GEO SCRIPTING

Introduction to

Geospatial Scripting

with TNTgis

Before Getting Started

This booklet provides an overview of the ways in which geospatial scripts can be used in TNTgis. You can create and use scripts to query attributes of geospatial layers, customize DataTips for layers and groups, create interactive tools for exploring and interpreting geospatial data, and implement custom processes. Sample scripts are presented to illustrate typical applications of each type of geospatial script.

Prerequisite Skills This booklet assumes that you have completed the exercises in the tutorial booklets *Getting Started with TNTmips, TNTedit, and TNTview*, and *Introduction to the Display Interface*. Those exercises introduce essential skills and basic techniques that are not covered again here. Please consult those booklets for any review you need.

Sample Data The exercises in this booklet use sample data that is distributed with TNTgis. If you do not have access to a TNTgis DVD, you can download the data from MicroImages' web site. In particular, this booklet uses data in the SML data collection and sample scripts that can be installed by following the instructions on page 6.

More Documentation This booklet is intended only as an introduction to geospatial scripting in TNTgis. Consult the tutorial booklets entitled *Building and Using Queries*, *Using CartoScripts*, *Using Geospatial Formulas*, *Writing Scripts with SML*, and *Building Dialogs in SML* for more information on writing and using geospatial scripts.

TNTmips® Pro and TNTmips Free TNTmips comes in three versions: TNTmips Pro (which requires a software license key), low-cost TNTmips Basic, and TNTmips Free. TNTmips Basic and TNTmips Free provide nearly all the capabilities of TNTmips Pro but limit the size of the geospatial objects and attribute tables that can be used in your project. TNTmips Basic and TNTmips Free allow you to create and use scripts in the Display process (database queries, style scripts, macroscripts, and others) and the Geoformula process, but only TNTmips Pro allows you to create and run complex standalone geospatial scripts.

Randall B. Smith, Ph.D., 17 December 2015

You can print or read this booklet in color from MicroImages' web site. The web site is also your source for the newest tutorial booklets on other topics. You can download an installation guide, sample data, and the latest version of TNTmips Free.

http://www.microimages.com

Introduction to Geospatial Scripting

TNTgis provides a comprehensive, integrated set of tools and processes that enable you to view, process, interpret, and publish almost any form of geospatial data. Because of the breadth and variety of available TNTgis processes, many clients are able to complete their geospatial projects using only the predefined interactive tools and procedures. For those less common occasions that require a customized approach or process, TNTgis also provides a geospatial scripting language called SML.

Geospatial scripts can be used for a variety of tasks. The task may be as simple as using a particular set of database attributes to select vector elements (points, lines, polygons, nodes) for display or processing. Or you may want to customize and enhance the DataTips that provide automatic, context-sensitive attribute information for geospatial data in a View window. On the other hand, you can also use SML for complex tasks such as implementing a custom process or automating a processing workflow that takes the place of several separate predefined TNTgis processes. Standalone processing scripts can be fairly simple to write in many cases because SML includes functions and classes that provide access to many predefined TNTgis processes.

This booklet provides an overview of the different types of script that can be used in TNTgis. It discusses the scope of each type of script, the program interface for script setup, and provides some examples of script applications. Once you have completed this scripting overview, you should consult the tutorial entitled *Writing Scripts with SML* for an introduction to the syntax and programming structure of SML. You can also refer to the tutorial booklets *Building and Using Queries*, *Using CartoScripts*, and *Using Geospatial Formulas* for more detailed information about these particular script types.

Pages 4-6 provide an overview of the different types of scripts that can be used in TNTgis and the large collection of TNTgis sample scripts. A brief introduction to the script editor is given on pages 7-8. Simple scripts to create computed fields in database tables are covered on pages 9-11, followed by a discussion of database queries and style scripts on pages 12-13. The GeoFormula interface for computing virtual spatial layers is discussed on pages 14-16. Pages 17-18 introduce the use of Display Control scripts for creating enhanced and complex DataTips. Standalone geospatial process scripts are covered on pages 19-24, including their use in TNTmips Job Processing and TNTscript. Use of Macro and Tool Scripts in geospatial Views is discussed on pages 25-29, followed by SML Layer Scripts on page 30. Additional uses of Display Control Scripts are described on page 31.

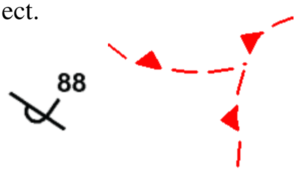
Geospatial Script Overview

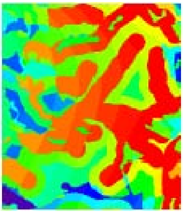
Database Computed Field Script: Use simple scripts to create dynamically-updated database fields that can use and combine values in other fields in the same or related tables. You can use mathematical expressions to create computed numeric fields or combine text in computed string fields.

Database Query: Use a database query to select elements in a geometric object (vector, CAD, shape, or TIN) on the basis of one or more attributes in the associated database. Query selection can control the marking, display, editing, or processing of geometric elements.

Style Script: Use a style script to set up custom display styling for elements in a geometric object. A style script can set colors, enable drawing of polygon borders and fills, and access patterns and styles in a style object.

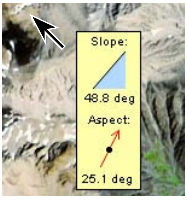
CartoScript: A CartoScript is a form of style script that uses specialized functions to create cartographic symbolism for points and lines in geometric objects.



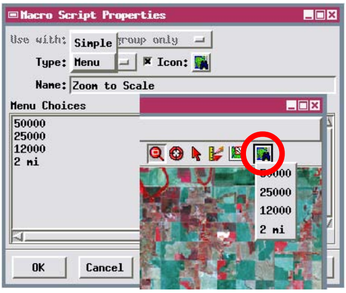


GeoFormula: Use the interactive GeoFormula dialog to create a virtual display layer or a saved raster object that is a mathematical combination of selected spatial objects. A GeoFormula can utilize raw or contrast-enhanced values from input raster objects and attribute values from elements in geometric objects.

Display Control Script: Use a Display Control Script to automatically modify the presentation of geospatial data in a View. For example, you can augment or replace the location-specific DataTip text that automatically pops in at the cursor location or replace the text entirely with a graphical presentation of the same location-specific information.



Macro Script: Use a macro script to implement custom commands that operate on the geospatial data in a View window. A macro script can change the view of the data, operate on data in the View, or start an external program and provide it with data from and about the layers in the View. You can run a macro script from the View window using an icon button or from the Macros menu.


Macro Script Properties

Use with: Simple group only

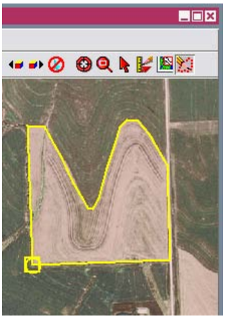
Type: Menu ☒ Icon:

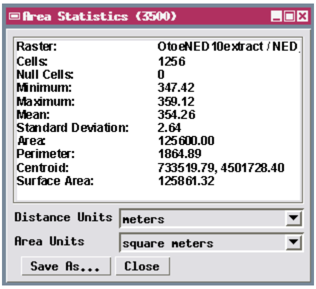
Name: Zoom to Scale

Menu Choices

50000
25000
12000
2 mi

Geospatial Script Overview (continued)





Area Statistics (3500)

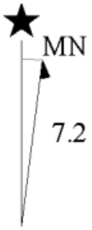
```
Raster:                  OtoeNED10extract / NED
Cells:                   1256
Null Cells:              0
Minimum:                 347.42
Maximum:                 359.12
Mean:                    354.26
Standard Deviation:      2.64
Area:                    125600.00
Perimeter:               1864.89
Centroid:                733519.79, 4501728.40
Surface Area:            125861.32
```

Distance Units: meters

Area Units: square meters

Tool Script: Use a tool script to provide a unique interactive graphic tool to select data in a view window and apply custom processing. A tool script can provide a point, line, or polyline/polygon tool to designate the portion of the data to be processed. It can also provide a dialog window with controls for the process or to show the results of the processing. The tool script template provides prototype functions that you can fill in to define the exact operations you need. You can activate a tool script from the View window using an icon button or from the Tools menu.

SML Layer Script: Use an SML Layer script to create a separate layer in a spatial display view with the Add Script display option. You can use such a script to render a custom cartographic element such as a magnetic declination symbol that can be used in a map layout.



Geospatial Process Script: Use the SML Editor in TNTmips Pro or TNTedit to create custom, standalone scripts to uniquely process your geospatial data. Geospatial process scripts can access and create data in TNTgis Project Files or in external file formats. You can use built-in functions and classes in a geospatial process script to automate a sequence of steps involving a series of TNTgis processes or even external programs. You can also implement your own custom geospatial processing algorithms to process raster or geometric data objects and their associated database information. Many geospatial functions are provided for processing raster and geometric objects. A set of image pipeline classes are provided for fast and efficient processing of images, and pipeline scripts can be constructed using the Script Builder, a graphical design interface (see the Technical Guide entitled *Script Builder Overview*).

Geospatial scripts can provide custom dialog windows that you design to allow interactive setting of program options and even views of the input or output data. They can also utilize TNTmips Job Processing to efficiently perform batch processing and can be run over enterprise networks or using cloud computing resources using TNTscript.

Sample Geospatial Scripts

A large collection of sample geospatial scripts of all types is distributed with TNTgis. On Windows computers the sample scripts are installed automatically in a subdirectory named *smlsamples* in your installation directory. The TNTgis installer for MacOS X provides a separate option to install the sample scripts in a subdirectory named *Geospatial Analysis Scripts*. You can also download the latest version of the sample script library from:

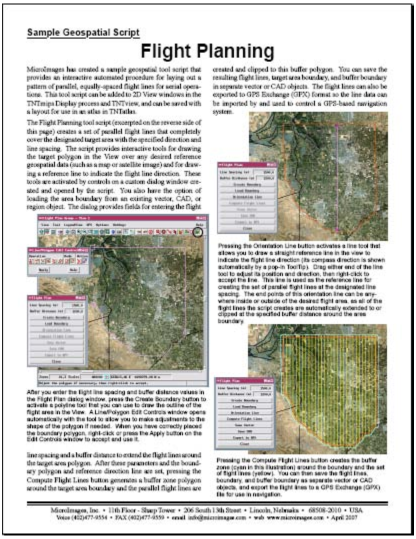
www.microimages.com/sml/index.htm

This Scripting page is also an excellent resource for the latest information about geospatial scripting in the TNT products. It provides links to pages covering different script categories and topics. The individual category pages provide examples and descriptions of different scripts in that category.

Many sample scripts are discussed in illustrated color Technical Guides that provide detailed information about the sample script along with annotated excerpts. Links to these TechGuides can be found with the script descriptions on the script category web pages. You can also access them from the Technical Guide catalog page:

www.microimages.com/documentation/html/category.htm

TechGuides illustrating sample scripts can be found under the following category links: Scripting, CartoScripts, Display Control & GraphTips, Macro Script, and Tool Script, among others.



Sample Geospatial Script

Flight Planning

MicroImages has created a sample geospatial tool script that provides an interactive automated procedure for laying out a pattern of parallel, equally-spaced flight lines for aerial operations. This tool script can be added to 2D View windows in the TNTmips Display process and TNTview, and can be saved with a layout for use in an atlas in TNTatlas.

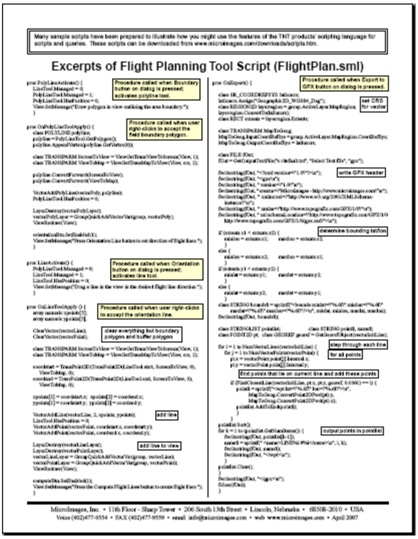
The Flight Planning tool script (excerpted on the reverse side of this page) creates a set of parallel flight lines that completely cover the designated target area with the specified direction and line spacing. The script provides interactive tools for drawing the target polygon in the View over any desired reference geospatial data (such as a map or satellite image) and for drawing a reference line to indicate the flight line direction. These tools are activated by controls on a custom dialog window created and opened by the script. You also have the option of loading the area boundary from an existing vector, CAD, or region object. The dialog provides fields for entering the flight line spacing and a buffer distance to extend the flight lines around the target area polygon. After these parameters and the boundary polygon and reference direction line are set, pressing the Compute Flight Lines button generates a buffer zone polygon around the target area boundary and the parallel flight lines are created and clipped to this buffer polygon. You can save the resulting flight lines, target area boundary, and buffer boundary in separate vector or CAD objects. The flight lines can also be exported to GPS Exchange (GPX) format so the line data can be imported by and used to control a GPS-based navigation system.

After you enter the flight line spacing and buffer distance values in the Flight Plan dialog window, press the Create Boundary button to activate a polyline tool that you can use to draw the outline of the flight area in the View. A Line/Polygon Edit Controls window opens automatically with the tool to allow you to make adjustments to the shape of the polygon if needed. When you have correctly placed the boundary polygon, right-click or press the Apply button on the Edit Controls window to accept and use it.

Pressing the Orientation Line button activates a line tool that allows you to draw a straight reference line in the view to indicate the flight line direction (its compass direction is shown automatically by a pop-in ToolTip). Drag either end of the line tool to adjust its position and direction, then right-click to accept the line. This line is used as the reference line for creating the set of parallel flight lines at the designated line spacing. The end points of this orientation line can be anywhere inside or outside of the desired flight area, as all of the flight lines the script creates are automatically extended to or clipped at the specified buffer distance around the area boundary.

Pressing the Compute Flight Lines button creates the buffer zone (cyan in this illustration) around the boundary and the set of flight lines (yellow). You can then save the flight lines, boundary, and buffer boundary as separate vector or CAD objects, and export the flight lines to a GPS Exchange (GPX) file for use in navigation.

MicroImages, Inc. • 11th Floor - Sharp Tower • 206 South 13th Street • Lincoln, Nebraska • 68508-2010 • USA
Voice (402)477-9554 • FAX (402)477-9559 • email info@microimages.com • web www.microimages.com • April 2007



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.

Excerpts of Flight Planning Tool Script (FlightPlan.sml)

```
proc PolyLineActivate() {
   LineTool.Managed = 0;
   PolyLineTool.Managed = 1;
   PolyLineTool.HasPosition = 0;
   View.SetMessage("Draw polygon in view outlining the area boundary.");
   }

proc OnPolyLineToolApply() {
   class POLYLINE polyline;
   polyline = PolyLineTool.GetPolygon();
   polyline.AppendVertex(polyline.GetVertex(0));

   class TRANSPARM ScreenToView = ViewGetTransViewToScreen(View, 1);
   class TRANSPARM ViewToMap = ViewGetTransMapToView(View, crs, 1);

   polyline.ConvertForward(ScreenToView);
   polyline.ConvertForward(ViewToMap);

   VectorAddPolyLine(vectorPoly, polyline);
   PolyLineTool.HasPosition = 0;

   LayerDestroy(vectorPolyLayer);
   vectorPolyLayer = GroupQuickAddVectorVar(group, vectorPoly);
   ViewRedraw(View);

   orientationBtn.SetEnabled(1);
   View.SetMessage("Press Orientation Line button to set direction of flight lines.");
   }

proc LineActivate() {
   PolyLineTool.Managed = 0;
   LineTool.Managed = 1;
   LineTool.HasPosition = 0;
   View.SetMessage("Drag a line in the view in the desired flight line direction.");
   }

proc OnLineToolApply () {
   array numeric xpoints[5];
   array numeric ypoints[5];

   ClearVector(vectorLine);
   ClearVector(vectorPoint);

   class TRANSPARM ScreenToView = ViewGetTransViewToScreen(View, 1);
   class TRANSPARM ViewToMap = ViewGetTransMapToView(View, crs, 1);

   coordstart = TransPoint2D(TransPoint2D(LineTool.start, ScreenToView, 0),
      ViewToMap, 0);
   coordend = TransPoint2D(TransPoint2D(LineTool.end, ScreenToView, 0),
      ViewToMap, 0);

   xpoints[1] = coordstart.x;  xpoints[2] = coordend.x;
   ypoints[1] = coordstart.y;  ypoints[2] = coordend.y;

   VectorAddLine(vectorLine, 2, xpoints, ypoints);
   LineTool.HasPosition = 0;
   VectorAddPoint(vectorPoint, coordstart.x, coordstart.y);
   VectorAddPoint(vectorPoint, coordend.x, coordend.y);

   LayerDestroy(vectorLineLayer);
   LayerDestroy(vectorPointLayer);
   vectorLineLayer = GroupQuickAddVectorVar(group, vectorLine);
   vectorPointLayer = GroupQuickAddVectorVar(group, vectorPoint);
   ViewRedraw(View);

   computeBtn.SetEnabled(1);
   View.SetMessage("Press the Compute Flight Lines button to create flight lines.");
   }
```

Procedure called when Boundary button on dialog is pressed; activates polyline tool.

Procedure called when user right-clicks to accept the field boundary polygon.

Procedure called when Orientation button on dialog is pressed; activates line tool.

Procedure called when user right-clicks to accept the orientation line.

clear everything but boundary polygon and buffer polygon

add line

add line to view

```
proc OnExport() {
   class SR_COORDREFSYS latloncrs;
   latloncrs.Assign("Geographic2D_WGS84_Deg");
   class REGION2D layerregion = group.ActiveLayer.MapRegion;
   layerregion.ConvertTo(latloncrs);
   class RECT extents = layerregion.Extents;

   class TRANSPARM MapToGeog;
   MapToGeog.InputCoordRefSys = group.ActiveLayer.MapRegion.CoordRefSys;
   MapToGeog.OutputCoordRefSys = latloncrs;

   class FILE fOut;
   fOut = GetOutputTextFile("c:/default.txt", "Select Text file", "gpx");

   fwritestring(fOut, "<?xml version=\"1.0\"?>\n");
   fwritestring(fOut, "<gpx\n");
   fwritestring(fOut, " version=\"1.0\"\n");
   fwritestring(fOut, " creator=\"MicroImages - http://www.microimages.com\"\n");
   fwritestring(fOut, " xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-
      instance\"\n");
   fwritestring(fOut, " xmlns=\"http://www.topografix.com/GPX/1/0\"\n");
   fwritestring(fOut, " xsi:schemaLocation=\"http://www.topografix.com/GPX/1/0
      http://www.topografix.com/GPX/1/0/gpx.xsd\">\n");

   if (extents.x1 < extents.x2) {
      minlon = extents.x1;      maxlon = extents.x2;
      }
   else {
      minlon = extents.x2;      maxlon = extents.x1;
      }
   if (extents.y1 < extents.y2) {
      minlat = extents.y1;      maxlat = extents.y2;
      }
   else {
      minlat = extents.y2;      maxlat = extents.y1;
      }
   class STRING boundsS = sprintf("<bounds minlat=\"%.6f\" minlon=\"%.6f\"
         maxlat=\"%.6f\" maxlon=\"%.6f\"/>\n", minlat, minlon, maxlat, maxlon);
   fwritestring(fOut, boundsS);

   class STRINGLIST pointlist;            class STRING pointS, nameS;
   class POINT2D pt;   class GEOREF georef = GetGeorefObject(vectorInitLine);

   for i = 1 to NumVectorLines(vectorInitLine) {
      for j = 1 to NumVectorPoints(vectorPoint) {
         pt.x = vectorPoint.point[j].Internal.x;
         pt.y = vectorPoint.point[j].Internal.y;

         if (FindClosestLine(vectorInitLine, pt.x, pt.y, georef, 0.0001) == i) {
            pointS = sprintf("<rtept lat=\"%.6f\" lon=\"%.6f\">\n",
                  MapToGeog.ConvertPoint2DFwd(pt).y,
                  MapToGeog.ConvertPoint2DFwd(pt).x);
            pointlist.AddToEnd(pointS);
            }
         }
      pointlist.Sort();
      for k = 1 to (pointlist.GetNumItems()) {
         fwritestring(fOut, pointlist[k-1]);
         nameS = sprintf(" <name>LINE%d-#%d</name>\n", i, k);
         fwritestring(fOut, nameS);
         fwritestring(fOut, "</rtept>\n");
         }
      pointlist.Clear();
      }
   fwritestring(fOut, "</gpx>\n");
   fclose(fOut);
   }
```

Procedure called when Export to GPX button on dialog is pressed.

set CRS for vector

write GPX header

determine bounding lat/lon

step through each line

for all points

find points that lie on current line and add these points

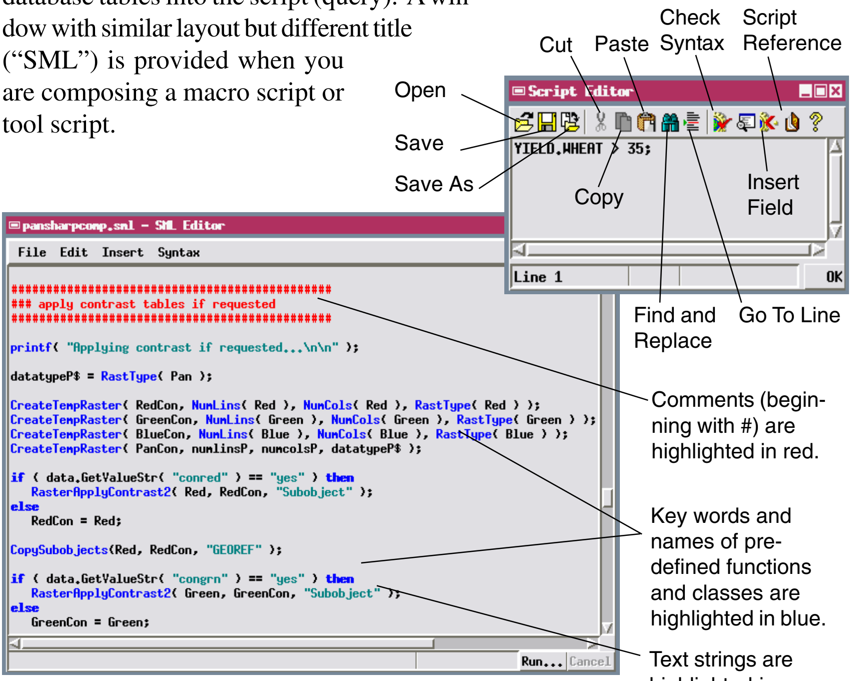
output points in pointlist

MicroImages, Inc. • 11th Floor - Sharp Tower • 206 South 13th Street • Lincoln, Nebraska • 68508-2010 • USA
Voice (402)477-9554 • FAX (402)477-9559 • email info@microimages.com • web www.microimages.com • April 2007

Creating and Editing Scripts

A script can be anything from a single expression to a long structured program with user-defined functions, nested loops, and branching constructs. Regardless of how simple or complex your script is, the script editor windows provided in TNTgis make it easy to compose, edit, and check the script. These editor windows provide simple tools for inserting operators, SML keywords, and predefined functions into your script. Spaces and tabs are ignored when a script is interpreted by SML, so you can use spaces and indents freely to make the script more readable. You can add comments to document script logic and structure. Automatic color highlighting for SML keywords, comments, and text strings enhances readability as you work. A syntax checker is also provided so you can make sure that your script conforms to the syntax rules of the SML programming language.

The name and layout of the editor window varies depending on the program environment in which you are working. For example, when you are composing a database selection query to mark elements in a vector object you are displaying, the Select by Query window (small window shown below) includes an Insert Field icon button to let you easily insert the references to fields in associated database tables into the script (query). A window with similar layout but different title ("SML") is provided when you are composing a macro script or tool script.

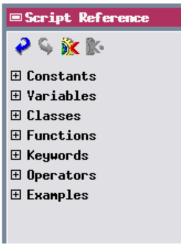

Check Syntax
Script Reference
Cut
Paste
Open
Save
Save As
Copy
Insert Field
Find and Replace
Go To Line
Script Editor
Line 1
OK
pansharpcomp.sml - SML Editor
File
Edit
Insert
Syntax
Run...
Cancel
Comments (beginning with #) are highlighted in red.
Key words and names of predefined functions and classes are highlighted in blue.
Text strings are highlighted in cyan.

```
YIELD.WHEAT > 35;
```

```
##############################################
### apply contrast tables if requested
##############################################

printf( "Applying contrast if requested...\n\n" );

datatypeP$ = RastType( Pan );

CreateTempRaster( RedCon, NumLins( Red ), NumCols( Red ), RastType( Red ) );
CreateTempRaster( GreenCon, NumLins( Green ), NumCols( Green ), RastType( Green ) );
CreateTempRaster( BlueCon, NumLins( Blue ), NumCols( Blue ), RastType( Blue ) );
CreateTempRaster( PanCon, numlinsP, numcolsP, datatypeP$ );

if ( data.GetValueStr( "conred" ) == "yes" ) then
   RasterApplyContrast2( Red, RedCon, "Subobject" );
else
   RedCon = Red;

CopySubobjects(Red, RedCon, "GEOREF" );

if ( data.GetValueStr( "congrn" ) == "yes" ) then
   RasterApplyContrast2( Green, GreenCon, "Subobject" );
else
   GreenCon = Green;
```

SML Editor window for standalone geospatial process scripts.

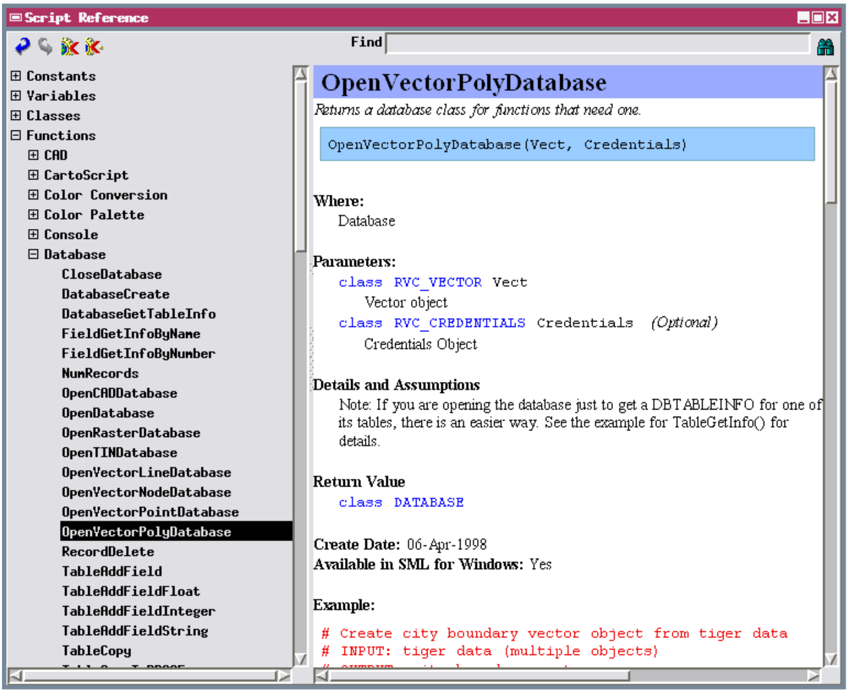
Integrated Script Reference



Script Reference

⊞ Constants
⊞ Variables
⊞ Classes
⊞ Functions
⊞ Keywords
⊞ Operators
⊞ Examples

SML provides hundreds of predefined elements including constants, arithmetic and logical operators, and mathematical and geospatial functions and classes. Documentation on these elements is provided in the SML Editor's Script Reference window. The left side of the window is a tree-control that lists these and other script element categories; click on the plus sign icon next to an entry to show a listing of items in that category. Functions and classes are further divided into subcategories that must also be expanded to show their elements. When you select an individual script element in the list, the pane in the right side of the Script Reference window documents the syntax and use of the element. For functions, the documentation lists the required and optional parameters, the value returned, any details and assumptions, and in most cases provides an example of its use. Class members and methods (class functions) are documented in similar fashion. You can also use the Insert button on the Script Reference window to insert constants, functions, keywords, and operators into your script.



Script Reference

Find

⊞ Constants
⊞ Variables
⊞ Classes
⊟ Functions
  ⊞ CAD
  ⊞ CartoScript
  ⊞ Color Conversion
  ⊞ Color Palette
  ⊞ Console
  ⊟ Database
    CloseDatabase
    DatabaseCreate
    DatabaseGetTableInfo
    FieldGetInfoByName
    FieldGetInfoByNumber
    NumRecords
    OpenCADDatabase
    OpenDatabase
    OpenRasterDatabase
    OpenTINDatabase
    OpenVectorLineDatabase
    OpenVectorNodeDatabase
    OpenVectorPointDatabase
    OpenVectorPolyDatabase
    RecordDelete
    TableAddField
    TableAddFieldFloat
    TableAddFieldInteger
    TableAddFieldString
    TableCopy

OpenVectorPolyDatabase

Returns a database class for functions that need one.

```
OpenVectorPolyDatabase(Vect, Credentials)
```

Where:
Database

Parameters:

```
class RVC_VECTOR Vect
```
Vector object

```
class RVC_CREDENTIALS Credentials
```
(Optional)
Credentials Object

Details and Assumptions
Note: If you are opening the database just to get a DBTABLEINFO for one of its tables, there is an easier way. See the example for TableGetInfo() for details.

Return Value

```
class DATABASE
```

Create Date: 06-Apr-1998
Available in SML for Windows: Yes

Example:

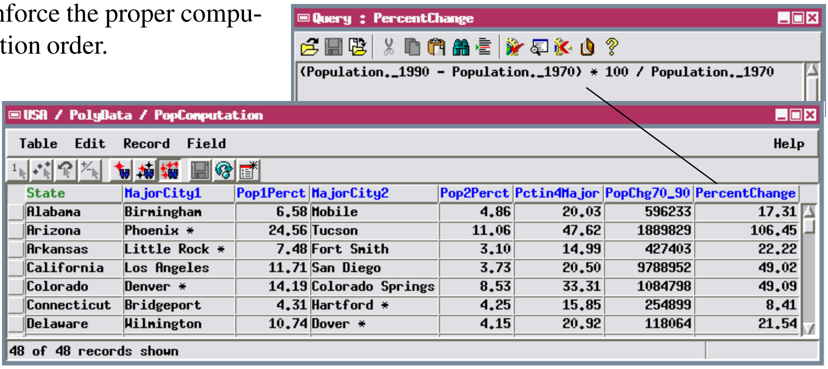
```
# Create city boundary vector object from tiger data
# INPUT: tiger data (multiple objects)
```

Depending on which SML Editor you are using, open the Script Reference window from any of the options on the Insert menu or by pressing the Script Reference icon button.

Scripts for Database Computed Fields

One of the simplest yet very useful applications of scripting in TNTgis is creating *computed fields* in database tables. A computed field does not contain static data but rather the result of an expression that can combine information from other fields in that table or in other related tables. The expression is evalutaed separately for each record in the table, so different values can be returned by the same expression in different records. There are two types of computed fields: *computed numeric fields* return a numeric result and *computed text* fields return the result as a text string. Names for computed fields are shown in blue in table windows, so it is easy to distinguish them from standard data fields. You can create computed fields and their expressions using the Table Properties window for any database table, as described in the Technical Guide entitled *Table Properties Window*.

Two examples of computed numeric fields are illustrated below. The first one uses decadal population values for major cities to compute the percent population change from 1970 to 1990. This expression uses standard arithmetic operators: subtraction (-), multiplication (*), and division (/), along with parentheses to enforce the proper computation order.

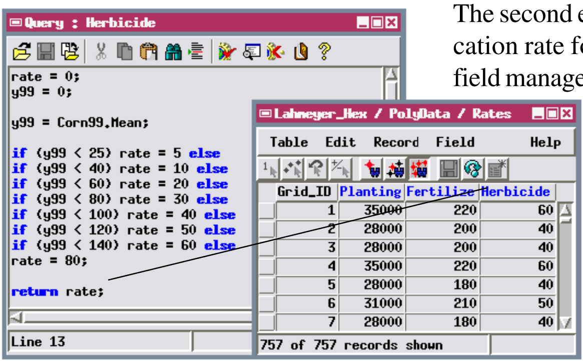


```
(Population._1990 - Population._1970) * 100 / Population._1970
```

State	MajorCity1	Pop1Perct	MajorCity2	Pop2Perct	Pctin4Major	PopChg70_90	PercentChange
Alabama	Birmingham	6.58	Mobile	4.86	20.03	596233	17.31
Arizona	Phoenix *	24.56	Tucson	11.06	47.62	1889829	106.45
Arkansas	Little Rock *	7.48	Fort Smith	3.10	14.99	427403	22.22
California	Los Angeles	11.71	San Diego	3.73	20.50	9788952	49.02
Colorado	Denver *	14.19	Colorado Springs	8.53	33.31	1084798	49.09
Connecticut	Bridgeport	4.31	Hartford *	4.25	15.85	254899	8.41
Delaware	Wilmington	10.74	Dover *	4.15	20.92	118064	21.54

48 of 48 records shown

The second example returns an application rate for herbicide for different field management grid polygons. The rate is determined from the previous year's corn yield value for that polygon, using a logical if-else structure to set higher application rates for higher corn yields.



```
rate = 0;
y99 = 0;

y99 = Corn99.Mean;

if (y99 < 25) rate = 5 else
if (y99 < 40) rate = 10 else
if (y99 < 60) rate = 20 else
if (y99 < 80) rate = 30 else
if (y99 < 100) rate = 40 else
if (y99 < 120) rate = 50 else
if (y99 < 140) rate = 60 else
rate = 80;

return rate;
```

Line 13

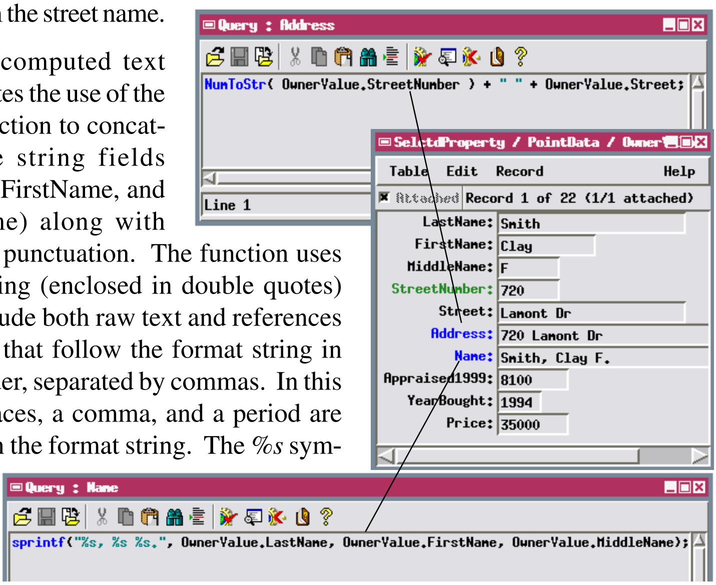
Grid_ID	Planting	Fertilize	Herbicide
1	35000	220	60
2	28000	200	40
3	28000	200	40
4	35000	220	60
5	28000	180	40
6	31000	210	50
7	28000	180	40

757 of 757 records shown

Scripts for Computed Text Fields

A computed text field script can be as simple as a reference to a string field in another related table, or it can be a more complex expression that concatenates other fields or constructs a string from text and numeric data. Raw text strings must be enclosed in double quotes in a string expression (and in SML generally). You can concatentate strings by simply using the addition operator (+) or use the *sprintf()* function to create a formatted string. String expressions can include references to numeric fields, but values returned by those fields must be converted to a number by the string expression.

The table illustrated below includes two computed text fields (blue) that merge information from other fields in the same table. The Address computed text field merges the information from the StreetNumber and Street fields. The former is an integer field, so its value is converted to a string using the *NumToStr()* function. The Street field is a string field, so its field reference can be used directly in the expression. The resulting strings from these fields are concatenated by simple addition, along with a text string consisting only of a space to separate the street number from the street name.



```
NumToStr( OwnerValue.StreetNumber ) + " " + OwnerValue.Street;
```

☒ Attached Record 1 of 22 (1/1 attached)

LastName:	Smith
FirstName:	Clay
MiddleName:	F
StreetNumber:	720
Street:	Lamont Dr
Address:	720 Lamont Dr
Name:	Smith, Clay F.
Appraised1999:	8100
YearBought:	1994
Price:	35000

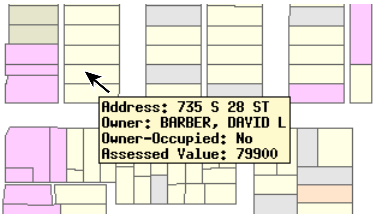
```
sprintf("%s, %s %s.", OwnerValue.LastName, OwnerValue.FirstName, OwnerValue.MiddleName);
```

The Name computed text field illustrates the use of the *sprintf()* function to concatenate three string fields (LastName, FirstName, and MiddleName) along with spacing and punctuation. The function uses a format string (enclosed in double quotes) that can include both raw text and references to variables that follow the format string in the same order, separated by commas. In this example spaces, a comma, and a period are embedded in the format string. The *%s* symbol is the reference to a string value; other symbols are available to refer to integer and floating point fields. For example, the symbol *%d* is used to reference an integer value and convert it to a string. The expression below produces the same result as the string expression shown above for the Address field:

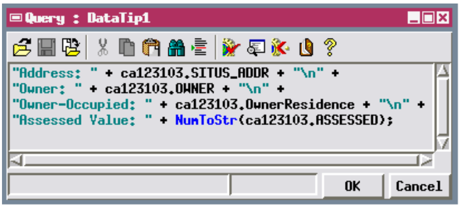
```
sprintf("%d %s", OwnerValue.StreetNumber, OwnerValue.Street);
```

String Expressions for DataTips

DataTips in View windows can automatically reveal important attribute information for elements when the cursor pauses over them. You can set up DataTips for a displayed layer using its Layer Controls window, as described in the *DataTips* Technical Guide. The DataTip information for each layer is read from a designated field in an attached database table. You can include information from multiple fields by reading the DataTip from a computed text field that merges and formats information from those fields. The expression can include newline codes to create a multiline DataTip and a text label for each line. The *Managing Geoattributes* tutorial includes more information on setting up and using multiline DataTips from computed text fields.



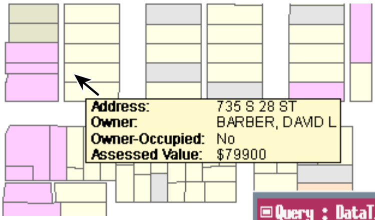
```
Address: 735 S 28 ST
Owner: BARBER, DAVID L
Owner-Occupied: No
Assessed Value: 79900
```



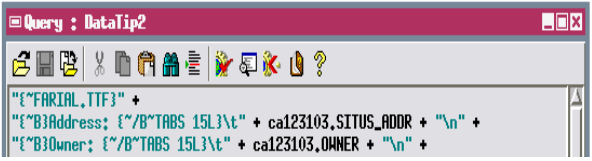
```
"Address: " + ca123103.SITUS_ADDR + "\n" +
"Owner: " + ca123103.OWNER + "\n" +
"Owner-Occupied: " + ca123103.OwnerResidence + "\n" +
"Assessed Value: " + NumToStr(ca123103.ASSESSED);
```

The illustrations above show an example of a multiline DataTip presenting information from property parcel polygons in a vector object. The string expression uses newline characters to create four lines of text, each of which begins with a text label for the attribute shown on that line. The assessed value is read from a numeric field and so must be converted to a string for use in the expression.

Text formatting control codes are also available that you can use to set fonts, create bold text, and insert tabs. Each formatting code begins with the tilde character (~). Codes must be enclosed by a set of curly brackets ({ })and included within the text strings they modify, or within quotes by themselves. The illustrations below show the same information as in the DataTip above, but with formatting added.



Address:	735 S 28 ST
Owner:	BARBER, DAVID L
Owner-Occupied:	No
Assessed Value:	$79900



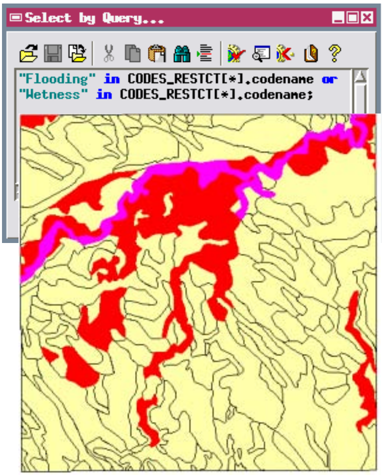
```
"{~FARIAL.TTF}" +
"{~B}Address: {~/B~TABS 15L}\t" + ca123103.SITUS_ADDR + "\n" +
"{~B}Owner: {~/B~TABS 15L}\t" + ca123103.OWNER + "\n" +
```

Arial is set as the font, labels are set to be bold, and a tab stop is set for each line and a tab character (\t) inserted to left-align the values. A complete list of formatting codes is presented in the Quick Guide entitled *Text Style Formatting Codes*.

Database Queries

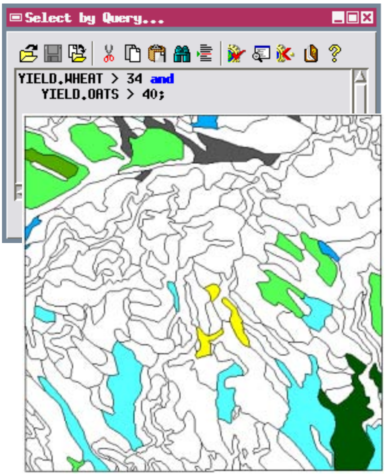
A database query is a script that is used to search the records in a database to find those that match specified criteria. In many cases the database records are attached to elements in geospatial objects, so the query in reality is used to identify one or more elements in that object. A query can reference fields in a single table or in multiple related tables.

You can use database queries in several ways in TNTgis. When you display or edit geometric objects (vector, CAD, shape, and TIN), you can use a query to temporarily mark (highlight in color) all elements that match your selection criteria. You can apply some editing operations (such as deletion) to the entire marked set or just to the current active element in the set. Icon buttons on the View window allow you to step through the marked set if needed.



```
"Flooding" in CODES_RESTCT[*].codename or
"Wetness" in CODES_RESTCT[*].codename;
```

Database query used to temporarily mark (in red and magenta colors) displayed polygons that match the selection criteria.



```
YIELD.WHEAT > 34 and
   YIELD.OATS > 40;
```

Database query used to select polygons to be displayed (filled in this case). All line elements are displayed, so boundaries of unfilled polygons are visible.

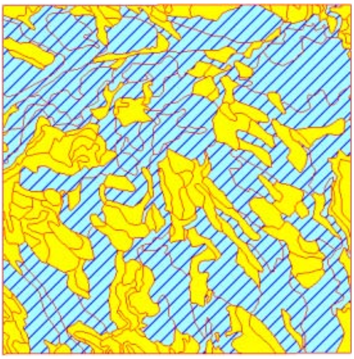
When you set up display parameters for a geometric object, you can use a query to determine which elements in a layer are displayed. For vector objects you can separately query the point, line, and polygon elements, though a single query can reference attributes of different element types if they are topologically related (such as lines and nodes, or lines and polygons). Some geometric object processes (such as extraction, buffer zone generation, and others) also allow you to use queries to determine the elements to use in the operation. Query procedures are discussed in detail with many examples in the tutorial booklet entitled *Building and Using Queries*.

Style Scripts and CartoScripts

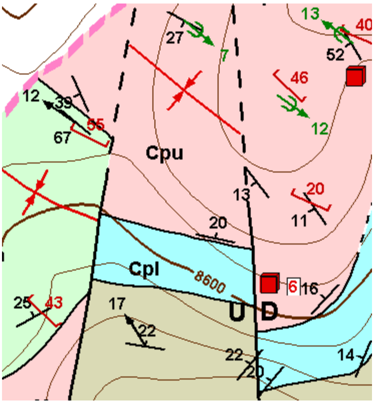
When you set up display styling for elements in geometric objects, the Style by Attribute option allows you to set up different styles for elements with different values in a single designated field in an attached database table. If you would like to set up styles based on a more complex assessment of the element attributes, you can use the By Script style option.

A style script can utilize named styles that have already been created and reference existing line patterns and polygon fill patterns. It can also directly set style components such as line width and the colors used for point symbols, lines, and polygon borders and fills. A special set of SML keywords are provided for use in style queries. The *Building and Using Queries* tutorial booklet provides an introduction to creating style scripts.

```
DrawBorder = 1;
DrawColor$ = "red";
if ( POLYSTATS.Area < 200000 ) then
   begin
      FillInside = 1;
      FillColor$ = "yellow";
   end
else
   begin
      FillInside = 1;
      FillBitmapPatt = 1;
      FillPatt$ = "BitmapPatt2";
   end
```



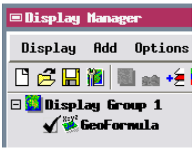
A style script that sets up different polygon fill styles based on an area threshold.



Geologic point and line symbols rendered by CartoScript.

CartoScripts are style scripts that make use of special SML functions to create complex symbology for points or lines. The CartoScript functions allow you to directly draw symbols, navigate and draw along line elements, and create text labels as part of the symbology. Functions are provided to draw lines, arcs, circles, and 3D cubes and cylinders. You can use CartoScripts to create specialized symbology for maps in particular disciplines, such as geology or environmental science. The tutorial booklet *Using Cartoscripts* provides a thorough introduction to creating and using CartoScripts.

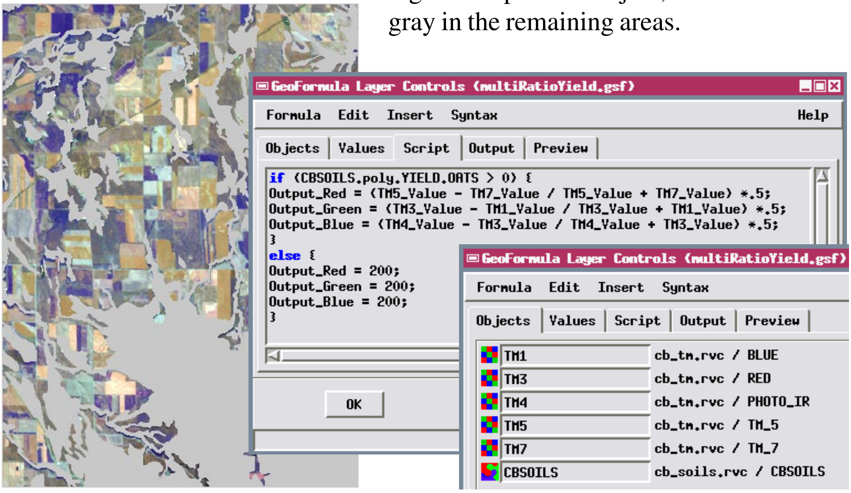
GeoFormula Display Layers



A GeoFormula Display Layer is a computed layer that combines information from one or more spatial objects (raster or vector). The result is rendered to the View as a "virtual raster" layer; there is no need to create and save an output object. The result can be either grayscale (values ranging from 0 to 255) or RGB color (range of 0 to 255 for each of the three color components). GeoFormula layers can be displayed along with any other type of layer in the View. You can add a GeoFormula layer to a View by using the Add / Layer menu cascade in the Display Manager.

The GeoFormula Layer Controls window provides a simplified programming interface in which selecting input objects automatically creates corresponding script variables and there are predefined variables for the output. The interface also provides push-button creation of numerical variables ("values"). You can thus easily compose a script that combines values from the various input objects in the desired way. With their simple interface and virtual output, GeoFormula layers are an ideal means of prototyping data analysis algorithms and testing different data fusion methods.

The tutorial booklet entitled *Using Geospatial Formulas* provides an introduction to GeoFormula display layers and provides many sample GeoFormulas. The example shown below computes a color layer in which each of the three color components is computed from the normalized ratio of two Landsat image bands, but only for areas that have a non-zero potential crop yield value in a corresponding soil map vector object; the color is set to gray in the remaining areas.

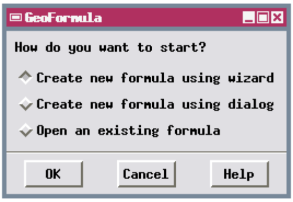


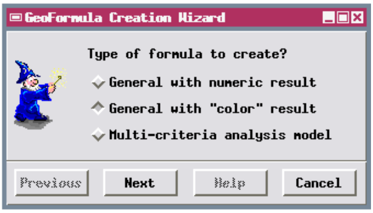
```
if (CBSOILS.poly.YIELD.OATS > 0) {
Output_Red = (TM5_Value - TM7_Value / TM5_Value + TM7_Value) *.5;
Output_Green = (TM3_Value - TM1_Value / TM3_Value + TM1_Value) *.5;
Output_Blue = (TM4_Value - TM3_Value / TM4_Value + TM3_Value) *.5;
}
else {
Output_Red = 200;
Output_Green = 200;
Output_Blue = 200;
}
```

TM1	cb_tm.rvc / BLUE
TM3	cb_tm.rvc / RED
TM4	cb_tm.rvc / PHOTO_IR
TM5	cb_tm.rvc / TM_5
TM7	cb_tm.rvc / TM_7
CBSOILS	cb_soils.rvc / CBSOILS

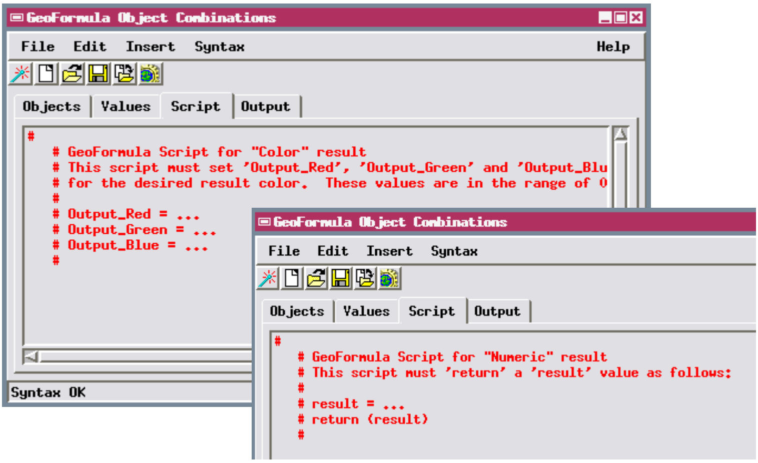
GeoFormula Process

A GeoFormula script that you create for display purposes can also be used in the TNTmips GeoFormula process (Script / GeoFormula from TNTmips menu) to create a raster object saved in a Project File. As in the Display process, you can combine objects to produce a simple numeric result (saved as a grayscale raster) or a color result (saved as an RGB composite raster). In addition, the GeoFormula process implements a specialized analysis procedure called multi-criteria decision analysis that is described on the next page.


GeoFormula
How do you want to start?
Create new formula using wizard
Create new formula using dialog
Open an existing formula
OK
Cancel
Help


GeoFormula Creation Wizard
Type of formula to create?
General with numeric result
General with "color" result
Multi-criteria analysis model
Previous
Next
Help
Cancel

The GeoFormula process further simplifies setup by providing a wizard interface for creating new GeoFormulas. When you use the wizard, you are prompted to select the input objects, then for the type of GeoFormula you want to create (numeric, color, or multi-criteria). When you choose either of the first two options, the wizard creates a stub script that includes guideline comments on how that type of script must be structured and what type of values must be returned. You can then edit this stub script, removing the comment characters from the functional lines and adding the statements needed to perform the desired computation.



```
#
   # GeoFormula Script for "Color" result
   # This script must set 'Output_Red', 'Output_Green' and 'Output_Blu
   # for the desired result color.  These values are in the range of 0
   #
   # Output_Red = ...
   # Output_Green = ...
   # Output_Blue = ...
   #
```

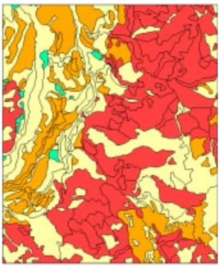
```
#
   # GeoFormula Script for "Numeric" result
   # This script must 'return' a 'result' value as follows:
   #
   # result = ...
   # return (result)
   #
```

The output tabbed panel includes controls for determining the extents, projection, and cell size or line/column dimensions of the output raster.

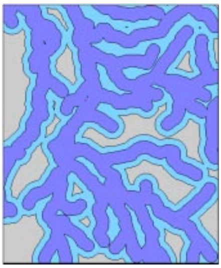
Multi-Criteria Decision Analysis

Multi-criteria decision analysis (MCDA) is a quantitative approach to evaluating decision problems that involve multiple variables (criteria). The GeoFormula process allows you to apply a spatial form of MCDA using cell values in raster objects and the database attributes of polygons in vector objects to represent the spatial variations of individual decision criteria.

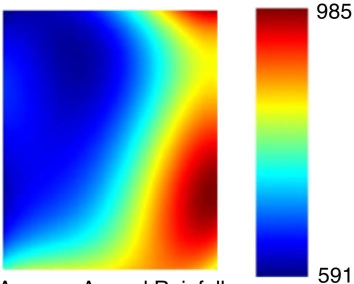
When you choose *Multi-criteria analysis model* from the GeoFormula Creation Wizard, the wizard prompts for the required input parameters and automatically generates the corresponding GeoFormula script. The process uses the most common form of MCDA, weighted linear combination. You first assign a “relative importance” weighting factor to each of the spatial objects. For each input object you are then prompted to select or set up a table of numerical scores that indicate the relative benefit (or cost) you want to assign to particular values or value ranges in the object. The MCDA script computes an overall score for each output cell location by multiplying the weighting factor for each object by the score for that location and summing the products from all of the input objects.



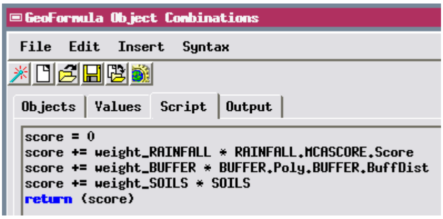
Soil Erodibility
(weight = 40%)



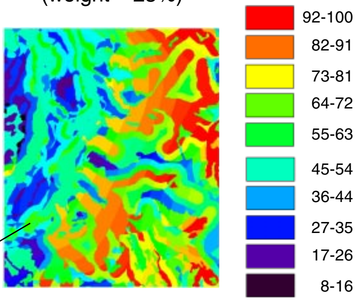
Distance from Streams
(weight = 35%)



Average Annual Rainfall
(weight = 25%)



```
score = 0
score += weight_RAINFALL * RAINFALL.MCASCORE.Score
score += weight_BUFFER * BUFFER.Poly.BUFFER.BuffDist
score += weight_SOILS * SOILS
return (score)
```



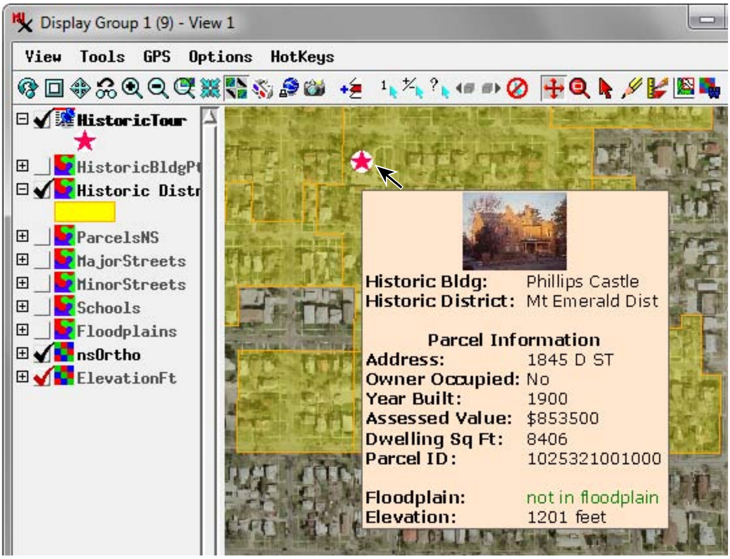
MCDA Result: high score indicates greater hazard.

The MCDA example illustrated here evaluates areas where clearcut logging could have adverse affects on streams due to sediment runoff. The criteria objects used are a vector soil map scored by erodibility, a vector map of stream buffer zones, and a raster with local variation in average annual rainfall.

Display Control Scripts for DataTips

Display Control Scripts allow you to add custom effects and procedures to Views of saved groups and layouts. A display control script runs automatically when its group or layout is loaded in the Display process, and provides access to events such as the beginning and end of rendering the group or layout; drawing, selecting, creating, or removing particular layers; showing and hiding a DataTip; and many others. Each event has a corresponding prototype function in the Display Control Script function group, and each such function is automatically called by its associated event. You can insert these prototype functions into your display control script and add your own code for the function to take the desired actions when the associated event occurs in the display. You can create and add a display control script to a group or layout by choosing Edit Control Script from the Display menu of the Display Manager window.

A simple use of a display control script is to apply custom formatting to text in a DataTip, including font styles, alignment, and background color, as shown by the illustration below. You can format DataTip text using MicroImages formatting codes or HTML markup and Cascading Style Sheet (CSS) methods. Sample scripts that illustrate these techniques are described in the Technical Guides entitled *Add Styling to DataTips* and *Design DataTips using HTML.*



Historic Bldg: Phillips Castle
Historic District: Mt Emerald Dist

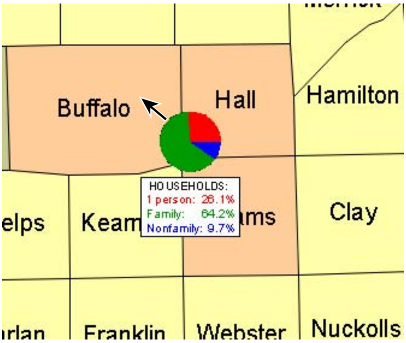
Parcel Information
Address: 1845 D ST
Owner Occupied: No
Year Built: 1900
Assessed Value: $853500
Dwelling Sq Ft: 8406
Parcel ID: 1025321001000

Floodplain: not in floodplain
Elevation: 1201 feet

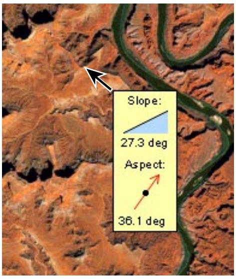
View of a saved group with DataTip formatted using HTML markup and CSS styles defined in a display control script.

Display Control Scripts using Graphics

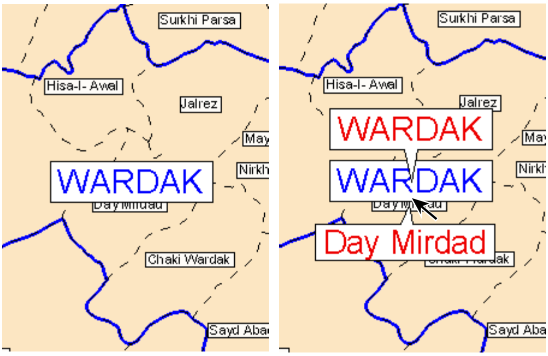
You can program a display control script to do more with DataTip events than simply format text. Each DataTip is initially composed as a small image in computer memory, and a display control script can use SML drawing functions to draw graphics into that image. The DataTip function prototypes also provide access to the location of the cursor on screen when the DataTip is requested. This location can then be used to obtain the cell value at that location in a raster layer or one or more attributes for the corresponding element in a geometric layer in the View. The script can use these context-sensitive values, or values computed from them, as the basis for creating and drawing bar graphs, pie charts, and other graphical representations of the data at that location. The script can also create a mask for the background area of the DataTip to make it transparent and suppress the DataTip frame. The illustrations on this page show some of the graphic applications that are provided by sample display control scripts. These scripts are documented in the TechGuides entitled *Enhance DataTips using Scripts*.



The PieChartGraphTip sample script creates a DataTip with transparent background and no frame. It shows a pie chart and text that are dynamically rendered to show census information for county polygons drawn from different fields in an associated database table.



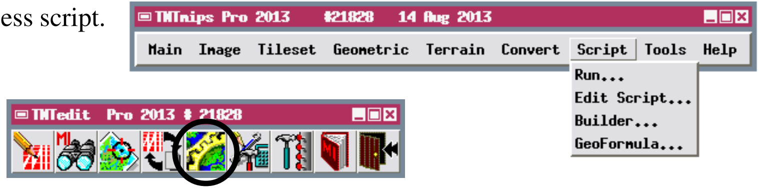
The SlopeAspectGraphTip sample script computes slope and aspect for the cursor location from a hidden elevation raster and portrays the results graphically.



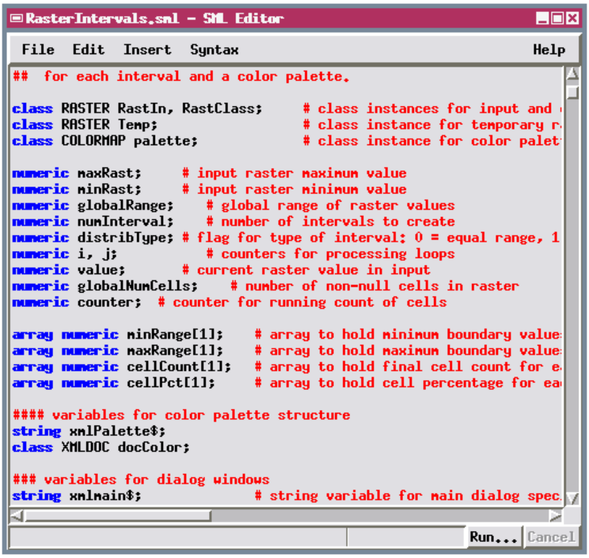
The Label_management sample script repositions and enlarges text (red in the right-hand illustration) for overlapping label elements in a vector object.

Geospatial Process Scripts

Geospatial process scripts execute standalone processes that run independently of other interactive TNTgis components. You can create or run a geospatial process script via the SML process in TNTmips Pro or TNTedit. To access the SML process, use the Script menu options from the TNTmips main menu or the menu options provided by the SML icon button on the TNTedit. The Run menu option allows you to select and run an existing geospatial process script. The Edit Script option opens the SML Editor window, where you can create, view, edit, and run a geospatial process script.



Geospatial process scripts can access, transform, and create virtually any form of geospatial data, including all of the spatial object types used in TNTgis as well as any associated database information. A process script can also import data from or export data to a host of external file formats. (Export capability is not available from SML scripts run in TNTmips Free.)



```
##  for each interval and a color palette.

class RASTER RastIn, RastClass;      # class instances for input and
class RASTER Temp;                   # class instance for temporary r
class COLORMAP palette;              # class instance for color palet

numeric maxRast;     # input raster maximum value
numeric minRast;     # input raster minimum value
numeric globalRange;    # global range of raster values
numeric numInterval;    # number of intervals to create
numeric distribType; # flag for type of interval: 0 = equal range, 1
numeric i, j;           # counters for processing loops
numeric value;       # current raster value in input
numeric globalNumCells;    # number of non-null cells in raster
numeric counter;  # counter for running count of cells

array numeric minRange[1];    # array to hold minimum boundary value
array numeric maxRange[1];    # array to hold maximum boundary value
array numeric cellCount[1];   # array to hold final cell count for e
array numeric cellPct[1];     # array to hold cell percentage for ea

#### variables for color palette structure
string xmlPalette$;
class XMLDOC docColor;

### variables for dialog windows
string xmlmain$;              # string variable for main dialog spec
```

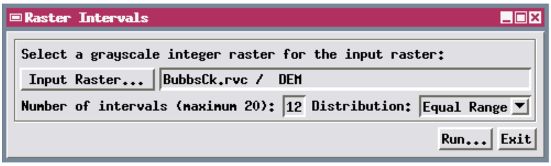
You can set up a script to use interactive dialogs so that those unfamiliar with the script can easily select input and output objects and supply any necessary processing choices and parameters. Popup dialogs created by predefined SML functions may suffice for this purpose in simple processing scripts. For more complex script processes, you can create and manage custom dialog windows to control the operation of the script, as described in the tutorial entitled *Building Dialogs in SML.*

Create Unique Geospatial Processes

STEPS

- ☑ select Script / Run from the TNTmips menu
- ☑ use the Select File dialog to navigate to the StandAlone directory in the SML sample script collection and select RasterIntervals.sml
- ☑ press [Input Raster...] on the Raster Intervals window
- ☑ select raster DEM from the BUBBSCK Project File in the SCRIPTING sample data directory
- ☑ set the Number of Intervals field to 12 and press [Run...]
- ☑ press [Preview result...] on the Distribution window
- ☑ press [Close] on the Preview window
- ☑ press [OK] on the Distribution window
- ☑ use the Select Object dialog to name a new Project File and output raster object
- ☑ press [Exit] on the Raster Intervals window

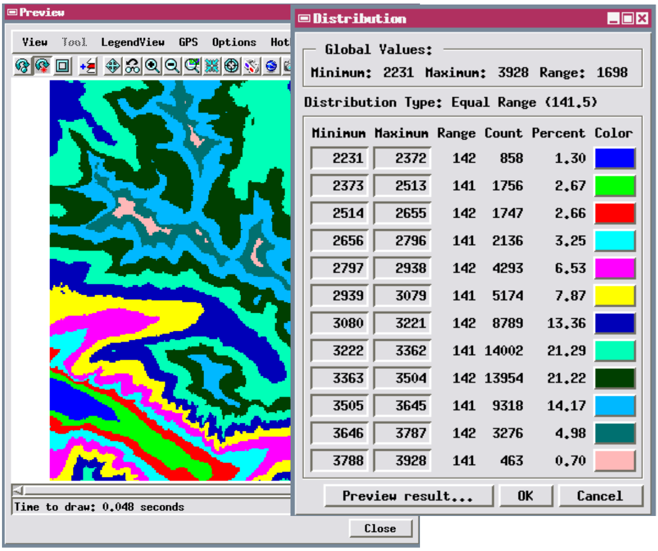
You can use a geospatial process script to implement a custom geospatial data operation. The RASTERINTERVALS script example illustrated here categorizes a grayscale raster into a user-specified number of grayscale-value intervals. The intervals can be set up using either equal value range or equal cell count per interval, but the resulting ranges can also be manually edited in the Distribution window. The script output is a categorical raster with a single value for each interval and a color palette for display. This script process can be used to subdivide an elevation raster into elevation ranges, a slope raster into ranges of different slope magnitude, and so on. The dialog and Preview windows are created and managed by this processing script.



Select a grayscale integer raster for the input raster:

BubbsCk.rvc / DEM

Number of intervals (maximum 20): 12 Distribution: Equal Range



Time to draw: 0.048 seconds

Global Values:

Minimum: 2231 Maximum: 3928 Range: 1698

Distribution Type: Equal Range (141.5)

Minimum	Maximum	Range	Count	Percent	Color
2231	2372	142	858	1.30	■
2373	2513	141	1756	2.67	■
2514	2655	142	1747	2.66	■
2656	2796	141	2136	3.25	■
2797	2938	142	4293	6.53	■
2939	3079	141	5174	7.87	■
3080	3221	142	8789	13.36	■
3222	3362	141	14002	21.29	■
3363	3504	142	13954	21.22	■
3505	3645	141	9318	14.17	■
3646	3787	142	3276	4.98	■
3788	3928	141	463	0.70	■

The Theme Mapping option for displaying raster layers in the TNT products provides a temporary visual equivalent of the permanent categorization performed by this script.

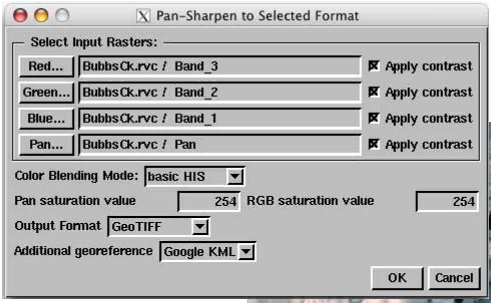
Automate a Processing Workflow

Many of the standard geospatial operations you perform interactively in TNTmips have corresponding predefined functions or class methods in SML. The output of one such operation can be used as input to another, allowing a script to tie together a number of discrete geospatial operations into one streamlined process.

The PIPELINEPANSHARP sample script illustrates this concept. The inputs to the script are three low-resolution raster bands of a multispectral image and a higher-resolution panchromatic raster. These rasters are processed to create a pan-sharpened color-composite with color content from the multispectral bands and spatial detail from the panchromatic (pan) band. The script uses a number of predefined compiled operations to handle the major processing steps. These steps include optionally applying contrast-enhancement to the input rasters, compositing the multispectral bands and resampling to match the extents of the pan band, multiresolution fusion using a choice of color blending methods to make the pan-sharpened image, and saving the image as an RVC raster or GeoTIFF, PNG, JP2, or JPEG file. Because each of these complex steps is handled by a compiled class method, script execution is fast and the script has a relatively simple structure.

Image processing operations are particularly easy to chain together in geospatial scripts because the same modular pipeline processing architecture used in the TNT products is also integrated into SML.

STEPS
- ☑ select Script / Run from the TNTmips menu
- ☑ select PIPELINEPANSHARP.SML from the PIPELINE directory in the SML sample script collection
- ☑ in the script's dialog window, press [Red...]
- ☑ select raster BAND_3 from the BUBBSCK Project File
- ☑ press [Green...] and select raster BAND_2
- ☑ press [Blue...] and select raster BAND_1
- ☑ press [Pan...] and select raster PAN
- ☑ choose a Color Blending Mode and an Output Format
- ☑ press [OK] to name the output object or file and run the process
- ☑ press [Close] to exit the script


Pan-Sharpen to Selected Format

Select Input Rasters:

Red...	BubbsCk.rvc / Band_3	☒ Apply contrast
Green...	BubbsCk.rvc / Band_2	☒ Apply contrast
Blue...	BubbsCk.rvc / Band_1	☒ Apply contrast
Pan...	BubbsCk.rvc / Pan	☒ Apply contrast

Color Blending Mode: basic HIS

Pan saturation value 254 RGB saturation value 254

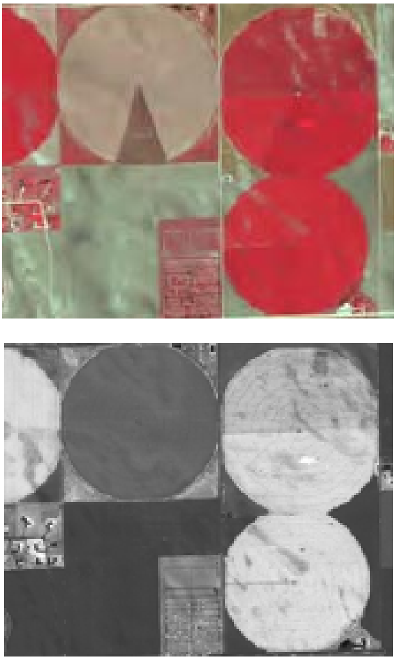
Output Format GeoTIFF

Additional georeference Google KML

Scripts by Jack™

Dr. Jack F. Paris, a remote-sensing and geospatial consultant, has developed a collection of advanced geospatial processing scripts, ***Scripts by Jack™***, that are in the public domain and can be used or modified in TNTgis by anyone. These scripts are distributed by MicroImages in the sample script collection noted previously or can be downloaded from the Scripting page at microimages.com.

Many of the ***Scripts by Jack*** focus on the calibration and interpretive processing of multispectral satellite images. The SRFI script processes images from a variety of satellite platforms to perform various radiometric corrections to a standardized reflectance factor index (SRFI). Images processed through SRFI can be used as input to several other scripts. The TERCOR script applies radiometric terrain correction to a SRFI image, and the GRUVI and TASCAP scripts compute measures of biophysical surface properties that can be used to interpret green vegetation biomass and soil conditions. Other scripts in the series provide color enhancement to the water areas of images of coastal areas (WATER) and subdivide grayscale images into contiguous areas with similar properties that are then outlined by polygons in a georeferenced vector object (OBJECT).



Top: color-infrared combination of QuickBird image bands of center-pivot irrigated fields. Bottom: optimized vegetation index image produced by the GRUVI script in the Scripts by Jack series. The GRUVI script adjusts for variations in background soil color and surface wetness to provide an accurate index of green vegetation biomass.

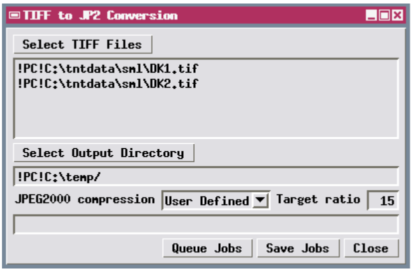
Each script in this series is documented in detail in the accompanying ***FAQs by Jack™*** series (available for download). These PDF documents provide the underlying concepts and theory behind the processing implemented in each script, an explanation of how the SML code is used to achieve the desired result, and step-by-step instructions on running the script.

Scripts Using Job Processing

Geospatial scripting can be combined with TNTmips Job Processing to efficiently perform specialized batch processing, allowing several identical script processes to be run simultaneously to take advantage of your computer's multiple processor cores.

To implement script-based batch job processing, the interactive user-input activities and the processing activities are performed in separate SML scripts. The interactive script provides an interface (such as a custom dialog window) to choose the input geospatial input objects, set global processing parameters, and designate the desired output destination. This script also uses the MIJOB class in SML to create a properly-formatted job file for each of the objects to be processed. This job file specifies the input object(s), provides variable values, and specifies the process script to be run. (Job files are automatically written to the TNT job directory you have set for your TNTmips installation.) You can specify in the TNT Job Manager window whether to immediately queue the jobs in the TNTmips Job Processing system or hold them to be queued later. You can also schedule jobs to be run at certain times (such as overnight or over the weekend).

When each job file is run, it calls the specified processing script and automatically passes the predefined variable values to it for execution. The process script must include variable declarations for all of the values to be passed from the job file. The process script then processes the input data specified by the current job file and creates the required output data. The illustration here shows the custom dialog window for a script that sets up batch conversion of TIFF image files to JPEG2000 files via job processing.

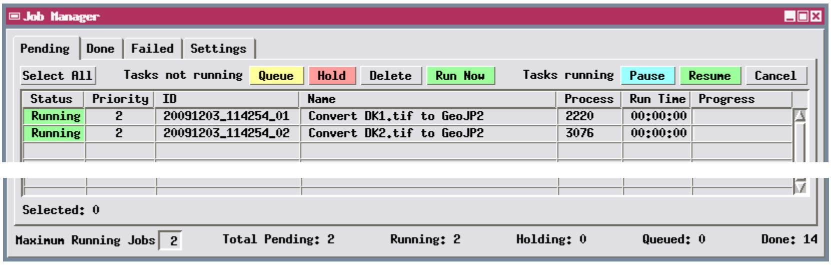


!PC!C:\tntdata\sml\DK1.tif
!PC!C:\tntdata\sml\DK2.tif

!PC!C:\temp/

JPEG2000 compression: User Defined — Target ratio: 15

Dialog window opened by an interactive script that sets up batch conversion of TIFF files to JPEG2000 via job processing.



Status	Priority	ID	Name	Process	Run Time	Progress
Running	2	20091203_114254_01	Convert DK1.tif to GeoJP2	2220	00:00:00	
Running	2	20091203_114254_02	Convert DK2.tif to GeoJP2	3076	00:00:00	

Selected: 0

Maximum Running Jobs 2 Total Pending: 2 Running: 2 Holding: 0 Queued: 0 Done: 14

TNT Job Manager running jobs utilizing SML process scripts.

Automated Processing with TNTscript

TNTscript is a professional product designed to enable automated production processing of geospatial data using SML scripts. TNTscript enables SML scripts to be executed on computers that do not have any other professional TNTgis products installed. If you develop SML processing scripts using TNTmips Pro or TNTedit Pro, TNTscript allows you to execute these scripts on additional computers, at remote sites, or using cloud-computing resources. TNTscript can also be used separately from the other TNTgis professional products to write, edit, and execute SML programs.

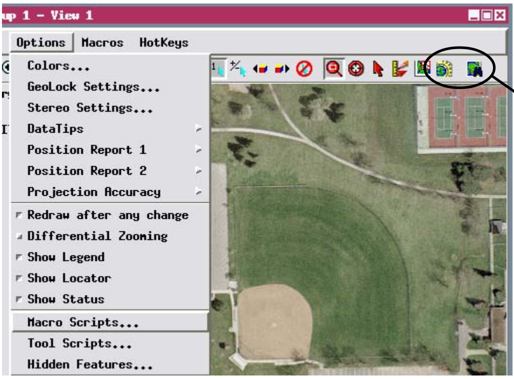
Runtime TNTscript is a Windows or Mac executable program that provides all of the noninteractive SML processing functionality found in TNTmips Pro. TNTscript executes SML scripts without user interaction during processing. It does not have a user interface, but instead is launched from the command line manually or by another program. Processing scripts written for use with TNTscript are therefore structured in a similar manner as those used in TNTmips job processing. All data-specific variables must be passed to the SML script by TNTscript at the start of execution. The script to be used and the required variable values can be specified on the command line or in an XML-formatted text file (.smlx).

TNTscript can be used to integrate custom SML processing into a production workflow involving other scripts and software products. A custom script controlling the work–flow can call TNTscript as needed and specify the name of an SML script and the required script parameters. The data produced by this processing script can then be passed along to the next step in the overall work–flow. Network licenses for TNTscript are available to allow TNTscript processing to be distributed over multiple networked computers. Licenses are also available for cloud-computing resources. TNTscript can therefore be used to integrate the capabilities of SML into geospatial processing workflows in an enterprise network to automatically create geospatial products as new data arrive, or in a web application that processes data by user request.

A TNTscript license includes the runtime TNTscript executable (with its own installer) and a professional license for TNTview (optional installation using the TNTgis installer). TNTview provides access to the SML Script Editor for writing, editing, and testing SML scripts for use with TNTscript. When you set the target product to TNTscript, the built-in Script Reference documentation indicates which functions and classes are available (everything except those related to interactive components such as pop-up dialogs, View windows, etc.). Fully interactive scripts can also be written and executed from TNTview under the TNTscript license.

Tool Scripts and Macro Scripts

Tool Scripts and Macro Scripts are specialized SML scripts launched from a View window. These script types are designed to allow easy access to and operations on the objects in the View. You can create tool scripts or macro scripts that enable any user to perform custom procedures on spatial data layers they are viewing. You can set up a general-purpose tool script or macro script to be available from any type of 2D View window or save a data-specific script with a particular group or layout. Scripts saved with a layout that becomes part of an atlas are also available for use in TNTatlas. Every View window offers menu selections that let you easily add and delete Tool scripts and Macro scripts (Options / Scripts).



Tool scripts and macro scripts can be launched from icon buttons on a View window's toolbar or from the Tool and Macros menus.

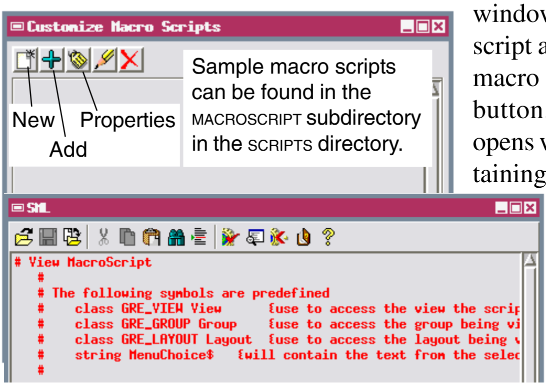
For the script writer, macro scripts and tool scripts provide a streamlined way to provide custom processing capabilities that require visual interaction with the spatial data. Because macro scripts and tool scripts are invoked from a View window, you don't need to provide code to create and manage the View window and its contents. Most of that management is taken care of automatically, so you can focus on coding the custom processing itself. Macro scripts and tool scripts:

- are executed from an icon button on a View window toolbar or from a menu;
- can access features of the current view such as layers, extents, projection, selected elements, zoom factor, scale, and styles;
- can operate on objects in the current view or objects containing the same area;
- can add a newly-created layer to the view;
- can start an external program and provide it with data derived from the current view.

A tool script can invoke a drawing tool and/or a dialog window (defined by the script-writer) that allow the user to interact with the spatial data in the View window. For example, the user could outline an area or select particular elements to be processed. A macro script does not allow such interaction but can be set up with a drop-down menu that provides program options.

Macro Script Setup

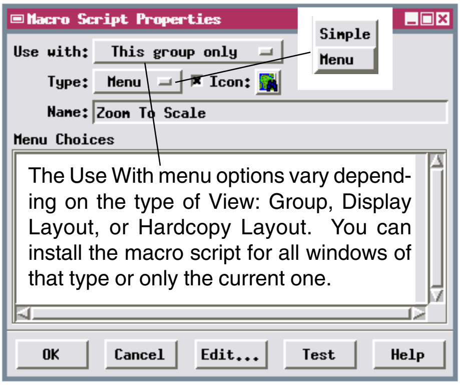
Macro scripts can be launched from the View window's Macros menu (which appears once you have installed a macro script) or from an optional icon button on the toolbar. To add a macro script, choose Options / Scripts / Macro Scripts from the View window, which opens the Customize Macro Scripts window. If you want to add an existing script, click on the Add icon button to open the Select File window so you can navigate to the script and select it. To create a new macro script, click on the New icon button. An SML editor window opens with a default stub script containing a list of predefined symbols that you can use in the macro script. The SML editor window includes all the script-creation and editing features of the standard SML window.



Sample macro scripts can be found in the MACROSCRIPT subdirectory in the SCRIPTS directory.

```
# View MacroScript
  #
  # The following symbols are predefined
  #    class GRE_VIEW View      {use to access the view the scrip
  #    class GRE_GROUP Group    {use to access the group being vi
  #    class GRE_LAYOUT Layout  {use to access the layout being v
  #    string MenuChoice$    {will contain the text from the selec
  #
```

Once you have created or added the macro script, the Macro Script Properties window opens. You can choose whether the script is accessible from all Views of the current type or only from the current saved group or layout. Choose Simple from the Type menu to have your tool script execute automatically without further input from the user. Choose Menu if you want drop-down choices presented from the Macros menu entry and the icon button; the Menu Choices text field then becomes active so you can enter the menu choices needed for the script. Use the Icon toggle button to indicate whether a script icon button appears on the View window toolbar; click on the default icon to open a dialog to select an appropriate icon. The text you enter in the Name field is used for the menu entry in the Macros menu and for the script icon button's ToolTip.



Use with: This group only
Type: Menu ☒ Icon:
Name: Zoom To Scale

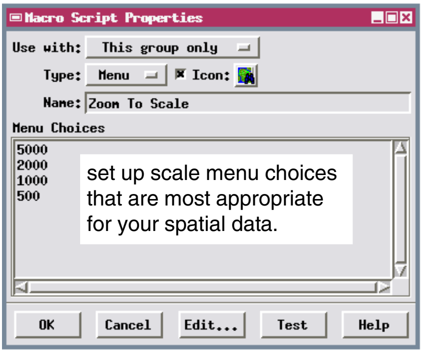
The Use With menu options vary depending on the type of View: Group, Display Layout, or Hardcopy Layout. You can install the macro script for all windows of that type or only the current one.

The Test button at the bottom of the window lets you run your script without closing the setup windows. Click OK in the Macro Script Properties and Customize Macro Scripts windows when you are done adding, developing, and/or testing your script.

Sample Macro Script: Zoom to Scale

The Zoom to Scale macro script (ZOOMTO.SML) is one of several sample macro scripts that are provided in the MACRSCR subdirectory of the SCRIPTS directory. This macro script lets the viewer redisplay the View window at one of several map scales selected from the script button's dropdown menu (or the script's submenu in the Macros menu cascade). For proper script function, the objects in the view window must be either georeferenced or scale-calibrated.

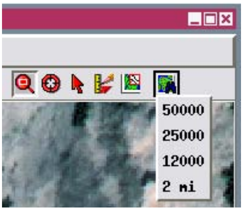
The scale menu selections are not predetermined by the Zoom to Scale script. When you install the script, you are free to set up the menu choices with the range of scale selections most appropriate for your data. The script accepts scale input from the menu as either map scale or ground dimensions. If the menu entry is purely numeric, it is interpreted as the denominator of the map scale fraction. For example, 12000 is interpreted as a map scale of 1:12000. If the menu entry is in two parts separated by a space (such as "1 mi"), the first part of the entry is interpreted as a ground dimension in miles. (This portion of the script can be easily modified to accept dimensions in kilometers or other distance units.) The script then performs the necessary calculations and sets the new map scale for the View window.



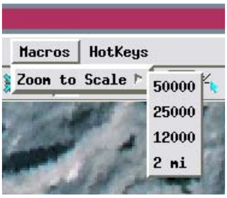
Use with: This group only
Type: Menu ☒ Icon:
Name: Zoom To Scale
Menu Choices
5000
2000
1000
500

set up scale menu choices that are most appropriate for your spatial data.

The predefined macro script variable `MenuChoice$` is used to represent the user's selection from the macro script menu button. If the menu choices are numeric, this string must be converted to a numeric value using the `StrToNum()` function.



A macro script can be launched and options selected using its icon button (left) or from the Macros menu (right).

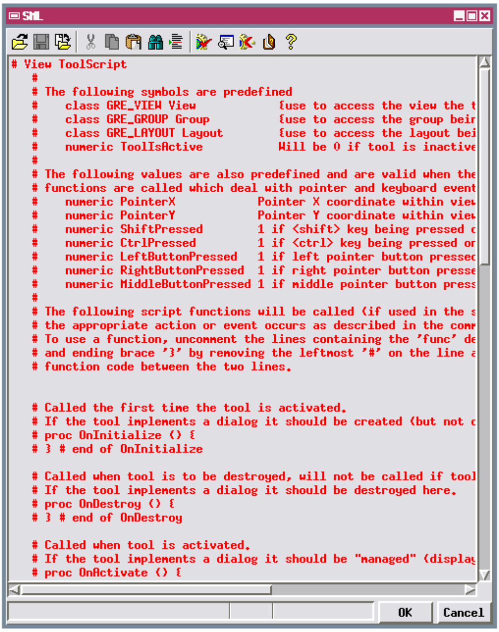


STEPS
- ☑ choose Main / Display from the TNTmips menu
- ☑ press the Open Display icon button on the Display Manager window
- ☑ navigate to the SML directory and select GROUPZOOMTO from the BUBBSCK Project File
- ☑ use the installed Zoom to Scale icon button on the View window to vary the zoom
- ☑ choose Options / Macro Scripts from the View window
- ☑ use the Properties and Edit icon buttons to examine the settings and script
- ☑ close the group

Tool Script Templates

A number of class and numeric variables are predefined and available for immediate use in tool scripts.

Tool scripts can be run from an icon button on the View window toolbar or from the Tools menu. To add a tool script, choose Options / Tool Scripts from the View window in any TNTgis process that has a View window. Making this selection opens the Customize Tool Scripts window, which is nearly identical to the Customize Macro Scripts window discussed previously.

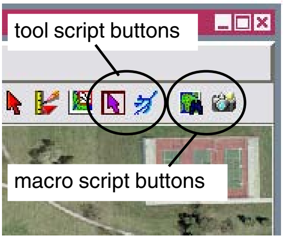


```
# View ToolScript
   #
   # The following symbols are predefined
   #    class GRE_VIEW View            {use to access the view the t
   #    class GRE_GROUP Group          {use to access the group beir
   #    class GRE_LAYOUT Layout        {use to access the layout bei
   #    numeric ToolIsActive           Will be 0 if tool is inactive
   #
   # The following values are also predefined and are valid when the
   # functions are called which deal with pointer and keyboard event
   #    numeric PointerX            Pointer X coordinate within view
   #    numeric PointerY            Pointer Y coordinate within view
   #    numeric ShiftPressed        1 if <shift> key being pressed c
   #    numeric CtrlPressed         1 if <ctrl> key being pressed or
   #    numeric LeftButtonPressed   1 if left pointer button pressec
   #    numeric RightButtonPressed  1 if right pointer button presse
   #    numeric MiddleButtonPressed 1 if middle pointer button press
   #
   # The following script functions will be called (if used in the s
   # the appropriate action or event occurs as described in the comr
   # To use a function, uncomment the lines containing the 'func' de
   # and ending brace '}' by removing the leftmost '#' on the line a
   # function code between the two lines.


   # Called the first time the tool is activated.
   # If the tool implements a dialog it should be created (but not c
   # proc OnInitialize () {
   # } # end of OnInitialize

   # Called when tool is to be destroyed, will not be called if tool
   # If the tool implements a dialog it should be destroyed here.
   # proc OnDestroy () {
   # } # end of OnDestroy

   # Called when tool is activated.
   # If the tool implements a dialog it should be "managed" (display
   # proc OnActivate () {
```

To create a new tool script, click on the New icon button to open an SML editor window, which shows the tool script template. The template lists a number of predefined symbols and values that you can use in any tool script. The predefined values include the X and Y coordinates of the screen cursor within the view (in pixels) and values that record mouse button actions.

Additionally, the tool script template includes skeletal definitions of procedures likely to be used in a tool script. These include procedures used the first time a tool is activated; when the tool is destroyed; when the tool is activated and deactivated; when the tool is suspended (during redraw) and resumed (after redraw); when the left, right, or middle mouse button is pressed or released; when the cursor moves without a button press; when the cursor moves with a button press; when the cursor enters or leaves the View window; and when the user presses a key. To create your script, remove the comment characters (#) to the left of each procedure definition you need and add code to specify the desired action to be carried out by that procedure.

Tool Script icon buttons appear to the left of any Macro Script icon buttons on the View window toolbar.



Sample Tool Script: Flow Path

The Flow Path tool script shows how custom analysis procedures can be performed on layers in the current view using an SML Tool Script. The script uses SML watershed functions that operate on an elevation raster (DEM) that must be the first layer in the View window.

When you launch the script, it first executes watershed functions to create a depressionless version of the DEM and a complete set of vector flow paths. These derived features are required by subsequent script steps; they are stored as temporary objects and are not displayed in the View. The script then opens a FlowPath and Buffer Zone window and creates a graphic tool that allows the user to place one or more watershed seed points on the DEM or on an overlying image layer. Toggle buttons on the window enable the user to choose to compute and display:

- the upstream basin (area with flow toward the seed point),
- the flow path downstream, and
- a buffer zone around the flow path.

If you prefer seed points to fall along a stream course, you can turn on the Move Seed Point to Flow Path toggle button. Each seed point is then moved to the nearest precomputed flow path line before the new flow path and basin are computed. You can place new seed points, repeat the analysis as many times as desired, and save the computed vector objects.

STEPS

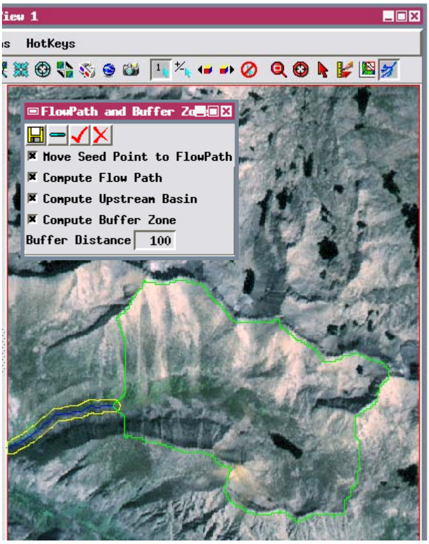
☑ press the Open Display icon button on the Display Manager window

☑ navigate to the SML directory and select GROUP FLOWPATH from the BUBBSCK Project File

☑ press the Flowpath icon button on the View window to launch the tool script

☑ left-click at a point along a stream to create a downstream flowpath line and upstream basin polygon (the basin polygon may be small unless the seed point is exactly on a DEM cell along a stream channel)

☑ close the group



FlowPath and Buffer Zo

☒ Move Seed Point to FlowPath

☒ Compute Flow Path

☒ Compute Upstream Basin

☒ Compute Buffer Zone

Buffer Distance 100

The script also creates and displays (in red) a vector layer outlining the extents of the DEM. If an overlying image layer is larger than the DEM, the user can use the extents box to guide placement of the seed points. The extents box is also used to automatically clip buffer zones computed from flow paths that intersect the DEM boundary.

Script Layer in Display

STEPS

☑ press the Open Display icon button on the Display Manager window

☑ navigate to the SML directory and select PAGE LAYOUT 1 from the BUBBSCK Project File

☑ expand the Group 2 entry in the Display Manager

☑ press the Layer Controls icon button for the ARROW layer in Group 2

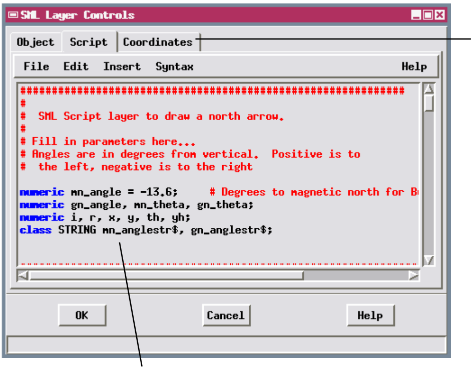
☑ examine the script in the Script tabbed panel in the SML Layer Controls window

☑ press [OK] on the SML Layer Controls window

☑ close the group

The standard display process (Display / Spatial Data) supports the use of an SML script as a layer, just as any spatial object can be a layer. An SML script layer can use flexible cartographic drawing functions to create special map symbols and neatlines. You can add a script as a layer by choosing Add/Layer/Script from the Display Manager.

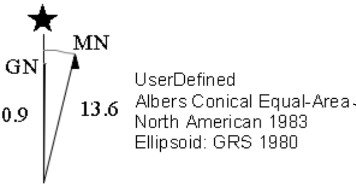
The sample script ARROW.SML is designed to draw an oriented magnetic declination map symbol in a layout, as in the example shown here. For this script the SML layer should be alone in a group. The script determines the angle between grid north and true north from the previous map group in the layout; the magnetic declination value in the script must be edited to suit the location of the map data. The declination symbol includes true north, grid north, and magnetic north lines, labels for the angles, and text describing the coordinate reference system used.

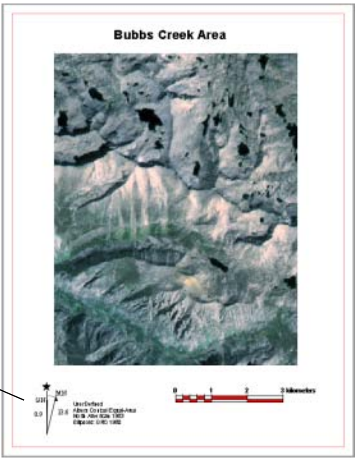


```
################################################################
#
#  SML Script layer to draw a north arrow.
#
# Fill in parameters here...
# Angles are in degrees from vertical.  Positive is to
#  the left, negative is to the right

numeric mn_angle = -13.6;      # Degrees to magnetic north for B
numeric gn_angle, mn_theta, gn_theta;
numeric i, r, x, y, th, yh;
class STRING mn_anglestr$, gn_anglestr$;
```

You can also add script layers to groups containing other spatial layers. The Coordinates panel lets you relate the script layer to the map coordinates of the other layers in the group.

The Script tabbed panel in the SML Layer Controls window contains the interface for editing and running scripts.



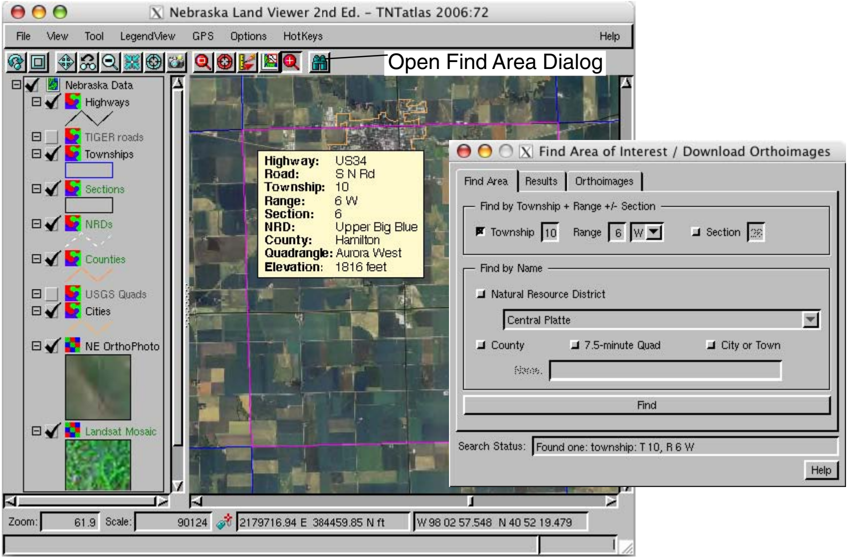


Interactive Display Control Scripts

In addition to their use to create enhanced DataTips that can include graphics, Display Controls scripts can be used to open a custom dialog window that provides interactive access to data layers in the View. One such control script application is to provide custom spatial or attribute search capability for a layout or atlas.

The Nebraska Land Viewer Atlas 2nd Edition created by MicroImages includes a Find Area of Interest dialog that is created and automatically opened by a display control script, NESearchCtrl.sml. The dialog and accompanying code in the control script enables viewers of the atlas to search by name for a township, section, Natural Resource District, map quadrangle, or city/town. Inexact matches are allowed by the search code, so multiple results are returned when needed to a results panel. When a final single result is returned or selected by the viewer, the atlas View automatically zooms to the spatial extent of the selected area. The search dialog also can provide a list of available orthoimages for the USGS quadrangle at the center of the view, any of which can be selected directly for download from a Nebraska state government web site.

Since this search utility is provided by a control script, it remains active if the viewer changes interactive tools or uses the tool script created for the atlas, which provides scale-variable zoom capabilities. The display control script also installs an icon button on the View to reopen the dialog if the viewer has closed it.



Open Find Area Dialog

Highway: US34
Road: S N Rd
Township: 10
Range: 6 W
Section: 6
NRD: Upper Big Blue
County: Hamilton
Quadrangle: Aurora West
Elevation: 1816 feet

☒ Township 10 Range 6 W ☐ Section 28

☐ Natural Resource District: Central Platte

☐ County ☐ 7.5-minute Quad ☐ City or Town

Name:

Search Status: Found one: township: T 10, R 6 W

Zoom: 61.9 Scale: 90124 2179716.94 E 384459.85 N ft W 98 02 57.548 N 40 52 19.479

Advanced Software for Geospatial Analysis

MicroImages, Inc. publishes a complete line of professional software for advanced geospatial data visualization, analysis, and publishing. Contact us or visit our web site for detailed product information.

TNTmips Pro TNTmips Pro is a professional system for fully integrated GIS, image analysis, CAD, TIN, desktop cartography, and geospatial database management.

TNTmips Basic TNTmips Basic is a low-cost version of TNTmips for small projects.

TNTmips Free TNTmips Free is a free version of TNTmips for students and professionals with small projects. You can download TNTlite from MicroImages' web site.

TNTedit TNTedit provides interactive tools to create, georeference, and edit vector, image, CAD, TIN, and relational database project materials in a wide variety of formats.

TNTscript TNTscript lets you execute SML scripts on additional computers, at remote sites, or using cloud-computing resources.

TNTview TNTview has the same powerful display features as TNTmips and is perfect for those who do not need the technical processing and preparation features of TNTmips.

TNTatlas TNTatlas lets you publish and distribute your spatial project materials on CD or DVD at low cost. TNTatlas CDs/DVDs can be used on any popular computing platform.

Index

MicroImages, Inc.

www.microimages.com