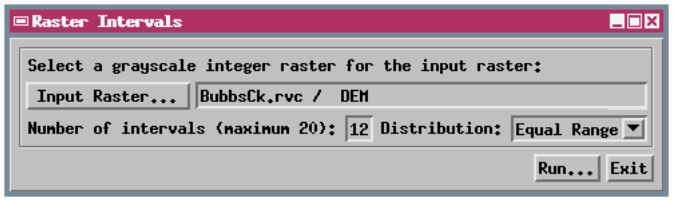
サンプルスクリプト

v72 新機能

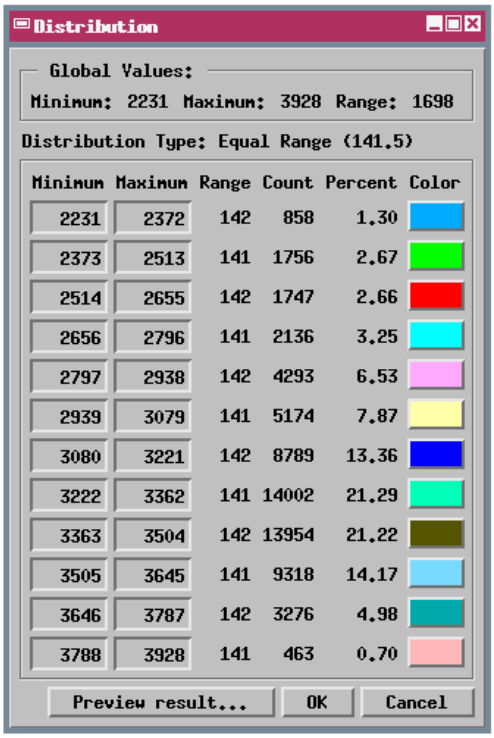
ラスタのテーママッピング

このカラープレートの裏面に抜粋を掲載した「Raster Intervals」スクリプトは、対話的なコントロールダイアログを持つスタンドアロンの処理スクリプトの例です。このスクリプトは、グレースケールラスタのセルを、ユーザが指定するグレースケール値の区間に分類し、区間ごとに単一の値を持つラスタを出力します。その際、カラーパレットも作られます。この手順は TNT 製品において、ベクタオブジェクトの要素の表示スタイルを、数値属性の値に基づいて設定する「テーママッピング」の方法に似ています。このスクリプトでは、区間の数 (最大 20 まで) や配分のタイプ (個数均等や区間均等) をユーザが指定できます。この Raster Intervals スクリプトでは、標高や傾斜角度、植生指数など、ある範囲の値を持つグレースケール整数値のラスタの分類にも使えます。

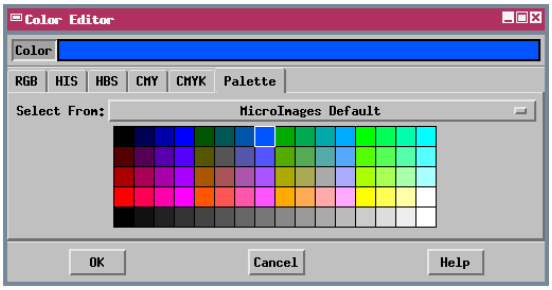
Raster Intervals サンプルスクリプトのメインコントロールダイアログウインドゥ。入力ラスタオブジェクトの選択、ラスタ値を区分する区間数の設定や配分タイプ (Equal Range: 等範囲か Equal Count: 等個数) を選択します。



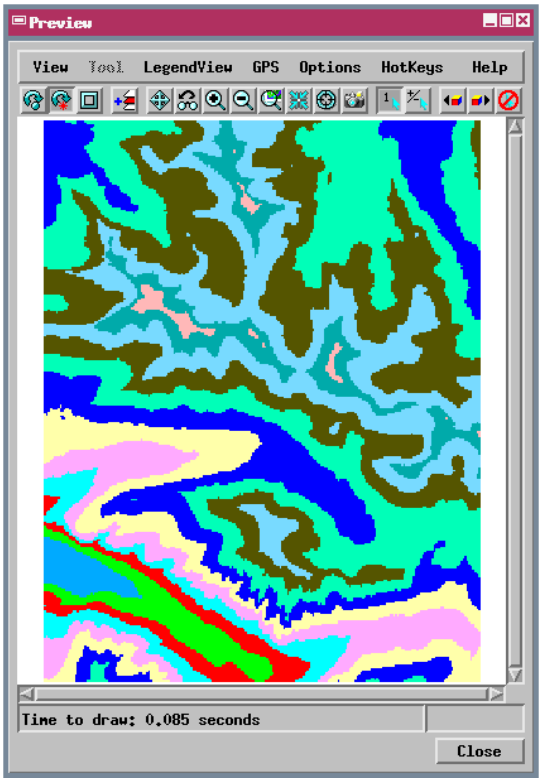
〈Raster Intervals〉ウィンドウの [Run(実行)] ボタンを押すと、〈Distribution(配分)〉ウィンドウが開きます。スクリプトにより初期設定された範囲と、それらの統計情報が一覧化されます。このダイアログウィンドウを使って、最大値や最小値を編集して初期設定の範囲を変えることができます；変更に応じて隣り合う範囲が自動調整されます。



このスクリプトは、出力ラスタオブジェクト用にカラーパレットを作ります。各ラスタ区間にデフォルトカラーが自動で割り当てられ、〈Distribution〉ウィンドウのカラーボタンに表示されます。カラーボタンを押すと、〈カラーエディタ〉ウィンドウが開き、区間の色を変更できます。



〈Distribution〉ウィンドウの [Preview Result(結果のプレビュー)] ボタンを押すと、〈プレビュー〉ウィンドウが開き、現在の範囲やカラー設定に基づいた一時的な出力ラスタが表示されます。



〈Distribution〉ウィンドウの [OK] ボタンを押すと、出力ラスタの保存先のプロンプトが出て、出力ラスタ作成の処理に進みます。

Raster Intervals スクリプトは、3 つの専用ウィンドウにより、(1) 入力ラスタの選択と処理パラメータの初期設定、(2) 範囲の値やカラーパレットの表示や修正、(3) 結果のラスタのプレビューを行います。Raster Intervals、Distribution、Preview の各ウィンドウは、各種コントロールやそれらのレイアウトを定義する XML によるダイアログ記述を使って設定されています。本スクリプトは、多様なダイアログコントロールに対するコールバック動作の設定方法、スクリプトによって作成された空間オブジェクトを表示するダイアログの設定や使用方法、相互に関連するスクリプトダイアログの作成方法について、実際の例を提供しています。さらに、デフォルトのカラーパレットは、XML の文字列に格納されており、スクリプトはメモリ中で XML 文書の解析を行います。従って、このスクリプトは、XML 構造 (この場合、赤、緑、青のカラー値) 中のデータにアクセスする場合に使われるメソッドの例を提供しています。

本機能は、v74 からメニューに組み込まれました。
「ラスタのテーママッピング (コマンドメニュー)」
http://www.opengis.co.jp/techguidej/74ThemeMappingaRaster_J.pdf

スクリプトやクエリで TNT 製品のスクリプト言語の各機能をどのように利用したらよいかを紹介するため、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/downloads/scripts.htm からダウンロードできます。

ラスタの主題図作成スクリプトから抜粋 (RasterIntervals.sml)

区間のセルの個数や割合を更新するためのプロシージャ

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

区間のセル数を 0 にリセット

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

指定の各区間に対してセルの個数を求めるため、全入力ラスタにわたってループ

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

区間毎のセルの割合を計算

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

カラーパレットを持つ区間ラスタを作るプロシージャ。一時的なプレビュー用ラスタや出力ラスタで使用

```
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");
```

Distribution ダイアログのカラーを基に出力ラスタのカラーパレットを作成

```
   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);
   }
```

Distribution ダイアログが開く時のコールバックのプロシージャ。開く前にカラーボタンの初期カラーを設定します (XML 構造を使ってこの処理を行う方法は無いため、カラーボタン用の GUI コントロールクラスに直接アクセスしなければいけません)。このダイアログは " モーダル (modal)" ですので、Domodal メソッドが呼び出されるまで作成されず、ボタンはダイアログの OnOpen コールバックを使って設定する必要があります。

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

色を割り当てるため、全区間にわたってループ

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

XML 構造から COLOR クラスインスタンスへ区間の赤、緑、青の値を割り当てます。

```
      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");
```

ダイアログ中の関連したカラーボタンコントロール用のハンドルを入手

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

Preview ダイアログを初期化するとき呼び出されるプロシージャ。ダイアログ中で表示の作成や一時ラスタの表示グループを作る時に使われます。

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

表示ウィンドウを含むレイアウトペインに対するクラスインスタンス

```
   local class widget viewpaneWidget;
```

レイアウトペインに対するウィジェットクラス。表示ウィンドウの親として振舞う。

```
   local class GRE_GROUP viewgp;
```

表示ウィンドウに表示するグループのクラス

```
   local class GRE_LAYER_RASTER tempLayer;
```

ダイアログからレイアウトペインへのハンドルを入手

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

ペインに対する親の Xm ウィジェットを入手

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

サイズ変更できるように親ウィジェットを設定

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

表示グループの作成

一時ラスタをグループに追加

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

ダイアログ中に 2 次元表示グループを作成

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

   view.ScalePosVisible = 0;
```

スケールと位置のレポートを隠します。

```
   }
```

範囲の境界と比較して現在のラスタ値のクラス区間を返す関数

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