORLAYER VecBuf;       class RasterLayer DEMLayer;
class VECTORLAYER VecFlow;      numeric haslayers;
class VECTORLAYER BasinLayer;   class XmForm dlgform;
class VECTORLAYER BoundaryLayer; class VECTOR VectIn;
class VECTOR VecBoundary;       array xPoints[10],yPoints[10];
numeric xMax,yMax,xMin,yMin;    class PromptNum PromptDistance;
```

is called the first time the tool is activated

```
func OnInitialize() {
   if (Group.FirstLayer.Type == "Raster") {
      DispGetRasterFromLayer(DEM,Group.FirstLayer);
      DEMLayer = Group.FirstLayer;
      }
   else {
      PopupString("First Layer must be a raster object for Watershed
            Toolscript");
      WaitForExit();
      }

   demFilename$ = GetObjectFileName(DEM);
   demInode = GetObjectNumber(DEM);
   demObjname$ = GetObjectName(demFilename$,demInode);
```

initializes watershed object (w); compute depressionless DEM

```
   w = WatershedInit(demFilename$,demObjname$);

   WatershedCompute(w,"FillAllDepressions");

   firstpass = 1;   haslayers = 0;   numpts = 1;
```

creates "FlowPath and Buffer Zone" dialog window

```
   dlgform = CreateFormDialog("FlowPath and Buffer Zone",View.Form);
   WidgetAddCallback(dlgform.Shell.PopdownCallback,DoClose);

   class PushButtonItem btnItemSave;
   class PushButtonItem btnItemRemove;
   class PushButtonItem btnItemSet;
   class PushButtonItem btnItemClose;

   btnItemSave = CreatePushButtonItem("Save Output Layers...",DoSave);
   btnItemSave.IconName = "save";
   btnItemRemove = CreatePushButtonItem("Remove Output Layers",
                cbDoRemove);
   btnItemRemove.IconName = "remove_sel";
   btnItemSet = CreatePushButtonItem("Set Number of Seedpoints...",
                DoSet);
   btnItemSet.IconName = "apply";
   btnItemClose = CreatePushButtonItem("Close",DoClose);
   btnItemClose.IconName = "delete";

   class XmRowColumn btnrowaction;
   btnrowaction = CreateIconButtonRow(dlgform,btnItemSave,
                  btnItemRemove,btnItemSet,btnItemClose);
   btnrowaction.TopWidget = dlgform;
   btnrowaction.RightWidget = dlgform;
   btnrowaction.LeftWidget = dlgform;

   class XmToggleButton btnFlow;
   class XmToggleButton btnBasin;
   class XmToggleButton btnBuffer;

   btnFlow = CreateToggleButton(dlgform,"Flow Path");
   btnBasin = CreateToggleButton(dlgform,"Upstream Basin");
   btnBuffer = CreateToggleButton(dlgform,"Buffer Zone");
   PromptDistance = CreatePromptNum(dlgform,"Buffer Distance",
                    5,0,100);
   btnFlow.Set = 1;    btnBasin.Set = 1;   btnBuffer.Set = 1;

   btnFlow.TopWidget = btnrowaction;
   btnFlow.LeftWidget = dlgform;
   btnBasin.TopWidget = btnFlow;
   btnBasin.LeftWidget = dlgform;
   btnBuffer.TopWidget = btnBasin;
   btnBuffer.LeftWidget = dlgform;
   PromptDistance.TopWidget = btnBuffer;
   PromptDistance.LeftWidget = dlgform;

   class XmSeparator btnsep;
   btnsep = CreateHorizontalSeparator(dlgform);
   btnsep.TopWidget = PromptDistance;
   btnsep.TopOffset = 4;
   btnsep.LeftWidget = dlgform;
   btnsep.RightWidget = dlgform;

   class PromptStr Promptflow;
   class PromptStr Promptzone;
   class PromptStr Promptbasin;
   class PromptStr Promptborder;

   Promptflow = CreatePromptStr(dlgform,"Flowpath Color",15,"blue");
   Promptzone = CreatePromptStr(dlgform,"Buffer Color   ",15,"yellow");
   Promptbasin =CreatePromptStr(dlgform,"Basin Color    ",15,"green");
   Promptborder=CreatePromptStr(dlgform,"Border Color   ",15,"red");

   Promptflow.TopWidget = btnsep;
   Promptflow.TopOffset = 4;
   Promptflow.LeftWidget = dlgform;

   Promptzone.TopWidget = Promptflow;
   Promptzone.LeftWidget = dlgform;
   Promptbasin.TopWidget = Promptzone;
   Promptbasin.LeftWidget = dlgform;
   Promptborder.TopWidget = Promptbasin;
   Promptborder.LeftWidget = dlgform;
   } # end of OnInitialize
```

computes flow path, buffer zone, and basin originating at seed point (depending on options selected in dialog)

```
proc DoFlowPath() {
   View.DisableRedraw = 1;
   if ((btnFlow.Set == 0) and (btnBuffer.Set == 0) and
         (btnBasin.Set == 0)) {
      return;
   }
   if ((btnFlow.Set == 1) or (btnBuffer.Set == 1) and
         (btnBasin.Set == 1)) {
      WatershedComputeElements(w,seedx,seedy,numpts,"FlowPath,Basin");
   }
   if ((btnBasin.Set == 1) and (btnBuffer.Set == 0) and
         (btnFlow.Set == 0)){
      WatershedComputeElements(w,seedx,seedy,numpts,"Basin");
   }
   if (btnBasin.Set == 0) {
      WatershedComputeElements(w,seedx,seedy,numpts,"FlowPath");
   }

   if  ((btnFlow.Set == 1) or (btnBuffer.Set == 1)) {
   WatershedGetObject(w,"VectorUserFlowPath",userflowpathFilename$,
         userflowpathObjname$);
   OpenVector(VectIn,userflowpathFilename$,userflowpathObjname$);
   }
```

adds flow path vector to view

```
   if (btnFlow.Set == 1) {
      VecFlow = GroupQuickAddVectorVar(Group,VectIn);
      VecFlow.Line.NormalStyle.Color.name = Promptflow.value;
   }
```

computes optional buffer zone around flow path and add to view

```
   if (btnBuffer.Set == 1) {
      CreateTempVector(Buffer);
      CreateTempVector(TempBuffer);
      TempBuffer = VectorToBufferZone(VectIn,"line",
            PromptDistance.value,"meters");
      Buffer = VectorExtract(VecBoundary,TempBuffer,"InsideClip");
      VecBuf = GroupQuickAddVectorVar(Group,Buffer);
      VecBuf.Line.NormalStyle.Color.name = Promptzone.value;
      }
```

adds basin vector to view

```
   if (btnBasin.Set == 1) {
      WatershedGetObject(w,"VectorUserBasin",userBasinFilename$,
            userBasinObjname$);
      OpenVector(BasinVector,userBasinFilename$,userBasinObjname$);
      BasinLayer = GroupQuickAddVectorVar(Group,BasinVector);
      BasinLayer.Line.NormalStyle.Color.name = Promptbasin.value;
   }

   BoundaryLayer.Line.NormalStyle.Color.name = Promptborder.value;
   View.DisableRedraw = 0;
   ViewRedrawIfNeeded(View);
   haslayers = 1;
   }  # end of DoFlowPath
```