

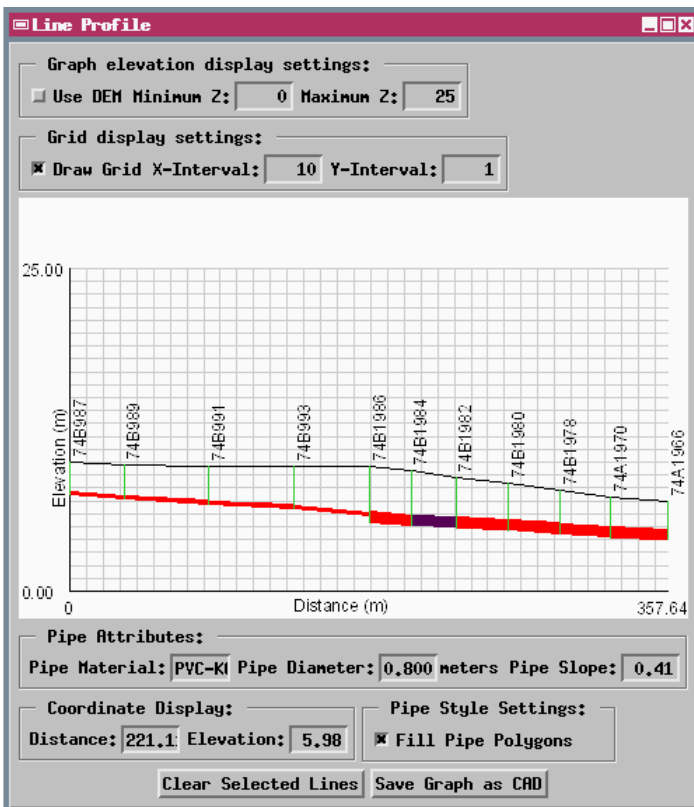
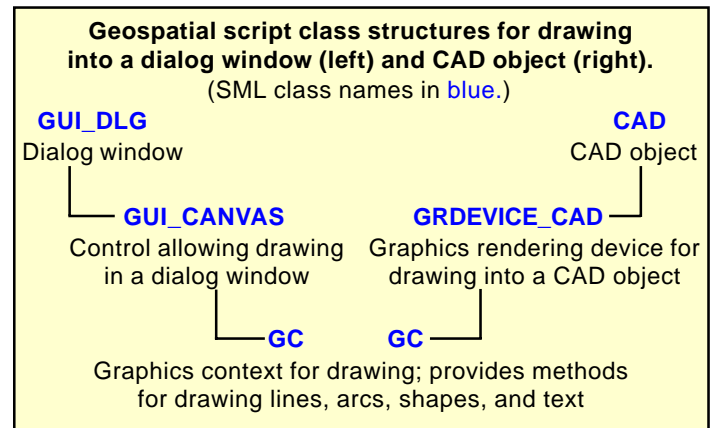
## Sample Tool Script

# Draw into CAD Object in a Script

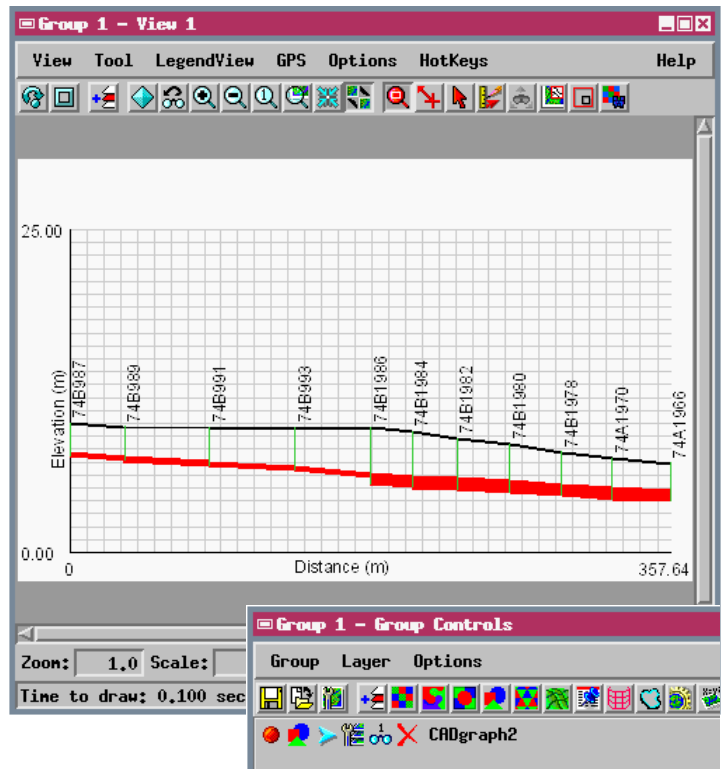
A CAD object in TNTmips can contain either geospatial data or nongeoreferenced graphics related to your geospatial data, such as data plots, profiles, a rendered database table, or map-margin graphics. The TNT geospatial scripting language (SML) can be used to draw graphical elements (lines, arcs, geometric shapes, and text) into a CAD object that can then be used in map layouts or exported to a CAD file format for use with other software products.

The PipeProfileCAD tool script shows that the same powerful drawing methods can be used to draw graphics into a dialog window created by the script and to then replicate and save the same plot as a CAD object when needed. The primary purpose of this script is to present an on-screen vertical profile of a selected portion of an underground pipe network. (The color plate entitled *Sample Tool Script: Infrastructure Graphical Profile* expands upon the design of this script for drawing the plot to an on-screen window.) The script can then save this on-screen profile to a CAD object when the Save Graph as CAD button on the window is

pressed. Generic methods for drawing graphic elements are provided by a GC (graphics context) class that can be used in conjunction with either a dialog window or a CAD object. The script class hierarchies for these two applications are illustrated in the diagram below, and their use is compared in the drawing procedure code excerpts included on the opposite side of this page.



Line Profile window created by the PipeProfileCAD tool script, showing a vertical profile of a selected section of a subsurface pipe network. The upper black line connects surface manholes; the subsurface pipes are drawn in red. The graph is drawn into the window by the script using database values for a contiguous set of line elements selected interactively in the View. The currently active line element is highlighted in purple.



Display of a CAD object that replicates the pipe profile graphic from the Line Profile window. Pushing the Save Graph As CAD button on the Line Profile window (left) redraws the profile to a selected new CAD object. In this example the same drawing attributes, colors, and font were used in the tool script to create both the window graphic and the CAD object, and the results are virtually identical.

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/freestuf/scripts.htm](http://www.microimages.com/freestuf/scripts.htm).

## Script Excerpts for Pipe Profile Script (PipeProfileCAD.sml)

Procedure used to draw the graph in the dialog window; uses global GC to allow interactive element highlighting

```
proc drawGraph() {
  pipeBottomSave = pipeBottom;
  pipeTopSave = pipeTop;
  pipeFaceSave = pipeFace;

  createGC();
  gc.DrawTextSetFont("ARIAL");
  local class COLOR color;

  setTrans(pipeBottom);

  local string xlabel = "Distance (m)";
  local string ylabel = "Elevation (m)";
  local numeric drawTwoPointLines = 0;
  local numeric drawStartEndPoints = 1;

  local numeric fontHeight = 12, axisLabelOffset = 3;
  setGraphOffsets(gc, fontHeight, axisLabelOffset);

  local class COLOR bgcolor;
  bgcolor.red = 98; bgcolor.green = 98; bgcolor.blue = 98;
  drawBackground(gc, bgcolor);

  color.red = 80; color.green = 80; color.blue = 80;
  drawGrid(gc, getGridIntervalX(), getGridIntervalY(), pipeBottom, color);

  color.red = 0; color.green = 0; color.blue = 0;
  drawGraphAxes(gc, pipeBottom.GetVertex(pipeBottom.GetNumPoints()-1).x,
    xlabel, ylabel, drawTwoPointLines, color, fontHeight, axisLabelOffset);

  local class COLOR fill = vectorLayer.SelectedElemColor;
  drawRectangles(gc, pipeFace, color, fill, fillToggle.GetValue());

  class POLYLINE manholeSurfaceLine;
  if (doUseDEM()) {
    gc.DrawSetLineStyle("");
    color.red = 0; color.green = 0; color.blue = 0;
    drawPolyline(gc, smoothedSurface, color);
    manholeSurfaceLine = demSurface;
  }
  else {
    color.red = 0; color.green = 0; color.blue = 0;
    drawPolyline(gc, surface, color);
    manholeSurfaceLine = surface;
  }

  color.red = 20; color.green = 80; color.blue = 20;
  drawManholes(gc, manholeDepth, manholeSurfaceLine, color);

  color.red = 0; color.green = 0; color.blue = 0;
  drawManholeNames(gc, manholeSurfaceLine, manholeNames, color,
    "ARIAL", 12);

  canvas.Refresh(1);
}
```

Procedure for drawing graph in CAD object; passed a local GC when called by the OnSaveGraph() procedure

```
proc drawGraphForCAD(class GC gc) {
  local class COLOR color;
  gc.DrawTextSetFont("ARIAL");

  setTrans(pipeBottom);
}
```

```
local string xlabel = "Distance (m)";
local string ylabel = "Elevation (m)";
local numeric drawTwoPointLines = 0;
local numeric drawStartEndPoints = 1;

local numeric fontHeight = 12, axisLabelOffset=3;
setGraphOffsets(gc, fontHeight, axisLabelOffset);

local class COLOR bgcolor;
bgcolor.red = 98; bgcolor.green = 98; bgcolor.blue = 98;
drawBackground(gc, bgcolor);

color.red = 80; color.green = 80; color.blue = 80;
drawGrid(gc, getGridIntervalX(), getGridIntervalY(), pipeBottom, color);

color.red = 0; color.green = 0; color.blue = 0;
drawGraphAxes(gc, pipeBottom.GetVertex(pipeBottom.GetNumPoints()-1).x,
  xlabel, ylabel, drawTwoPointLines, color, fontHeight, axisLabelOffset);

local class COLOR fill = vectorLayer.SelectedElemColor;
drawRectangles(gc, pipeFace, color, fill, fillToggle.GetValue());

class POLYLINE manholeSurfaceLine;
if (doUseDEM()) {
  gc.DrawSetLineStyle("");
  color.red = 0; color.green = 0; color.blue = 0;
  drawPolyline(gc, smoothedSurface, color);
  manholeSurfaceLine = demSurface;
}
else {
  color.red = 0; color.green = 0; color.blue = 0;
  drawPolyline(gc, surface, color);
  manholeSurfaceLine = surface;
}

color.red = 20; color.green = 80; color.blue = 20;
drawManholes(gc, manholeDepth, manholeSurfaceLine, color);

color.red = 0; color.green = 0; color.blue = 0;
drawManholeNames(gc, manholeSurfaceLine, manholeNames, color,
  "ARIAL", 12);
}
```

Procedure called when Save Graph as CAD button on dialog is pressed

```
proc OnSaveGraph() {
  GetOutputCAD(CADgraph);
  deviceCAD.Create(CADgraph, getHeight(), getWidth());

  local class GC gcCAD;
  gcCAD = deviceCAD.CreateGC();

  drawGraphForCAD(gcCAD);

  deviceCAD.Close();
  CloseCAD(CADgraph);
}
```