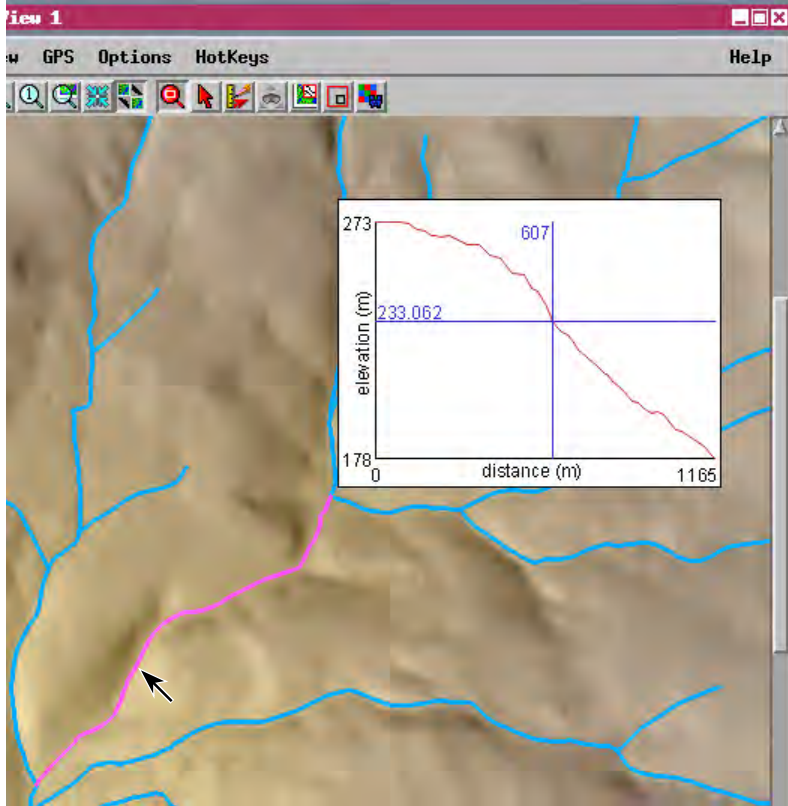
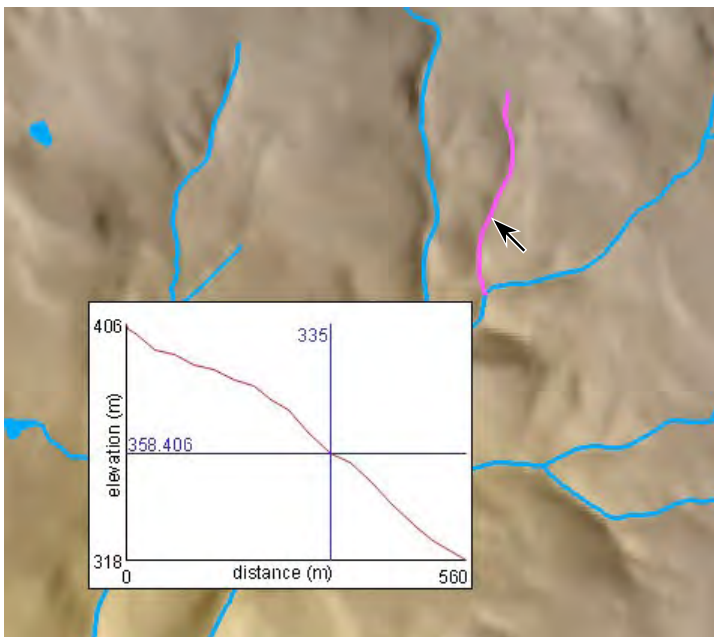


## Sample GraphTip Script

# Profile of Nearest Line



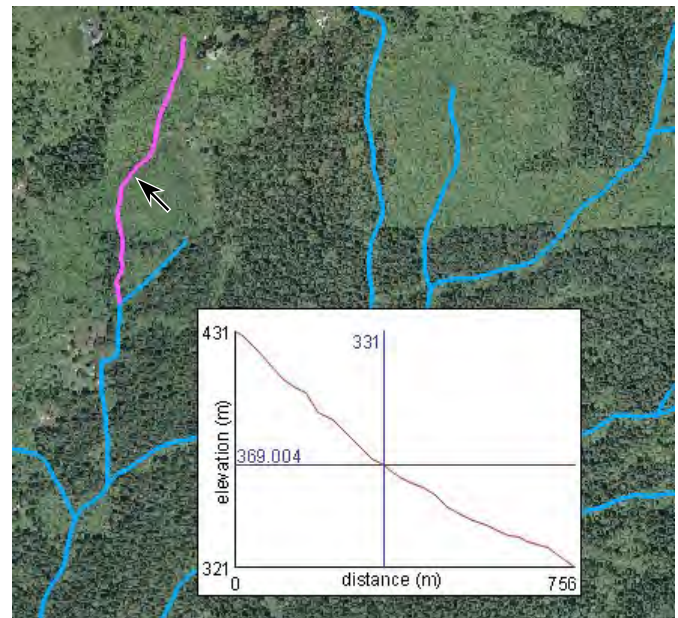
A GraphTip can include multiple graphic elements including text, with each element independently positioned. In this example the vertical and horizontal cross-hair lines indicate the position of the cursor along the line in both distance and elevation.



In this example the position of the GraphTip is also varied depending on the position of the cursor in the View in order to prevent it from obscuring the highlighted line element.

The illustrations on this page show a GraphTip example that executes automatically whenever the cursor is near a vector line. The line nearest the cursor is highlighted and a GraphTip pops-in automatically to show an elevation profile computed along the line using values from an elevation raster. Excerpts of the Display Control Script that creates this GraphTip are shown on the opposite side of this page, and the complete script is available for download.

Creating GraphTips for your saved groups, layouts, and atlases provides a data exploration capability that is available automatically to anyone using the data. A GraphTip can access and process data from several layers in the view and pop-in a graphical presentation of the information for the current cursor position. It can even detect the nearest point, line, or polygon in a vector layer and compute characteristics of that feature from some other layer (or from an object that is not currently part of the Group or Layout). For example, a Graph tip could find the vector polygon that encloses the cursor location, use that area to determine some statistical properties of a raster layer or off-line raster object, and present a graph of the result (such as a frequency histogram of elevation values from an elevation raster).



This GraphTip obtains elevation values from a raster object that is the bottom layer in the group, and highlights vector lines in the top layer in the group. Intervening layers can be turned off or on without affecting the operation of the GraphTip, as shown by the photo layer turned on in this illustration.

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 Excerpt for Line Profile GraphTip (LineProfileGraphTip.sml)

procedure to draw the graph with the given polyline

```

proc drawGraph (class POLYLINE graphLine, numeric vertexNum) {
  drawGraphAxes(graphLine); plot out the axes first

  gc.SetColorRGB(200, 50, 50); set profile color and plot point zero
  local class POINT2D linePoint = graphLine.GetVertex(0);
  local class POINT2D graphPoint = transPointToGraph(linePoint, graphLine);
  gc.DrawPoint(graphPoint.x, graphPoint.y);

  local numeric i;
  for i=1 to graphLine.GetNumPoints()-1 { plot the rest of the profile vertices
    linePoint = graphLine.GetVertex(i);
    graphPoint = transPointToGraph(linePoint, graphLine);

    if (isNull(linePoint.y)) { if null skip point and move to next
      linePoint = graphLine.GetVertex(i+1);
      graphPoint = transPointToGraph(linePoint, graphLine);
      gc.MoveTo(graphPoint.x, graphPoint.y);
    }
    else gc.DrawTo(graphPoint.x, graphPoint.y);
  }

  local class COLOR horizLineColor, vertLineColor;
  horizLineColor.red = 20;
  horizLineColor.green = 20;
  horizLineColor.blue = 80;
  vertLineColor.red = 20;
  vertLineColor.green = 20;
  vertLineColor.blue = 80;
  local class POINT2D graphCircle = graphLine.GetVertex(vertexNum);
  graphCircle = transPointToGraph(graphCircle, graphLine);

  gc.SetColorRGB(vertLineColor.red, vertLineColor.green,
    vertLineColor.blue, 100);
  gc.MoveTo(graphCircle.x, topGraphOffset);
  gc.DrawTo(graphCircle.x, getHeight() - bottomGraphOffset); draw the vertical line

  gc.DrawTextSetColors(vertLineColor); draw label
  string dist = NumToStr(int(graphLine.GetVertex(vertexNum).x));
  local numeric x, y; set x position of measurement label
  if (graphCircle.x < (leftGraphOffset + gc.TextGetWidth(dist) + 1))
    x = graphCircle.x + 1; draw on right
  else x = graphCircle.x - gc.TextGetWidth(dist) - 1; draw on left

  y = topGraphOffset + fontHeight; set y position of measurement label
  gc.DrawTextSimple(dist, x, y); draw distance measurement label

  if (!isNull(graphCircle.y)) { draw horizontal line
    gc.SetColorRGB(horizLineColor.red, horizLineColor.green,
      horizLineColor.blue, 100); set the line color

    gc.MoveTo(leftGraphOffset, graphCircle.y); draw line
    gc.DrawTo(getWidth() - rightGraphOffset, graphCircle.y);

    gc.DrawTextSetColors(horizLineColor); draw label
    string elev = NumToStr(graphLine.GetVertex(vertexNum).y);
    if (graphCircle.x > (getWidth() - rightGraphOffset - leftGraphOffset)/2) {
      gc.DrawTextSimple(elev, leftGraphOffset+1, graphCircle.y-1);
    }
    else { draw on left
      gc.DrawTextSimple(elev, getWidth() - rightGraphOffset -
        gc.TextGetWidth(elev)-1, graphCircle.y-1); draw on right
    }
  }
}

```

procedure to convert the polyline from obj to map coordinates

```

func class POLYLINE convertObjectToMap(class POLYLINE line) {
  local class POLYLINE ret;
  local class POINT2D obj, map;

  local numeric i;
  for i=0 to line.GetNumPoints()-1 {
    obj = line.GetVertex(i);
    map = ObjectToMap(line.Vector, obj.x, obj.y, vecGeoref);
    ret.AppendVertex(map);
  }

  return ret;
}

func OnViewDataTipShowRequest ( predefined function called when the cursor pauses triggering a datatip action event
  class GRE_VIEW view,
  class POINT2D point,
  class TOOLTIP datatip) {

  datatip.Delay = 500;
  local class POINT2D cursor = point; store the cursor position
  get the cursor position in map coords

  local class TRANSPARM screenToView = ViewGetTransViewToScreen(view, 1);
  local class TRANSPARM viewToMap =
    ViewGetTransMapToView(view, vectorLayer.Projection, 1);
  point = TransPoint2D(point, screenToView);
  point = TransPoint2D(point, viewToMap); translate 16 pixel distance to map projected distance

  local class POINT2D tmppoint0;
  tmppoint0.x = 0; tmppoint0.y = 0;
  tmppoint0 = TransPoint2D(tmppoint0, screenToView);
  tmppoint0 = TransPoint2D(tmppoint0, viewToMap);

  local class POINT2D tmppoint;
  tmppoint.x = sqrt(128); tmppoint.y = sqrt(128);
  tmppoint = TransPoint2D(tmppoint, screenToView);
  tmppoint = TransPoint2D(tmppoint, viewToMap);

  local numeric dist = computeDistance(tmppoint0, tmppoint);

  local numeric lineNum = get the line from the cursor position
  FindClosestLine(line.Vector, point.x, point.y, vecGeoref, dist);
  if (lineNum == 0) return -1; if not close enough, don't display graph

  local class POLYLINE line = GetVectorLine(line.Vector, lineNum);
  line = convertObjectToMap(line);

  vectorLayer.line.HighlightSingle(lineNum); highlight the line
  view.RedrawLayer(vectorLayer); get the closest vertex for display

  local numeric vertexNum = line.FindClosestVertex(point);
  get the line to graph - x-dimension is distance, y is elevation

  local class POLYLINE graphLine = constructGraphLine(line);
  createGC(); create the graphics context to draw the graph to
  drawGraph(graphLine, vertexNum); draw the graph

  local class POINT2D offset;
  offset = computeOffset(line, view, cursor); compute Image tip offset and display
  datatip.SetImageTip(imagedev, maskdev, offset);
  return 1;
}

```