

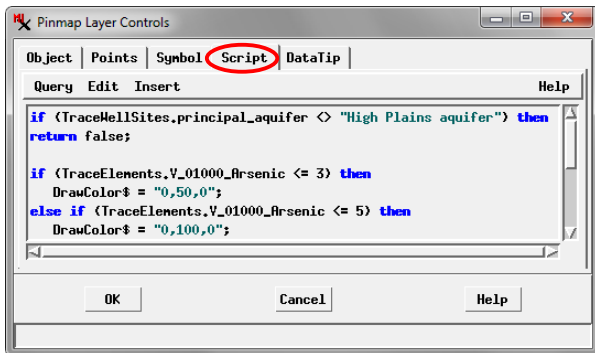
# Scripts for Pin Map Layers

A *pinmap* layer in TNTgis displays point location information directly from a database table (or set of related tables) that include coordinate values (see the Technical Guide entitled *Database Pin Mapping*). A pinmap layer can utilize a script to create custom settings for a number of aspects of the pinmap display, including:

- selection of records by attribute
- X and Y coordinates
- assignment of pin symbol, color, or size by attribute
- custom pie chart or bar graph symbols
- label text, position, and orientation

## Script Tabbed Panel

A pinmap script is entered on the Script tabbed panel of the Pinmap Layer Controls window. Most code sections entered in the text pane on this panel are applied only if the corresponding By Script option is selected in the controls on the other tabbed panels, as described below.



**Insert Menu:** Scripts for pinmaps can reference fields in the selected table (or tables related to it) and can use a number of predefined variables and functions for setting point coordinates, symbols, and label text and orientation. The Insert menu on the Script panel allows you to insert items into the script at the cursor location.

The Field option on the Insert menu allows you to insert a reference to a field; it opens the Insert Field window, which provides a Table menu listing available tables and a Field menu listing the fields for the selected table. The other options on the Insert menu (Symbol, Function, Operator, Keyword, and Class) all open the Script Reference window, from which you can select any of these categories and insert the desired item. Although you can manually type predefined variable and function names and field references, using the Insert option helps reduce typographical errors.

**Edit Menu:** The Edit menu provides options to Copy or Cut selected text, to Paste text from the clipboard, and to Clear selected text. The Insert File option prompts you to select a text file containing text to insert into the script. The Find option prompts you for a text string to search for; pressing OK on the Find window finds the first instance of the text (if any).

The Find Again menu option repeats the previous search from the current cursor location. The Go to Line option prompts for a line number in the script and repositions the cursor at that line. The Character Map option opens the Character Map window, which provides a list of special characters; double-click on an entry in the window to insert the character into the script at the cursor location.

**Query Menu:** The Query menu provides a New option to start a new script (which clears the script pane), Save and Save As options, and the option to Open a saved script. The Syntax option on the Query menu checks the syntax of the script and reports any errors. The List option opens a window displaying pseudocode for the current script.

## Records by Script

If the Records menu on the Object tabbed panel is set to By Script, the pinmap script can be used to select records for pin display based on the values in specified fields in the table(s). Standard syntax for selection queries can be used (see the *Building and Using Queries* tutorial for many examples). The compound selection query example below selects water sample well records for three different aquifer systems:

```

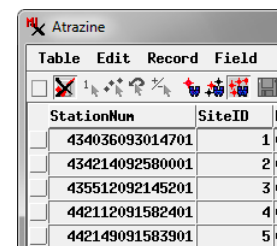
if (TraceWellSites.principal_aquifer == "High Plains aquifer" or
TraceWellSites.principal_aquifer == "Rio Grande aquifer system" or
TraceWellSites.principal_aquifer == "Rocky Mountain Front Range")
return true;
else
return false;
    
```

For selection queries it is good practice to have the script return "true" or "false" for records meeting the comparison criteria and then return the opposite value for other cases.

## Point Coordinates by Script

If the X, Y, and/or Z menus on the Points tabbed panel are set to By Script, code on the Script tabbed panel can be used to compute the coordinates from field values that are not in a directly useable form. The pinmap script context provides the following predefined numeric variables (in the Variables group in the Script Reference window) for setting coordinate values: XCoord, YCoord, and ZCoord. The calculated coordinates must conform to the coordinate reference system specified using the Projection button and field on the Points tabbed panel. For latitude and longitude coordinates, the computed values must be in decimal degrees.

As an example, the table illustrated to the right has records in which the numeric value in the StationNum field encodes both the latitude and longitude coordinates for the site. The first six digits provide the latitude in degrees, minutes, and seconds (two digits for each). The next seven dig-



StationNum	SiteID
434036093014701	1
434214092580001	2
435512092145201	3
442112091582401	4
442149091583901	5

(continued)

its provide the west longitude coordinate with 3 digits for degrees, and 2 digits each for minutes and seconds. (The trailing two digits specify a sample number at the location; in this example there is only one sample per station).

The script shown to the right converts the number in the StationNum field to a string so the separate parts can be parsed using methods in the STRING class. Numeric variables corresponding to degrees, minutes, and seconds are declared, and the string portion for each is extracted and converted to a number, first for the latitude, then for the longitude. For each coordinate the three variable values are used to compute the value in decimal degrees, which is assigned to the corresponding predefined variable. (The value for XCoord must be multiplied by -1 to correspond to west longitude.)

### Symbol and/or Color by Script

If the Symbol menu on the Symbol tabbed panel is set to By Script, you can use the pinmap script to assign different pin styling based on attributes in the table. You can assign point styles or point symbols from the designated style object, set the color used for the “variable” color in a point symbol, and vary the scale (size) of symbols and their offset from the pin coordinates. A number of predefined variables (listed to the right) are available for use in the script for these purposes.

The style portion of a pinmap script allows you to modify or override selections made using the Symbol settings on the Symbol tabbed panel. For example, if you have selected a graphic symbol with a “variable” color for the default symbol (one of the built-in predefined point symbols or a custom point symbol from the style object with variable color), you can set the color to use for the variable color using the DrawColor\$ variable. This variable lets you set the color by name (using colors defined in the rgb.txt reference file in your TNTgis installation directory) or using a set of red, green, and blue percentage values. The script shown in the example to the right assigns a color to the variable symbol color in the default symbol based on value ranges of a numeric variable. Note that the selection query at the beginning of the script first excludes all records not matching the desired attribute value; the style portion of the script then only has to examine the desired records to make the color assignments. See the *Database Pin Mapping TechGuide* for a description of the data used in this example.

You can use the Symbol\$ variable to select a graphic point symbol that is stored in the style object you have chosen for use with the pinmap. The names of the available symbols are shown in the Style Editor window (opened from the Style pushbutton on the Symbol tabbed panel) when you choose the Symbol option (see the TechGuide entitled *Creating Styles for Points* for information on the Style Editor). The sample script at the top of the next page uses the Symbol\$ variable to assign three different named symbols to pins based on attribute values. It also assigns the drawing color to use for the variable color for each symbol, and uses the XScale and YScale numeric variables to set scaling factors that determine the size of the symbols in millimeters at the designated map scale (from the default style or set by the MapScale script variable).

```
class STRING coord$ = sprintf("%0.f", Atrazine.StationNum);
numeric degr, min, secnd;
```

```
degr = StrToNum( coord$.substr(0, 2) ); # first 2 characters
min = StrToNum( coord$.substr(2, 2) ); # next 2 characters
secnd = StrToNum( coord$.substr(4, 2) ); # next 2 characters
```

```
min += secnd / 60;
YCoord = degr + (min / 60);
```

```
degr = StrToNum( coord$. substr(6, 3) );
min = StrToNum( coord$.substr(9, 2) );
secnd = StrToNum( coord$.substr(11, 2) );
```

```
min += secnd / 60;
XCoord = -1 * (degr + (min / 60) );
```

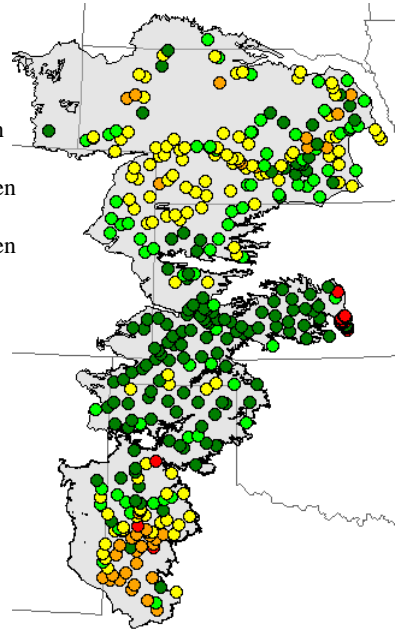
#### Predefined Variables for Pinmap Symbols

Symbol\$	Name of point symbol to use
DrawSymbol	Flag: 1 = yes, 0 = no
DrawColor\$	Color to use for the “variable” color in a symbol (RGB percent e.g. “50,100,50” or color name)
DrawPatt\$	Name of bitmap pattern
DrawBitmapPatt	Flag: 1 = yes, 0 = no
SymbolOffsetX	Offset of symbol from point coordinates
SymbolOffsetY	
Style\$	Name of a point style to use
UseStyle	Flag: 1 = yes, 0 = no
MapScale	Map scale number
XScale	Scale factor in X direction
YScale	Scale factor in Y direction

```
if (TraceWellSites.principal_aquifer <> "High Plains aquifer") then
return false;

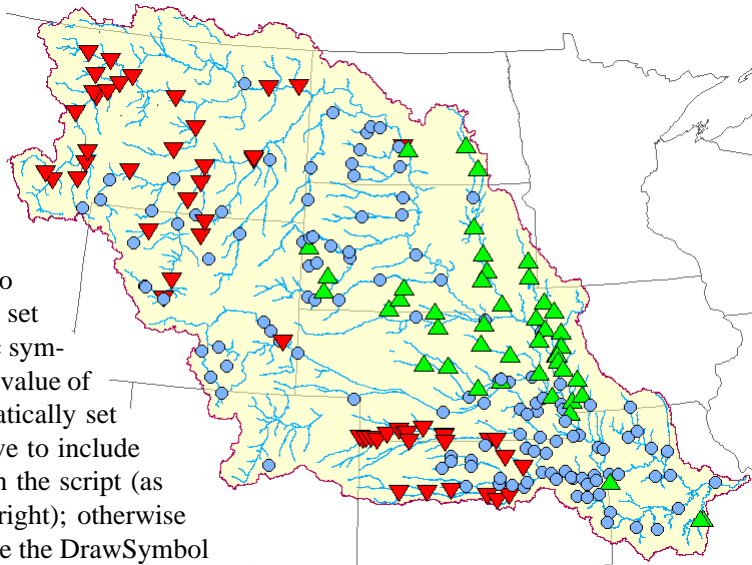
if (TraceElements.V_01000_Arsenic <= 3) then
DrawColor$ = "0,50,0";
else if (TraceElements.V_01000_Arsenic <= 5) then
DrawColor$ = "0,100,0";
else if (TraceElements.V_01000_Arsenic <= 10) then
DrawColor$ = "yellow";
else if (TraceElements.V_01000_Arsenic <= 50) then
DrawColor$ = "orange";
else
DrawColor$ = "red";

return true;
```



(continued)

For graphic symbols there is an associated numeric flag variable `DrawSymbol` that determines whether the symbol or pattern designated in the script is actually drawn (set the variable to 1 to draw). If the pins are set to use a custom graphic symbol by default, then the value of `DrawSymbol` is automatically set to 1, and you don't have to include the value assignment in the script (as in the example to the right); otherwise the script should include the `DrawSymbol` statement.



```

if (StreamTrends.Trend == "Up")
{
Symbol$ = "TriangleUp";
DrawColor$ = "green";
XScale = 5;
YScale = 5;
}
else if (StreamTrends.Trend == "Down")
{
Symbol$ = "TriangleDown";
DrawColor$ = "red";
XScale = 5;
YScale = 5;
}
else
{
Symbol$ = "circle2blsolid";
DrawColor$ = "50,70,100";
}
MapScale = 10000000;

```

You can also choose bitmap patterns in the style object for the pin markers using the `DrawPatt$` variable, and use the associated `DrawBitmapPatt` flag variable if the pins are not set to use a bitmap pattern by default.

Symbol Functions for Use with Symbol\$	
<code>BarGraph\$ (max, bordercolor\$, value1, color1\$, value2, color2\$, ...)</code>	draw bar graph
<code>Circle\$ (x, y, radius, color, filled)</code>	draw circle
<code>Ellipse\$ (x, y, r1, r2, color, filled, angle)</code>	draw ellipse
<code>Line\$ (x1, y1, x2, y2, color, thickness)</code>	draw line
<code>PieChart\$ (bordercolor\$, value 1, color1\$, value2, color2\$, ...)</code>	draw pie chart
<code>Rectangle\$ (x, y, height, width, angle, color, filled)</code>	draw rectangle

## Custom Symbols: Pie Charts and Bar Graphs

A number of drawing functions (in the Symbol function group) are available to enable a pinmap script to draw custom pin symbols. Functions are provided to draw lines, rectangles, circles, and ellipses (see box above). The selected function is assigned to the `Symbol$` variable in place of a symbol name. A pinmap script can therefore draw custom symbols composed of these geometric elements. More interesting applications are provided by the `PieChart$` and `BarGraph$` functions, which render custom symbols that show information from multiple fields in the table.

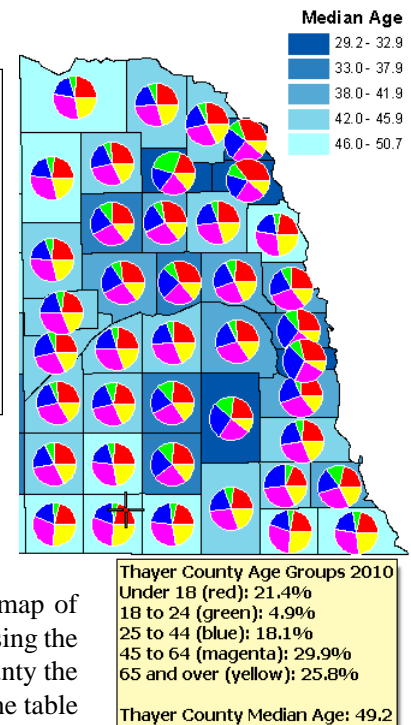
```

Symbol$ = PieChart$("white",
NE_AgeGroups.UNDER18, "red",
NE_AgeGroups.`18TO24`, "green",
NE_AgeGroups.`25TO44`, "blue",
NE_AgeGroups.`45TO64`, "magenta",
NE_AgeGroups.`65ANDOVER`, "yellow");

DrawSymbol = 1;
MapScale = 3000000;
XScale = 10;
YScale = 10;

```

**Pie Charts:** Using a pie chart symbol for pins lets you show the differing proportions of the components that make up a whole. The example shown to the right uses a breakdown of county population by age group in a polygon database table that also includes fields with the coordinates of the polygon centroids. The vector polygons are displayed using a theme map of median age (see legend in illustration), while the polygon table is used as a pinmap overlay using the polygon centroid coordinates for the pin locations. The pie chart symbols show for each county the proportion of population in each of five age groups. A computed string field was set up in the table to provide a multiline `DataTip` for the pins to show the age classes, color, and their percentages.



In the `PieChart$` function, the first parameter specifies the border color for the pie and pie wedges. Using a contrasting color (white in this example) makes it easier to distinguish the wedge boundaries. Next are any number of pairs of parameters, with one pair for each wedge in the chart; the first parameter in the pair specifies the wedge value (provided by different fields in the table) and the second parameter specifies the fill color for the wedge. The color specifications can be omitted, in which case a default progression of six colors (red, green, blue, cyan, magenta, and yellow) is used automatically in that order for the wedge fill colors.

**Bar Graphs:** Using a bar graph symbol for pins lets you depict varying relative values of a number of independent numerical attributes. In the example on the following page the bars in the bar graph symbols indicate relative abundances of five chemical

(continued)



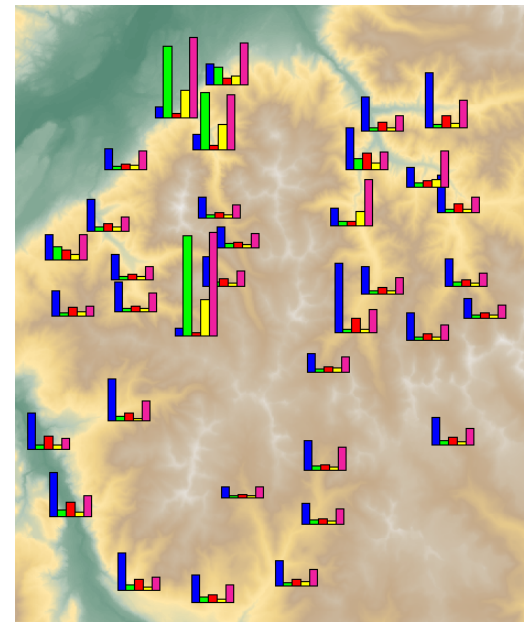
elements (chromium, copper, nickel, lead, and zinc) measured in stream sediment samples.

In the BarGraph\$ function the first parameter (*max*) sets the maximum data value that is used to control the vertical scaling for the bars. Setting the value of the *max* parameter equal to the actual maximum data value in the selected fields should produce bars of reasonable height relative to the width of the bar graph base line (assuming equal values for XScale and YScale). Bar heights vary inversely with the value of the *max* parameter. Setting *max* to be greater than the actual maximum value produces shorter bars, while setting it to be less than the actual maximum increases the bar heights. The second parameter in the function sets the border color for the bars. This is followed by pairs of parameters, with value (field reference) and color for each of the bars to be drawn, starting with the one on the left side of the graph. Omitting the color parameters results in default color assignments as in the PieChart\$ function.

```
Symbol$ = BarGraph$(200, "black",
  geochem_q46112.CR_ICP40, "blue",
  geochem_q46112.CU_ICP40, "green",
  geochem_q46112.NI_ICP40, "red",
  geochem_q46112.PB_ICP40, "yellow",
  geochem_q46112.ZN_ICP40, "purple");

DrawSymbol = 1;

MapScale = 200000;
XScale = 10;
YScale = 10;
```



### Label Text and Position by Script

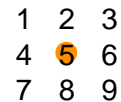
If the Label option on the Symbol tabbed panel is set to By Script, you can use the pinmap script to designate the text to use for pin labels using the Label\$ variable. You can use the script to create label text by concatenating the contents of several fields or assign different label text depending on the value of a designated field.

Whether or not the label text is generated by script, you can use the pinmap script to customize the positioning of labels relative to their pin symbols. Label position is set in the script by assigning a value between 1 and 9 to the LabelPosn variable (see illustration to the right). For example, position 1 corresponds to the Upper Left choice on the Label Position menu and position 9 corresponds to the Lower Right option. You can also change the angle of the label baseline relative to horizontal by assigning an angle value (degrees counterclockwise) to the LabelAngle variable.

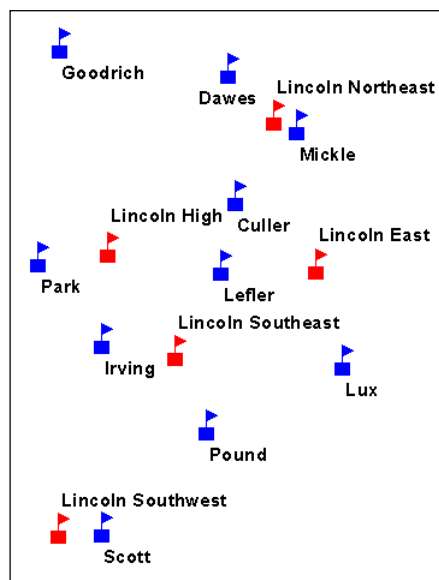
For labels that are not centered on the pin location, you can move the label farther away from the pin by assigning a positive number to the LabelOffset variable. This moves the label outward in the direction indicated by the label position. For example, a label assigned to position 9 (lower right) would be moved further outward in that diagonal direction. For labels at the corner positions you can use the LabelOffsetX and LabelOffsetY variables to assign different offsets in the X (horizontal) and Y (vertical) directions on the screen. Labels positioned directly above or below the pin (positions 2 and 8, respectively) can only be moved in the Y direction (LabelOffsetX has no effect); for those positioned to the left or right (positions 4 or 6), LabelOffsetY has no effect.

The pinmap and script to the right assigns different label positions to high schools (red symbols) and middle schools (blue symbols). Label positioning by script is provided primarily to prevent overlap between a label and its symbol, but here it is also used to avoid overlap with other symbols and labels (see the special positioning set for the label for Dawes Middle School).

Predefined Variables for Pin Label Position	
Label\$	Text string for label
LabelPosn	Label position number
LabelAngle	Label angle (degrees counter-clockwise)
LabelOffset	Offset from symbol in direction of position
LabelOffsetX	Offset in the X direction
LabelOffsetY	Offset in the Y direction



Label positions relative to the pin location (orange circle) set by values of the LabelPosn variable.



```
if (PUBLIC.ED_LEVEL == "High")
{
  DrawColor$ = "red";
  LabelPosn = 3;
  LabelOffsetX = 0;
  LabelOffsetY = 12;
}
else if (PUBLIC.ED_LEVEL == "Middle")
{
  DrawColor$ = "blue";

  if (PUBLIC.SCHOOL <> "Dawes")
  {
    LabelPosn = 9;
    LabelOffsetX = 0;
    LabelOffsetY = 5;
  }
  else
  {
    LabelPosn = 8;
    LabelOffset = 5;
  }
}
}
```