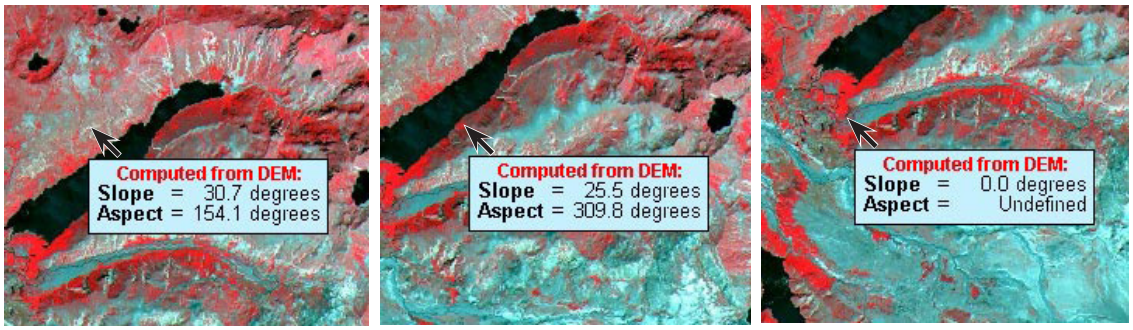
v70 新機能

機能強化版データティップとグラフティップ

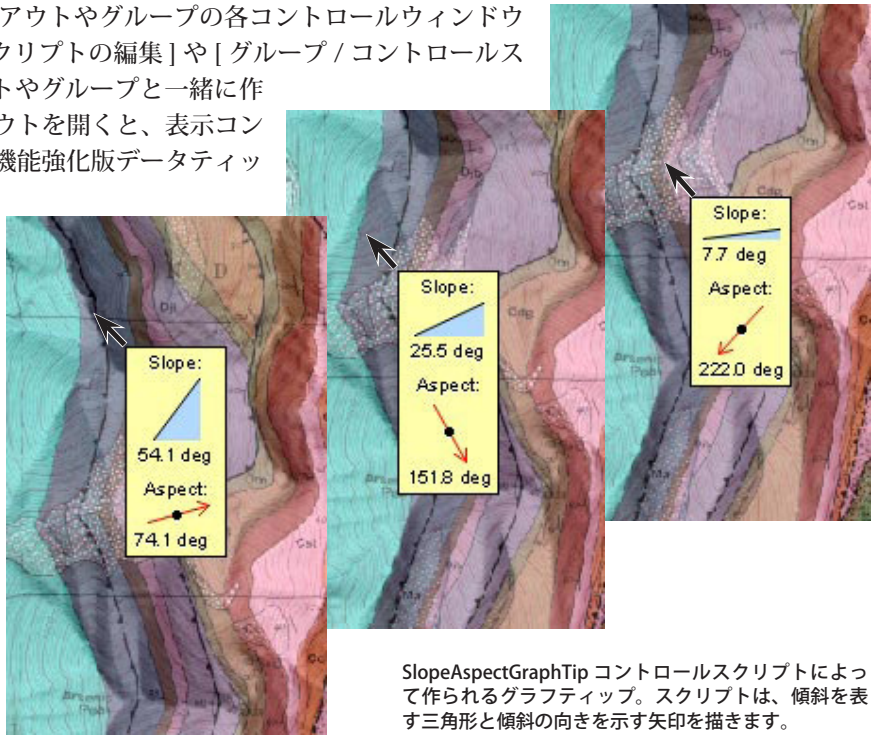
標準のデータティップは、表示ウィンドウ上でカーソルを置いた場所にある地理データレイヤの情報をテキスト形式で自動で表示します。表示コントロールスクリプトを使うと、より情報量が豊富な機能強化版**データティップ**や**グラフティップ**を作って、自分の用途に合った計算処理を行うコンテンツを表示することができます。表示コントロールスクリプトは、データティップが表示されるカーソル位置にアクセスし、この位置の 1 つまたは複数のレイヤからデータ (ラスタのセル値、囲んでいるポリゴンのデータベース属性値など) を得て、それから計算される結果を表示します。機能強化版データティップでは、スクリプトにより標準のデータティップのテキストスタイルを変えたりして、目的に合ったテキスト情報を提供できます。グラフティップでは、スクリプトを使ってテキストのみのデータティップの代わりに図を主体とした表現を行えます。SML のクラスメソッドを使って、グラフィカルな要素やテキストを描けます。表示コントロールスクリプトによって、データティップは強力な情報センターに変わり、他の人は、あなたの作った空間データの上でカーソルを動かすたびに、ダイナミックに変化する多くの情報を自動で表示させることができます。

例として、標高ラスタを含むグループやレイアウトを考えてみましょう。通常のデータティップは、カーソルの下のラスタセルの標高値を (他のレイヤからの情報と一緒に) レポートします。下図に示す機能強化版データティップとグラフティップのサンプルは、標高ラスタを使ってカーソル位置の局所的な地形の傾斜や傾斜の向き (方角) を計算し、表示します。機能強化版データティップは、結果をラベルや書式設定されたテキストとして表示します。それに対してグラフティップは、図とテキストを組み合わせて表示します。次ページにグラフティップを作成する表示コントロールスクリプトの例 (抜粋) があります。スクリプトは www.microimages.com/freestuf/scripts.htm からタウンロードできます。



SlopeAspectDataTip スクリプトで生成される機能強化版データティップ。平坦な場所では方角が定義されないので、スクリプトは傾斜＝ 0 であれば方角は計算されません。代わりに “Undefined(未定義)” と表示します。

表示コントロールスクリプトは、レイアウトやグループの各コントロールウィンドウにおいて [レイアウト / コントロールスクリプトの編集] や [グループ / コントロールスクリプトの編集] を選択して、レイアウトやグループと一緒に作成や保存ができます。グループやレイアウトを開くと、表示コントロールスクリプトが自動で実行され、機能強化版データティップやグラフティップがアクティブになり、表示中の任意のセル上にマウスカーソルを置くたびに自動で現れます。また、TNT アトラスにレイアウトを含めると、機能強化版データティップやグラフティップが、アトラスのウィンドウに自動的に現れます。このように、表示コントロールスクリプトを使うと、顧客や他のユーザ向けに準備した地理データ上に、あなたの目的に合わせてダイナミックにデータを検索する機能を追加できます。この機能は表示コントロールスクリプトによって自動的にアクティブになり、顧客が選択したり設定する必要はありません。



SlopeAspectGraphTip コントロールスクリプトによって作られるグラフティップ。スクリプトは、傾斜を表す三角形と傾斜の向きを示す矢印を描きます。

TNT のスクリプト言語がスクリプトやクエリにおいてどのように利用できるか紹介するために、多くのサンプルスクリプトが用意されています。これらのスクリプトは www.microimages.com/freestuf/scripts.htm からダウンロードできます。

傾斜と方角を表示するグラフティップのスクリプト (抜粋)

(SlopeAspectGraphTip.sml)

```
proc OnGroupCreateView (                          グループに対して表示ウィンドウを作るとき呼ぶ
  class GRE_GROUP group                           定義済みクラス変数
  ) {
  DEM_layer = group.FirstLayer;                   グループの最初のレイヤをセット
   DispGetRasterFromLayer(DEM, DEM_layer);        レイヤから DEM レイヤをセット
   w = ColScale(DEM);
   h = LinScale(DEM);                             DEM の行と列のセルサイズ及びヌル値をセット
   nullDEM = NullValue(DEM);
   }

func OnViewDataTipShowRequest (                   データティップイベントを起動する際呼ぶ
  class GRE_VIEW view,
  class POINT2D point,                            定義済みクラス変数
  class TOOLTIP datatip
  ) {
  numeric retval = 1;                             グラフティップの中でスクリプトで作成されるもののみ使う

                                                  画面からレイヤ座標への変換
   trans = view.GetTransLayerToScreen(DEM_layer, 1);
   ptLayer = trans.ConvertPoint2DFwd(point);      レイヤ座標でのカーソル位置

   lin = floor(ptLayer.y);                        カーソルの下の DEM セルの行と列の位置
   col = floor(ptLayer.x);

   if ( DEM[lin,col] <> nullDEM ) {               カーソルがヌルセルの上にない場合

      dzX = ( DEM[lin, col + 1] - DEM[lin, col - 1] ) / (2 * w);
      dzY = ( DEM[lin - 1, col] - DEM[lin + 1, col] ) / (2 * h);      x と y 方向の差分と傾斜の計算

      slope = atand( sqrt( sqr(dzX) + sqr(dzY) ) );

      baseline = 40 * cosd(slope);
      triHeight = baseline * tand(slope);
      tribaseY = 17 + round(triHeight);           傾斜の三角形の図の大きさを計算

      方角の計算
      if (slope == 0) {                           平坦な場所での処理；矢印の表示は不要
         aspect$ = "Undefined";
         aspHeight = 5;                           矢印が描かれる領域の描画高さを設定
         }

      else {
         if (dzY != 0) {                          0 での割り算のチェック
            aspect = deg * atan2(-dzX, -dzY);

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

         aspect$ = sprintf("%.1f deg", aspect);   方角を表すラベルの文字列を作成
         aspectDrawAngle = (aspect - 90);         方角を示す矢印を描く角度

            矢印を描く領域の高さの計算
         aspHeight = round( abs( 18 * sind(aspectDrawAngle) ) );
         if (aspHeight < 5) then
            aspHeight = 5;                        矢印の領域の描画高さの最小値
         }
            三角形の図と方角を示す矢印の大きさに基づいてグラフティップを描く領域の大きさを計算

      width = 54;                                 描画領域の幅
      center.x = width / 2;                       回転する矢印の中心位置
      center.y = tribaseY + 35 + aspHeight;
      height = center.y + aspHeight + 18;         描画領域の高さ

      imagedev.Create(height,width);              指定された大きさでグラフティップを描く領域を作る
      maskdev.Create(height,width);

                                                  グラフティップ要素の描画開始
      gc = imagedev.CreateGC();
                                                  グラフティップのグラフィック・コンテキストの作成
      gc.SetColorRGB(100,100,67,100);
      gc.FillRect(0, 0, width, height);           グラフティップの背景 ( 黄色の長方形 )

      gc.SetColorName("black");
      gc.DrawRect(0, 0, width - 1, height - 1);   黒のフレームでグラフティップを囲む

      gc.DrawTextSetFont("ARIAL.TTF");
      gc.DrawTextSetHeightPixels(10);
                                                  Slope と Aspect というタイトルを表示
      gc.DrawTextSimple("Slope:", 12, 12);
      gc.DrawTextSimple("Aspect:", 10, tribaseY + 30);

            傾斜の図 ( 直角三角形 ) と値のラベルを描く
      x_points[1] = center.x - baseline / 2;
      x_points[2] = center.x + baseline / 2;
      y_points[1] = tribaseY;                     ベースラインの両端の座標を計算して配列に格納
      y_points[2] = tribaseY;

      if (slope < 5) {                            三角形の代わりに水平線を描く
         gc.MoveTo(x_points[1], y_points[1]);
         gc.DrawTo(x_points[2], y_points[2]);
         }

      else {                                      三角形を描く
         x_points[3] = center.x + baseline / 2;
         y_points[3] = tribaseY - triHeight;      三角形の上の頂点の xy 座標

         gc.SetColorRGB(70, 85, 100, 100);
         gc.DrawPolyLine(x_points, y_points, 3);
         gc.FillPolyLine(x_points, y_points, 3);  傾斜を示す三角形を水色で描く

               傾斜 ( 直角三角形の斜辺 ) に沿って黒いラインを描く
         gc.MoveTo(x_points[1], y_points[1]);
         gc.SetColorRGB(0, 0, 0, 100);
         gc.DrawTo(x_points[3], y_points[3]);     三角形の左下の頂点に移動して上の頂点までラインを描く
         }
            三角形の下に傾斜の値のテキストを表示
      gc.DrawTextSimple( sprintf("%.1f deg", slope), 7, tribaseY + 12);
      if (slope > 0) {                            方角を示す矢印 ( 矢の形が加えられたライン ) とラベルを描く

               矢印のラインの端の座標を読み取って保存
         arrowEnd.x = center.x + 18 * cosd(aspectDrawAngle);
         arrowEnd.y = center.y + 18 * sind(aspectDrawAngle);

               矢印のラインの始点に移動して終点まで描画
         gc.MoveTo( center.x - 16 * cosd(aspectDrawAngle), center.y - 16
                  * sind(aspectDrawAngle) );
         gc.SetColorRGB(100,0,0,100);
         gc.DrawTo(arrowEnd.x, arrowEnd.y );
                                                  矢の先端を描く
         gc.DrawArrow(arrowEnd.x, arrowEnd.y, aspectDrawAngle, 6, 30, "Open");

         gc.SetColorRGB(0,0,0,100);               矢印の回転中心に小さな赤丸を描く
         gc.FillCircle(center.x, center.y, 2);
         }
                                                  矢印の下に方角の値をテキストで示す
      gc.DrawTextSimple(aspect$, 5, center.y + aspHeight + 12);

         グラフティップのソースとしてレンダリングした画像とマスクを設定
      datatip.SetImageTip(imagedev, maskdev, offset);
      }
   else
      retval = -1;
                                                  DEM の外またはヌルセルの上ではデータティップを表示しない
   return (retval);
   }
```