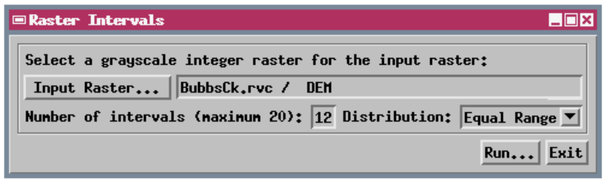
Sample Geospatial Script

Theme Mapping a Raster

The Raster Intervals script, which is excerpted on the opposite side of this plate, provides an example of a standalone processing script with its own interactive control dialogs. The script categorizes the cells in a grayscale raster into a selected number of grayscale-value intervals and outputs a raster with a single value for all cells in each such interval with an accompanying color palette for display. This procedure is analogous to the Theme Mapping option provided in the TNT products for setting up display styling for elements in a vector object based on the values of associated numerical attributes. The script allows the user to specify the number of intervals (up to a maximum of twenty) and the distribution type (equal count or equal interval). The Raster Intervals script can be used to categorize any grayscale integer raster containing a range of values such as elevation, slope angle, vegetation index, and so on.

The main control dialog window for the Raster Intervals sample script provides controls for selecting the input raster object, setting the number of intervals into which to divide the range of raster values, and selecting the distribution type (Equal Range or Equal Interval).

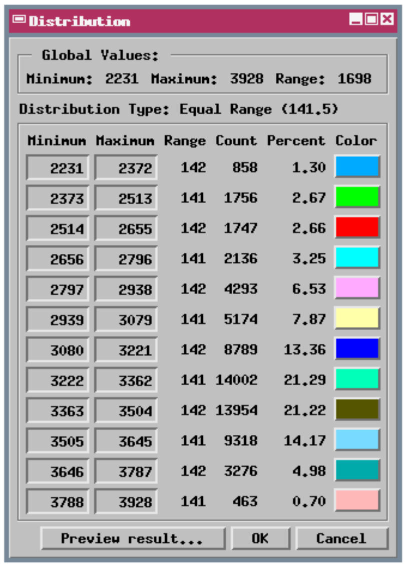

Raster Intervals

Select a grayscale integer raster for the input raster:

Input Raster... BubbsCk.rvc / DEM

Number of intervals (maximum 20): 12 Distribution: Equal Range

Run... Exit


Distribution

Global Values:
Minimum: 2231 Maximum: 3928 Range: 1698

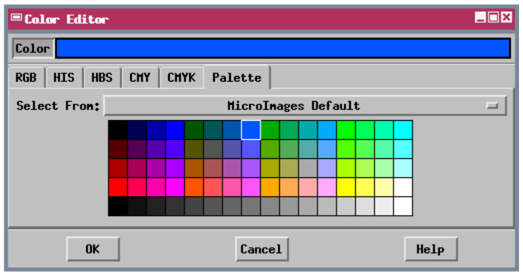
Distribution Type: Equal Range (141.5)

Minimum	Maximum	Range	Count	Percent	Color
2231	2372	142	858	1.30	🟦
2373	2513	141	1756	2.67	🟩
2514	2655	142	1747	2.66	🟥
2656	2796	141	2136	3.25	🟦
2797	2938	142	4293	6.53	🟪
2939	3079	141	5174	7.87	🟨
3080	3221	142	8789	13.36	🟦
3222	3362	141	14002	21.29	🟩
3363	3504	142	13954	21.22	🟫
3505	3645	141	9318	14.17	🟦
3646	3787	142	3276	4.98	🟦
3788	3928	141	463	0.70	🟥

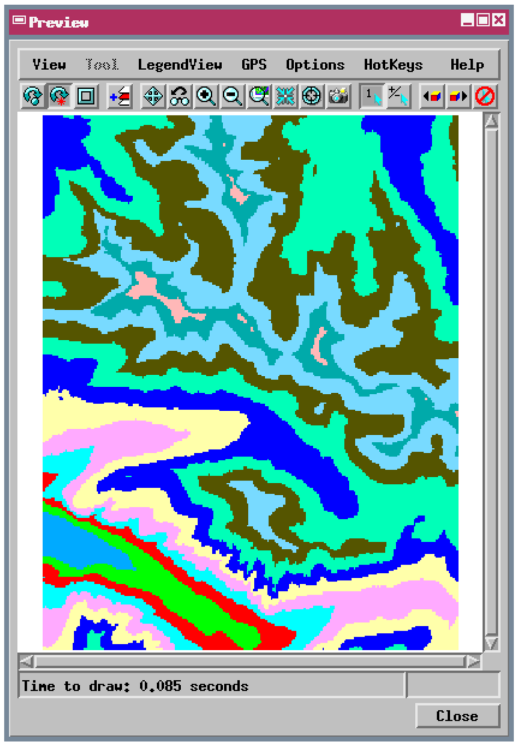
Preview result... OK Cancel

Pressing the Run button on the Raster Intervals window opens the Distribution window, which lists the value ranges initially determined by the script and statistical information about them. You can use this dialog to customize the ranges by editing the minimum or maximum values; adjacent ranges are automatically adjusted in response to your changes.

The script creates a color palette for the output raster object with default colors automatically assigned to each raster interval, as shown on the color buttons on the Distribution window. You can change the color for any interval by pressing its color button, which opens the standard TNT Color Editor window.



The Preview Result button on the Distribution window opens a Preview window that displays a temporary output raster based on the current range and color settings.



Pressing OK on the Distribution window prompts for a destination for the output raster and launches the final processing to create the output.

The Raster Intervals script provides three custom windows to allow the user to select the input raster and set up initial process parameters, view and edit the range values and palette colors, and preview the resulting raster. The Raster Intervals, Distribution, and Preview windows are each set up in the script using a dialog specification in XML to define the controls and their layout. The script provides a number of examples of how to set up callback actions for varied dialog controls, how to set up and use a dialog to display spatial objects created by a script, and how to create a series of interrelated script dialogs. In addition, the default color palette is stored in an XML text string that the script automatically parses into an XML document in memory. The script therefore provides an example of the methods used for accessing data in such an XML structure (in this case red, green and blue color values) when they are needed during processing.

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/downloads/scripts.htm.

Excerpts of Script for Raster Theme-Mapping (RasterIntervals.sml)

Procedure to update cell counts and percentages for intervals

```
proc updateCounts (numeric indexStart, numeric indexEnd)
    {
    local numeric i, counter;
```

reset cell counts for relevant intervals to 0

```
    for i = indexStart to indexEnd
        {
        cellCount[i] = 0;
        }
```

loop through input raster to get cell count for each specified interval

```
    for each RastIn
        {
        ++ counter;
        for i = indexStart to indexEnd
            {
            if (RastIn >= minRange[i] && RastIn <= maxRange[i]) then
                ++ cellCount[i];
            }
        }
```

compute cell percentage for each interval

```
    for i = indexStart to indexEnd
        {
        cellPct[i] = 100 * cellCount[i] / counter;
        }
    }
```

Procedure to make interval raster with color palette. Used for temporary preview raster and for output raster.

```
proc makeIntervalRaster(class Raster Rast)
    {
    local class GUI_CTRL_COLORBUTTON colorbutton;
    local class ColorMap palette;
    local class COLOR color;

    for each RastIn
        {
        value = RastIn;
        Rast = SetInterval(value);
        }

    CreateHistogram(Rast, 0);
    CreatePyramid(Rast);
    CopySubobjects(RastIn, Rast, "GEOREF");
```

create color palette for output raster from the colors in the distribution dialog

```
    for i = 1 to numInterval
        {
        local string interval$ = NumToStr(i);
        colorbutton = dlgDistrib.GetCtrlByID(sprintf("colorbut%s", interval$));
        color = colorbutton.GetColor();
        ColorMapSetColor(palette, i, color);
        }

    ColorMapWriteToRastVar(Rast, palette, "IntervalColors", "Color palette");

    SetNull(Rast, 0);
    }
```

Callback procedure when Distribution dialog opens. Sets initial colors of colorbuttons before opening (there is no way to do this using the XML structure, so must access the GUI control class for the colorbutton directly. Since this dialog is modal, it isn't created until DoModal method is called, so buttons must be set using the dialog's OnOpen callback.

```
proc onOpenDistrib ()
    {
    for i = 1 to numInterval
```

loop through the intervals to assign colors

```
        {
        local string interval$ = NumToStr(i);
```

assign red, green, and blue values for interval from XML structure to COLOR class instance

```
        class COLOR color;
        class XMLNODE colorNode;
        colorNode = docColor.GetElementByID(interval$);
        color.red = colorNode.GetAttributeNum("red");
        color.green = colorNode.GetAttributeNum("green");
        color.blue = colorNode.GetAttributeNum("blue");
```

get handle for relevant colorbutton control in dialog

```
        class GUI_CTRL_COLORBUTTON colorbutton;
        colorbutton = dlgDistrib.GetCtrlByID(sprintf("colorbut%s", interval$));
        colorbutton.SetColor(color);
        }
    }
```

Procedure called when Preview dialog is initialized. Used to create view within the dialog and to create a group to view the temporary raster.

```
proc onInitDialog ()
    {
    local class GUI_LAYOUT_PANE viewpane;
```

class instance for layout pane to contain the view

```
    local class widget viewpaneWidget;
```

class for widget for layout pane to serve as parent for the view

```
    local class GRE_GROUP viewgp;
```

class for group to be shown in the view

```
    local class GRE_LAYER_RASTER tempLayer;
```

get handle for layout pane from dialog

```
    viewpane = dlgPreview.GetPaneByID("viewpane");
    viewpaneWidget = viewpane.GetWidget();
```

get parent Xm widget for pane

```
    viewpaneWidget.Resizable = 1;
```

set parent widget to be resizable

```
    viewgp = GroupCreate();
```

create display group

add the Temp raster to the group

```
    tempLayer = GroupQuickAddRasterVar(viewgp, Temp);
    tempLayer.DataTip.Prefix = "Interval: ";
```

create 2D view of group in dialog

```
    view = viewgp.CreateView(viewpaneWidget,"",450, 350, "NoCloseOption");

    view.ScalePosVisible = 0;
```

hide scale/position report

```
    }
```

Function to return the class interval of the current raster value by comparison to range boundaries.

```
func SetInterval(valueIn) {
    for i = 1 to numInterval {
        if (valueIn >= minRange[i] && valueIn <= maxRange[i]) then
        return i;
        }
    }
```