# Enhanced DataTips and GraphTips

Standard DataTips automatically pop into view to show information (in text form) about the cursor location from the geodata layers in the View window. With a Display Control Script you can create more informative *Enhanced DataTips* and *GraphTips* that pop in with customized or computed content. A Display Control Script can access the cursor position as the DataTip is about to appear, use this position to obtain data from one or more layers in the view (raster cell value, database attribute value for the enclosing polygon, and so on), and present the data or results computed from it. To create an Enhanced DataTip, the script restyles, adds to, and/or replaces the standard DataTip text to provide custom text information. In a GraphTip, the script replaces the text-only DataTip with information presented primarily in graphical form using SML class methods to draw graphical elements and text. A Display Control Script can transform the DataTip into a powerful information center that automatically presents a wealth of dynamically-changing information every time the user of your spatial data moves the mouse cursor.

As an example, consider a group or layout that includes an elevation raster. A standard DataTip can report the elevation value for the raster cell under the cursor (along with information from other layers in the view). The sample Enhanced DataTip and GraphTip illustrated here use the elevation raster to compute the local terrain slope and slope direction (aspect) at the cursor position and pop-in the result. The Enhanced DataTip presents the results as labeled and formatted text, whereas the GraphTip combines graphics and text. Excerpts from the Display Control Script that creates the GraphTip are shown on the opposite side of this plate, and both scripts are available for download from www.microimages.com/freestuf/scripts.htm.
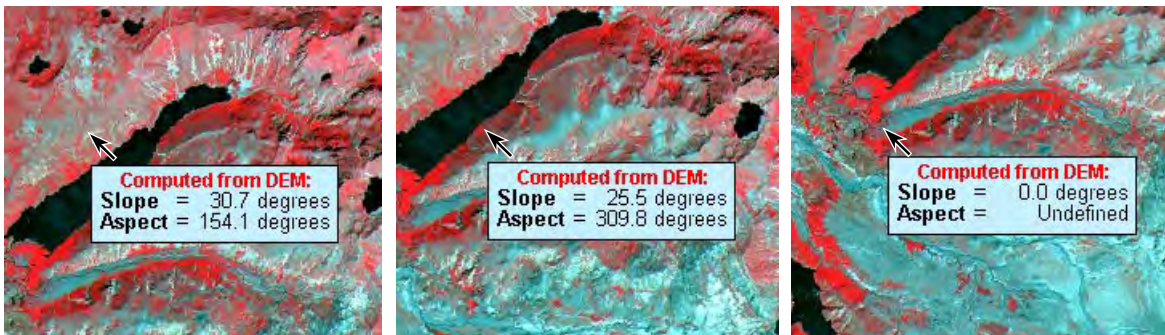
Enhanced DataTip created by the SlopeAspectDataTip script. Since aspect is undefined for flat areas, the script does not compute aspect if slope = 0, but instead reports aspect as "Undefined."

Display Control Scripts can be created and saved with your TNT layouts and groups by selecting Layout / Edit Control Script or Group / Edit Control Script in the Layout or Group Controls window, respectively. When the group or layout is opened, the Display Control Script executes automatically to activate the Enhanced DataTip or GraphTip, which then appears automatically whenever the mouse cursor pauses over any cell in the view. And when you include the layout in an atlas, the Enhanced DataTip or GraphTip automatically appears in the TNTatlas window. You can therefore use Display Control Scripts to add dynamic custom data exploration capability to the geodata sets you prepare for clients or other end users. This capability is activated automatically and requires no selection or set-up by them.
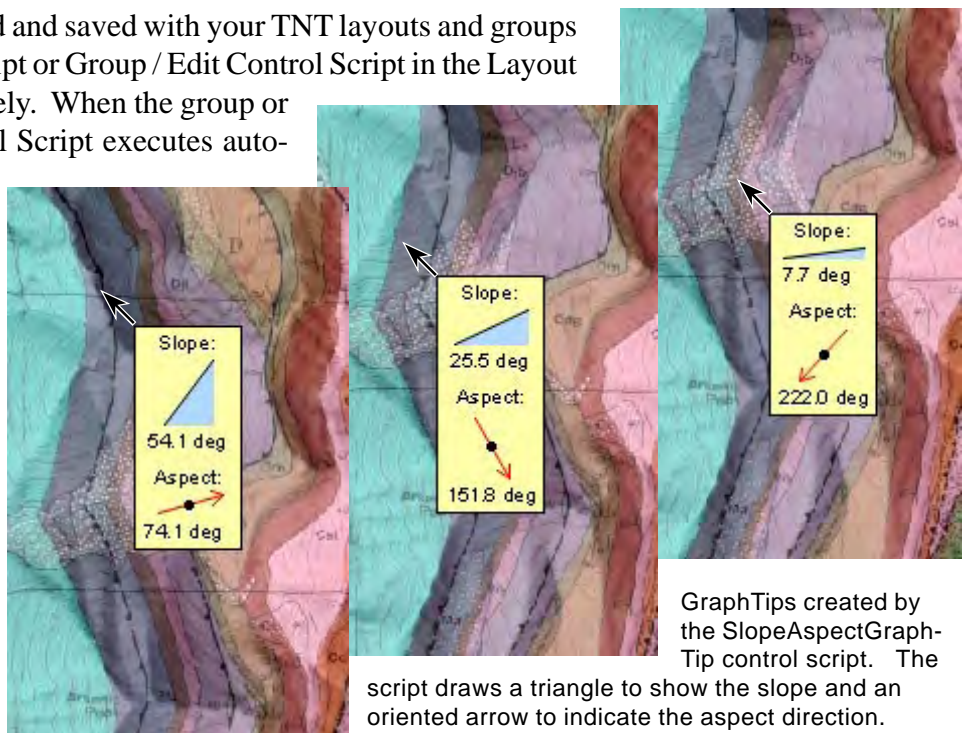
GraphTips created by the SlopeAspectGraph-Tip control script.   The script draws a triangle to show the slope and an oriented arrow to indicate the aspect direction.

Script Excerpt for Slope-Aspect GraphTip
(SlopeAspectGraphTip.sml)

```
proc OnGroupCreateView (          called when View is created for group
  class GRE_GROUP group           predefined class variable
  ) {
  DEM_layer = group.FirstLayer;            get first layer in group
  DispGetRasterFromLayer(DEM, DEM_layer);       get DEM from layer
  w = ColScale(DEM);
  h = LinScale(DEM);                get line and column cell size
  nullDEM = NullValue(DEM);         and null value for DEM
  }

func OnViewDataTipShowRequest (    called when DataTip
  class GRE_VIEW view,             event is triggered
  class POINT2D point,
  class TOOLTIP datatip            predefined class variables
  ) {
  numeric retval = 1;             in GraphTip use only what is created by script

                    get transform from screen to layer coordinates
  trans = view.GetTransLayerToScreen(DEM_layer, 1);
  ptLayer = trans.ConvertPoint2DFwd(point);     cursor position in
                                                 layer coordinates

  lin = floor(ptLayer.y);         line and column position
  col = floor(ptLayer.x);         of DEM cell under cursor

  if ( DEM[lin,col] <> nullDEM ) {      if cursor is not over a null cell

    dzX = ( DEM[lin, col + 1] - DEM[lin, col - 1] ) / (2 * w);    compute x
    dzY = ( DEM[lin - 1, col] - DEM[lin + 1, col] ) / (2 * h);    and y
                                                                  derivatives
    slope = atand( sqrt( sqr(dzX) + sqr(dzY) ) );                 and slope

    baseline = 40 * cosd(slope);
    triHeight = baseline * tand(slope);      compute dimensions of
    tribaseY = 17 + round(triHeight);        slope triangle graphic

    compute aspect
    if (slope == 0) {          undefined for flat areas; no pointer needed
      aspect$ = "Undefined";
      aspHeight = 5;                 set drawing height for area where
      }                             arrow would otherwise be drawn

    else {
      if (dzY != 0)  {          check for division by zero
        aspect = deg * atan2(-dzX, -dzY);

        if (aspect < 0 ) then
          aspect += 360;
        }
      else
        aspect = 90;           when dzY = 0

      aspect$ = sprintf("%.1f deg", aspect);      create aspect label string
      aspectDrawAngle = (aspect - 90);    drawing angle for aspect pointer

        compute height of area pointer will be drawn in
      aspHeight = round( abs( 18 * sind(aspectDrawAngle) ) );
      if (aspHeight < 5) then
        aspHeight = 5;         minimum drawing height for pointer area
      }

    compute dimensions of GraphTip drawing area based
    on triangle graphic and aspect pointer dimensions

    width = 54;               width of drawing area
    center.x = width / 2;
    center.y = tribaseY + 35 + aspHeight;    position of center for
    height = center.y + aspHeight + 18;      rotating pointer
                                             height of drawing area

    imagedev.Create(height,width);    create GraphTip drawing area
    maskdev.Create(height,width);     with specified dimensions
```

```
                            begin drawing GraphTip elements
    gc = imagedev.CreateGC();          create graphics context for GraphTip

    gc.SetColorRGB(100,100,67,100);    yellow background
    gc.FillRect(0, 0, width, height);  rectangle for GraphTip

    gc.SetColorName("black");          black frame
    gc.DrawRect(0, 0, width - 1, height - 1);    around GraphTip

    gc.DrawTextSetFont("ARIAL.TTF");
    gc.DrawTextSetHeightPixels(10);

    gc.DrawTextSimple("Slope:", 12, 12);       draw Slope and Aspect titles
    gc.DrawTextSimple("Aspect:", 10, tribaseY + 30);

        draw slope graphic (right triangle) and value label
    x_points[1] = center.x - baseline / 2;     compute coordinates
    x_points[2] = center.x + baseline / 2;     for ends of baseline
    y_points[1] = tribaseY;                    and store in arrays
    y_points[2] = tribaseY;

    if (slope < 5) {          draw horizontal line instead of triangle
      gc.MoveTo(x_points[1], y_points[1]);
      gc.DrawTo(x_points[2], y_points[2]);
      }

    else {                    draw triangle
      x_points[3] = center.x + baseline / 2;   x and y coordinates of
      y_points[3] = tribaseY - triHeight;      top vertex of triangle

      gc.SetColorRGB(70, 85, 100, 100);
      gc.DrawPolyLine(x_points, y_points, 3);      draw light blue
      gc.FillPolyLine(x_points, y_points, 3);      slope triangle

        draw black line along slope (hypotenuse of right triangle)
      gc.MoveTo(x_points[1], y_points[1]);       move to lower left
      gc.SetColorRGB(0, 0, 0, 100);              triangle vertex and
      gc.DrawTo(x_points[3], y_points[3]);       draw line to top vertex
      }
                    draw text for slope value below triangle
    gc.DrawTextSimple( sprintf("%.1f deg", slope), 7, tribaseY + 12);

    if (slope > 0) {          draw the aspect pointer (line with
                              arrowhead added) and label
        find and store coordinates of end of pointer line
      arrowEnd.x = center.x + 18 * cosd(aspectDrawAngle);
      arrowEnd.y = center.y + 18 * sind(aspectDrawAngle);

        move to start of pointer line and draw to end
      gc.MoveTo( center.x - 16 * cosd(aspectDrawAngle), center.y - 16
                 * sind(aspectDrawAngle) );
      gc.SetColorRGB(100,0,0,100);
      gc.DrawTo(arrowEnd.x, arrowEnd.y );          draw arrowhead

      gc.DrawArrow(arrowEnd.x, arrowEnd.y, aspectDrawAngle, 6, 30, "Open");

      gc.SetColorRGB(0,0,0,100);        draw small red circle at
      gc.FillCircle(center.x, center.y, 2);    center of rotation of pointer
      }
                    draw text with aspect value under pointer
    gc.DrawTextSimple(aspect$, 5, center.y + aspHeight + 12);

      set the rendered image and mask as source for the GraphTip
    datatip.SetImageTip(imagedev, maskdev, offset);
    }
  else
    retval = -1;              if outside DEM or on null cell
                              then don't show any datatip

  return (retval);
}
```