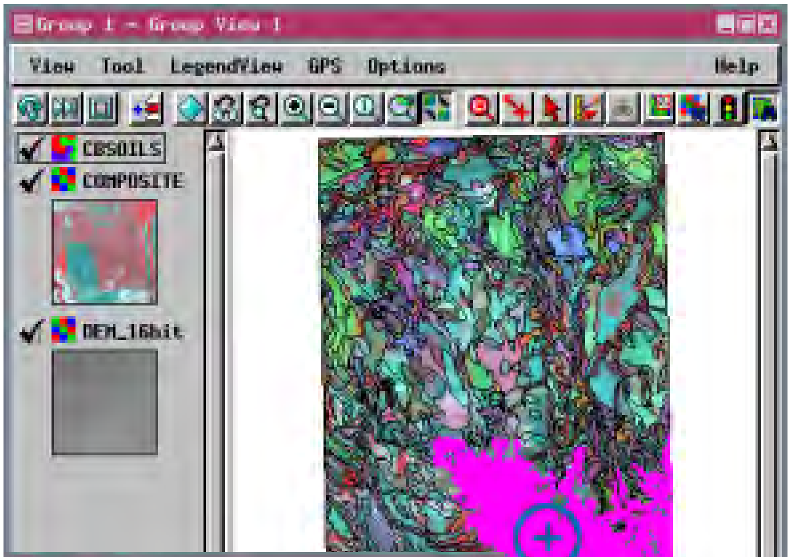
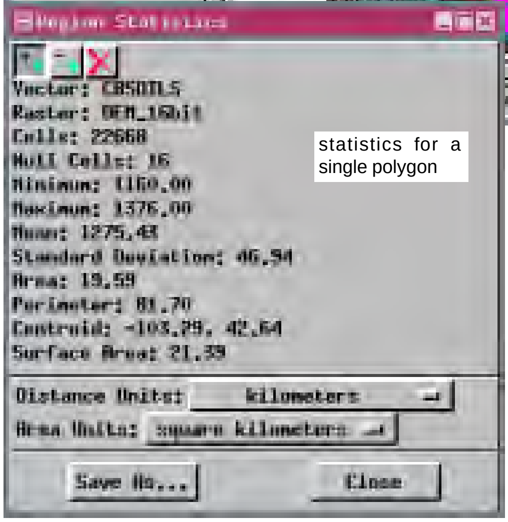
Sample SML Tool Script

Region Statistics



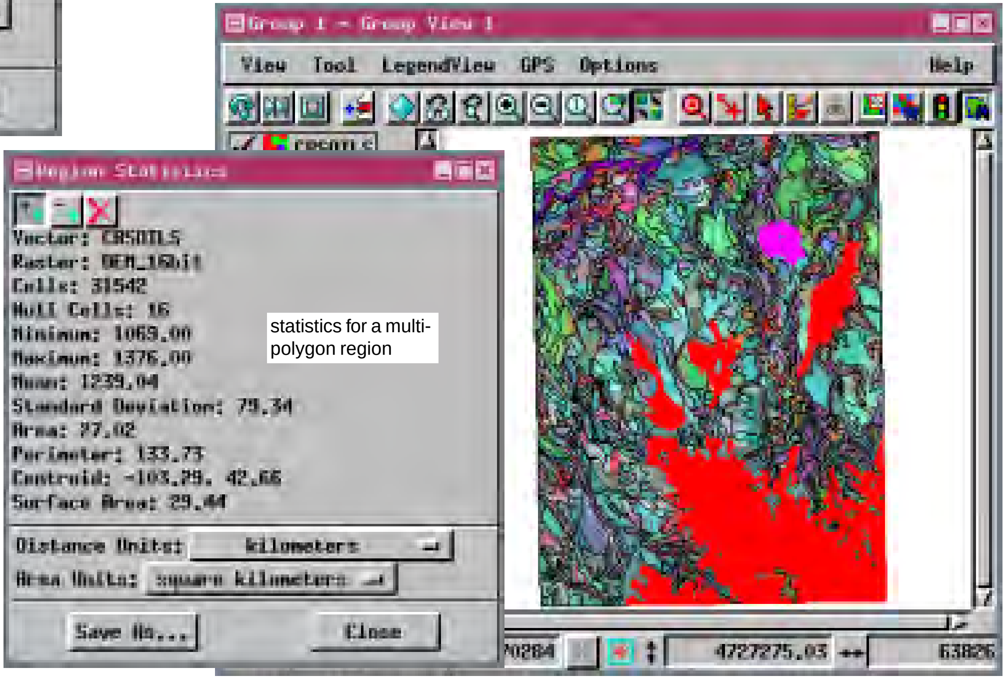
The Region Statistics tool script demonstrates how you can design a custom tool to visually select polygons and convert them to a temporary region to define the area for action on another coregistered layer. A tool with these functions could then use the region in a variety of operations. In this example, the region is used in the simple operation of extracting statistics from a raster object. This same tool could be modified to perform many other functions with the regions it creates. For example, it could extract points, lines, or polygons from another vector layer or use the extract functions to create rasters of the regions.



```
Vector: CRSOILS
Raster: DEM_16bit
Cells: 22668
Null Cells: 16
Minimum: 1160.00
Maximum: 1376.00
Mean: 1275.48
Standard Deviation: 46.94
Area: 19.59
Perimeter: 81.70
Centroid: -103.29, 42.64
Surface Area: 21.39
Distance Units: kilometers
Area Units: square kilometers
```

statistics for a single polygon

The Region Statistics script lets you select one or more vector polygons then calculates a set of statistics from an underlying raster for the area covered by the selected polygon(s). The top vector layer is used for polygon selection and the statistics are calculated for the bottom raster layer. The statistics calculated include the number of cells, the number of null cells, the minimum and maximum cell values, the mean cell value, the standard deviation of cell values, the area of the region, its perimeter length, the coordinates of the centroid, and a surface area estimation. You can choose from any of the 25 length units and 13 area units standardly available throughout the TNT products for viewing the perimeter and area statistics. You can also save the calculated statistics as a text file. If you select the same text file again, the current statistics will be appended to earlier entries.



```
Vector: CRSOILS
Raster: DEM_16bit
Cells: 31542
Null Cells: 16
Minimum: 1069.00
Maximum: 1376.00
Mean: 1239.04
Standard Deviation: 79.34
Area: 27.02
Perimeter: 133.73
Centroid: -103.29, 42.66
Surface Area: 29.44
Distance Units: kilometers
Area Units: square kilometers
```

statistics for a multi-polygon region

This custom script reports the same statistics as the Area Statistics tool script. The method of area definition differs between the two—one has you draw the area and the other uses selected polygons to create a region. Together these scripts provide an excellent example of how to use the sample scripts provided by MicroImages to put together the pieces you need for your own custom tool. Think of the sample macro and tool scripts provided as a series of modules to be reused for the same or different purposes in a script that creates the custom tool you want. This script, for example, has modules concerned with selecting polygons, making a region from selected polygons, using a status window, calculating standard raster statistics for a local area, and estimating surface area from an elevation raster.

Macro and Tool Scripts can be created using SML in any TNTmips process that uses a View window (Options / Customize from the View window menu bar). These scripts are then available from an icon, which you select or design, on the toolbar. Sample scripts have been prepared to illustrate how you might use these features, which are available only in TNTmips 6.4 or later, to assist with specific tasks you perform on a regular basis. If possible, the full script is printed below for your quick perusal. When a script is too long to fit on one page, key sections are reproduced below. The sample Tool Script illustrated can be downloaded from the SML script exchange at www.microimages.com/sml/ftpsmllink/TNT_Products_V6.4_CD.

Partial Script for Region Statistics (regstatp.sml)

checks to see if layers are valid

```
func checkLayer() {
    local boolean valid = false;
```

gets layer name if top layer is vector, gives error if not

```
    if (Group.LastLayer.Type == “Vector”) {
        vectorLayer = Group.LastLayer;
        DispGetVectorFromLayer(targetVector, vectorLayer);
        if (targetVector.$Info.NumPolys > 0) {
            vectorName$ = vectorLayer.Name;
            valid = true;
            }
        else vectorName$ = “No polygons!”;
        }
    else vectorName$ = “Not a vector!”;
```

gets layer name if bottom layer is valid raster type, gives error if not

```
    if (Group.FirstLayer.Type == “Raster”) {
        rasterLayer = Group.FirstLayer;
        DispGetRasterFromLayer(targetRaster, rasterLayer);
        if (targetRaster.$Info.Type != “binary” and
            targetRaster.$Info.Type != “4-bit unsigned” and
            targetRaster.$Info.Type != “8-bit signed” and
            targetRaster.$Info.Type != “8-bit unsigned” and
            targetRaster.$Info.Type != “16-bit signed” and
            targetRaster.$Info.Type != “16-bit unsigned” and
            targetRaster.$Info.Type != “32-bit signed” and
            targetRaster.$Info.Type != “32-bit unsigned” and
            targetRaster.$Info.Type != “32-bit float” and
            targetRaster.$Info.Type != “64-bit signed” and
            targetRaster.$Info.Type != “64-bit unsigned” and
            targetRaster.$Info.Type != “64-bit float”) {
            rasterName$ = “Type not supported!”;
            valid = false;
            }
        else
            rasterName$ = rasterLayer.Name;
        }
    else {
        rasterName$ = “Not a raster!”;
        valid = false;
        }
    return valid;
    }

proc cbRedraw() {
    local numeric larea, lperimeter, lsurface;
```

scales the values to specified units

```
    larea = areaScale * area;
    lperimeter = distScale * perimeter;
    lsurface = areaScale * surface;
    if (gc == 0) return;
    ActivateGC(gc);
```

redraws Region Statistics window when new statistics computed

```
    SetColorName(“gray75”);
    FillRect(0, 0, da.width, da.height);
    SetColorName(“black”);
    if (cells > 0) {
        DrawInterfaceText(sprintf(“Vector: %s\nRaster: %s\nCells: %d\nNull Cells: %d\nMinimum:
        %.2f\nMaximum: %.2f\nMean: %.2f\nStandard Deviation: %.2f\nArea: %.2f\nPerimeter:
        %.2f\nCentroid: %.2f, %.2f\nSurface Area: %.2f”,
        vectorName$, rasterName$, count, cells - count, min, max, mean, stdDev, larea, lperimeter,
        centroid.x, centroid.y, lsurface), 0, 10);
        }
    else DrawInterfaceText(sprintf(“Vector: %s\nRaster: %s\nCells:\nNull
Cells:\nMinimum:\nMaximum:\nMean:\nStandard Deviation:\nArea:\nPerimeter:\nCentroid:\nSurface
Area:”,vectorName$, rasterName$), 0, 10);
    }
```

procedures after right mouse button click (tool applied)

```
proc cbToolApply(class pointTool tool) {
    if (checkLayer()) {
```

defines local variables for statistics calculations

```
        local numeric sum, sumsqr, xscale, yscale, zscale;
        local numeric current, right, down, downright, elemNum;
        local region MyRgn;
        local class POINT2D point;
        local class StatusHandle status;
        local class StatusContext context;
        cells = 0; min = 0; max = 0; mean = 0; stdDev = 0; sum = 0; sumsqr = 0; count = 0; surface = 0;
        area = 0; perimeter = 0;
        centroid.x = 0; centroid.y = 0; current = 0; right = 0; down = 0; downright = 0;
        xscale = ColScale(targetRaster);
        yscale = LinScale(targetRaster);
        zscale = Group.FirstLayer.zscale;
```

gets point coordinates of tool

```
point.x = tool.Point.x;
point.y = tool.Point.y;

point = TransPoint2D(point, ViewGetTransViewToScreen(View, 1));
point = TransPoint2D(point, ViewGetTransMapToView(View, vectorLayer.Projection, 1));

elementNum = FindClosestPoly(targetVector, point.x, point.y, GetLastUsedGeorefObject(targetVector));
```

convert selected polygon(s) to region and transform to appropriate coordinate system

```
if (elementNum > 0) {
    MyRgn = ConvertVectorPolyToRegion(targetVector, elementNum,
    GetLastUsedGeorefObject(targetVector));

    if (regionMode$ == “plus”) {
        if (numRegions > 0) {
            MyRgn = RegionOR(reg, MyRgn);
            }
        numRegions += 1;
        vectorLayer.Poly.HighlightSingle(elementNum, 2);
        }
    else {
        if (numRegions > 1) {
            MyRgn = RegionSubtract(reg, MyRgn);
            numRegions -= 1;
            }
        else {
            MyRgn = RegionXOR(MyRgn, MyRgn);
            numRegions = 0;
            }
        vectorLayer.Poly.HighlightSingle(elementNum, 3);
        }
    reg = CopyRegion(MyRgn);
```

creates status dialog

```
    status = StatusDialogCreate(form);
    context = StatusContextCreate(status);
    StatusSetMessage(context, “Computing values...”);
```

calculates statistics for the region

```
    foreach targetRaster[lin, col] in reg {
        if (!IsNull(targetRaster)) {
            if (count == 0) {
                max = targetRaster;
                min = targetRaster;
                }
            else if (targetRaster > max) {
                max = targetRaster;
                }
            else if (targetRaster < min) {
                min = targetRaster;
                }
            sum += targetRaster;
            sumsqr += sqr(targetRaster);
            count += 1;
            }
```

method for estimating surface area

```
        if (!IsNull(targetRaster))
            current = targetRaster;
        if (!IsNull(targetRaster[lin,col+1]))
            right = targetRaster[lin,col+1];
        if (!IsNull(targetRaster[lin+1,col]))
            down = targetRaster[lin+1,col];
        if (!IsNull(targetRaster[lin+1,col+1]))
            downright = targetRaster[lin+1,col+1];
        surface += .5*sqrt(sqr(yscale*current*zscale-yscale*right*zscale)
                        +sqr(xscale*current*zscale-xscale*down*zscale)
                        +sqr(xscale*yscale));
        surface += .5*sqrt(sqr(yscale*downright*zscale-yscale*right*zscale)
                        +sqr(xscale*downright*zscale-xscale*down*zscale)
                        +sqr(xscale*yscale));
        cells += 1;
        }
    CloseRaster(targetRaster);
```

additional statistics calculated while avoiding division by zero

```
    if (count > 1) {
        mean = sum / count;
        stdDev = sqrt((sumsqr - sqr(sum) / count) / (count - 1));
        area = reg.$Data.GetArea();
        perimeter = reg.$Data.GetPerimeter();
        centroid = reg.$Data.GetCentroid();
        }
```

(see regstatp.sml for full script)