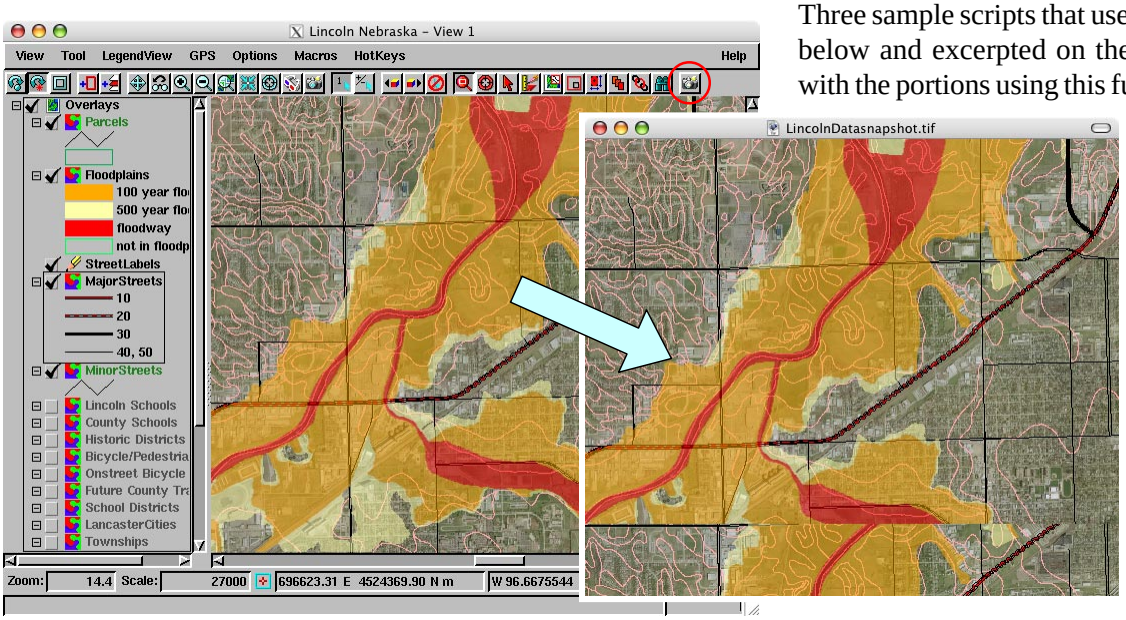
DRAFT
1 Oct 2007

Launch Programs with TNTmips Data

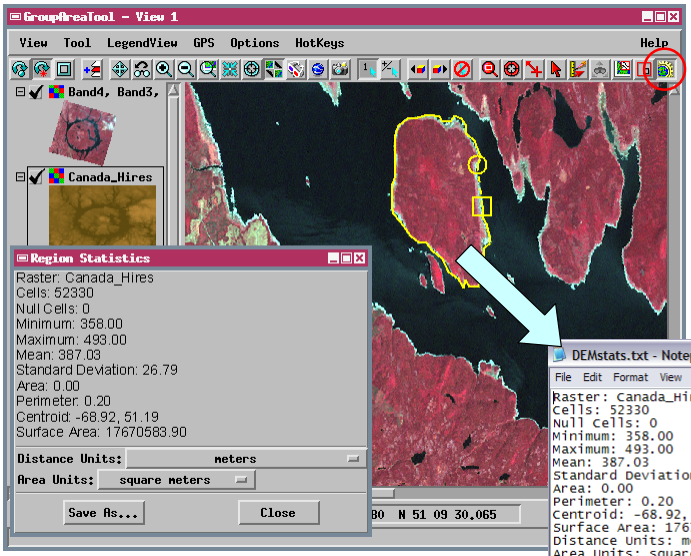
Interactive geospatial scripts such as Tool Scripts and Macro Scripts can process the spatial objects in a View window to derive information and/or new spatial objects that can be saved or exported to various external file formats. Once such a file has been created, the script can also automatically open the file in the software program with which you have registered that file type on your computer. A simple function called RunAssociatedApplication() is used for this purpose.

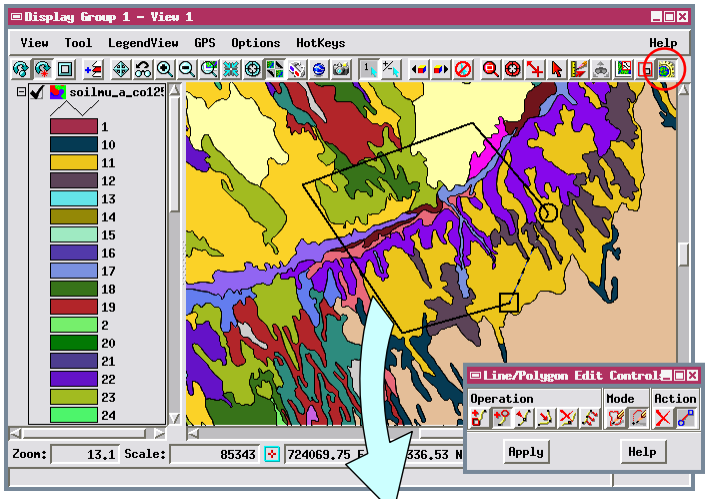
Three sample scripts that use this function are illustrated below and excerpted on the reverse side of this page, with the portions using this function highlighted in color.

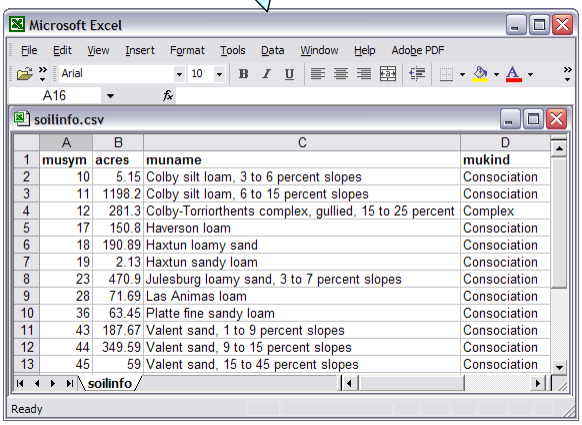


The Snapshot macro script takes a snapshot of the View contents, exports it to the desired raster file format (such as JPEG, TIFF, PNG, and BMP) and then opens the file in the program associated with that file type. In this example run under Mac OS X, the output TIFF file opens in Preview.

In the illustrations to the right, the Soil_Info tool script compiles cumulative area and soil attributes from a user-defined portion of a vector soil map, writes the information to a CSV (Comma-Separated Values) file, and opens that file in the program associated with that file type (in this case, Microsoft Excel).







	A	B	C	D
1	musym	acres	muname	mukind
2	10	5.15	Colby silt loam, 3 to 6 percent slopes	Consociation
3	11	1198.2	Colby silt loam, 6 to 15 percent slopes	Consociation
4	12	281.3	Colby-Torriorthents complex, gullied, 15 to 25 percent	Complex
5	17	150.8	Haverson loam	Consociation
6	18	190.89	Haxtun loamy sand	Consociation
7	19	2.13	Haxtun sandy loam	Consociation
8	23	470.9	Julesburg loamy sand, 3 to 7 percent slopes	Consociation
9	28	71.69	Las Animas loam	Consociation
10	36	63.45	Platte fine sandy loam	Consociation
11	43	187.67	Valent sand, 1 to 9 percent slopes	Consociation
12	44	349.59	Valent sand, 9 to 15 percent slopes	Consociation
13	45	59	Valent sand, 15 to 45 percent slopes	Consociation

The Area Statistics tool script (described in the Technical Guide entitled *Sample SML Tool Script: Area Statistics*) computes raster statistics for a user-defined area and allows the statistics to be saved as a text file. For the illustration above, this script was modified so that its save procedure (excerpted on the reverse) automatically opens that text file in the associated software program (in this case, Notepad).

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/downloads/scripts.htm.

Excerpt from Snapshot Macro Script (Snapshot.sml)

```
CreateTempRaster(rast, 1, 1, "binary");
```

Create a temporary raster and then get the filename from it.

Get the snapshot.

```
ViewSaveSnapshot(View, rast.$Info.Filename, "snap", "snapshot of view");
```

Export and open the snapshot.

```
if (MenuChoice$ == "JPEG") {
   OpenRaster(ras, rast.$Info.Filename, "snap");
   ConvertCompToComp(ras, rast.$Info.Filename, "snapa", 24);
   CloseRaster(ras);
   jpegHandle.exportCompressFactor = 75;
   ExportRaster(jpegHandle, _context.ScriptDir + "snapshot.jpg",
         rast.$Info.Filename, "snapa");
   RunAssociatedApplication(_context.ScriptDir + "snapshot.jpg");
   }

   [... similar steps for PNG, BMP, PCX, and GIF formats omitted...]

else if (MenuChoice$ == "TIFF") {
   tiffHandle.CompressionType = "NONE";
   tiffHandle.PlanarConfiguration = "Pixel interleaved";
   tiffHandle.ExportGeoTags = 1;
   OpenRaster(ras, rast.$Info.Filename, "snap");
   ConvertCompToComp(ras, rast.$Info.Filename, "snapa", 24);
   CloseRaster(ras);
   ExportRaster(tiffHandle, _context.ScriptDir + "snapshot.tif",
         rast.$Info.Filename, "snapa");
   RunAssociatedApplication(_context.ScriptDir + "snapshot.tif");
   }

else if (MenuChoice$ == "TXT") {
   ExportRaster(asciiHandle, _context.ScriptDir + "snapshot.txt",
         rast.$Info.Filename, "snap");
   RunAssociatedApplication(_context.ScriptDir + "snapshot.txt");
   }

else if (MenuChoice$ == "DOC") {
   ExportRaster(asciiHandle, _context.ScriptDir + "snapshot.doc",
         rast.$Info.Filename, "snap");
   RunAssociatedApplication(_context.ScriptDir + "snapshot.doc");
   }

DeleteTempRaster(rast);
```

Delete the temporary file.

Area Statistics Tool Script (regstats.sml) with modifications (red) to open output file (modified version: regstats2.sml)

```
proc cbSave() {
```

Called when the save button is pressed. Saves statistics to text file.

```
#  outfile = GetOutputTextFile(_context.ScriptDir, "Choose file...", "txt");
```

Prompt user for the name of the output text file, open the file, write to it, and close it

```
   textfile$ = GetOutputFileName(_context.ScriptDir, "Choose file...", "txt");
   outfile = fopen(textfile$, "w", "ASCII");
   fprintf(outfile, "Raster: %s\nCells: %d\nNull Cells: %d\nMinimum:
         %.2f\nMaximum: %.2f\nMean: %.2f\nStandard Deviation:
         %.2f\nArea: %.2f\nPerimeter: %.2f\nCentroid: %.2f, %.2f\n
         Surface Area: %.2f", rasterName$, count, cells - count, min, max,
         mean, stdDev, larea, lperimeter, centroid.x, centroid.y, lsurface);
   fprintf(outfile, "\nDistance Units: %s\nArea Units: %s\n\n",
         distMenu.value, areaMenu.value);
   fclose(outfile);

   RunAssociatedApplication(textfile$);
   }
```

Open text file in associated program

Excerpt from Soil Info Tool Script (Soil_Info.sml)

```
proc OnRightButtonPress () {
```

Called when user presses right mouse button.

```
   if ( checkLayer() ) {
      soilVec.GetDefaultGeoref(vGeoref);
```

get the georeference for the soil vector

Region created by tool has screen coordinates; must translate to map coordinates of the view (view coordinates) and then to map coordinates of the vector object in the active layer.

```
      toolRegion = RegionTrans(tool.RegionData, View.GetTransViewToScreen(1));
      toolRegion = RegionTrans(toolRegion,
         View.GetTransMapToView(vGeoref.GetCoordRefSys(), 1) );
```

convert region to temporary vector object to use for extraction

```
      CreateTempVector(regionVec);
      regionVec = ConvertRegionToVect(toolRegion);

      View.SetMessage("Extracting area...");
      CreateTempVector(xSoilVec);
      xSoilVec = VectorExtract(regionVec, soilVec,
            "InsideClip", "AddBorder,RemExRecords");
      CloseVector(regionVec);
      View.SetMessage("Making CSV file...");
```

extract the area of the soil vector object to a temporary vector for processing

prompt user for name of output CSV file; open file and write header line

```
      outfileName$ = GetOutputFileName(dfltName$, "Choose output CSV file:", "csv");
      outfile = fopen(outfileName$, "w");
      fprint(outfile, "musym,acres,muname,mukind");
```

loop through extracted polygons to get areas from POLYSTATS table; use HASH to keep track of soil types and accumulate areas

```
      for i = 1 to xSoilVec.$Info.NumPolys {
         key$ = xSoilVec.Poly[i].mapunit.musym$;
         polyarea = xSoilVec.Poly[i].POLYSTATS.Area;

         if (soilAreaHash.Exists(key$) ) {
```

soil type is already in the soil area hash

add area of current polygon to value already stored for that soil

```
            soilAreaHash[key$] = soilAreaHash[key$] + polyarea;
            }
         else {
```

new soil type

```
            soilAreaHash[key$] = polyarea;
            polyNumHash[key$] = i;
            }
         }
```

assign area of polygon to soil area hash key

assign element number of polygon to polynum hash key

get list of hash keys as a stringlist to loop through them

```
      keylist = soilAreaHash.GetKeys();
```

loop through keys by numeric position in stringlist (starting with 0)

```
      for i = 0 to keylist.GetNumItems() - 1 {
         musym$ = keylist.GetString(i);
         polyarea = soilAreaHash[musym$];
         polyarea = polyarea * areaScale;
```

soil identifier

area for soil type; convert area from square meters to acres

get soil name and kind from fields in mapunit table using polygon numbers stored in polynum hash

```
         polynum = polyNumHash[musym$];
```

field contains commas, so must be quoted for use in CSV file

```
         muname$ = "\"" +xSoilVec.Poly[polynum].mapunit.muname$ + "\"";
         mukind$ = xSoilVec.Poly[polynum].mapunit.mukind$;
```

write values to a line in the output CSV file

```
         fprintf(outfile, "%s,%.2f,%s,%s\n", musym$, polyarea, muname$, mukind$);
         }
      View.SetMessage("Done.");
      fclose(outfile);
      RunAssociatedApplication(outfileName$);
      CloseVector(xSoilVec);
      }
   }
```

launch program associated with the CSV file type (e.g. Excel)

end of OnRightButtonPress