Raster Data Operations in DotSpatial

**Tutorial (3)**

Purpose of this tutorial: Become familiar with the following raster operations in DotSpatial:

Implementing the hill shade, changing the color of the raster data, multiplying the raster data, reclassifying the raster data, using the mouse click event to get point values on the raster data layer.

This tutorial has 5 important steps.

***Step 1****: Download the DotSpatial class library*

***Step 2****: Add the DotSpatial reference and change the compile option.*

***Step 3:*** *Add the DotSpatial Controls into the Visual Studio Toolbox.*

***Step 4****: Design the GUI. (GUI - Graphical User Interface)*

***Step 5****: Write the code for implementing the map operations.*

This tutorial will not go into detail on how to add support for using additional formats through GDAL. If you need to know how to do this please see tutorial nine or the Developer Getting Started Guide.

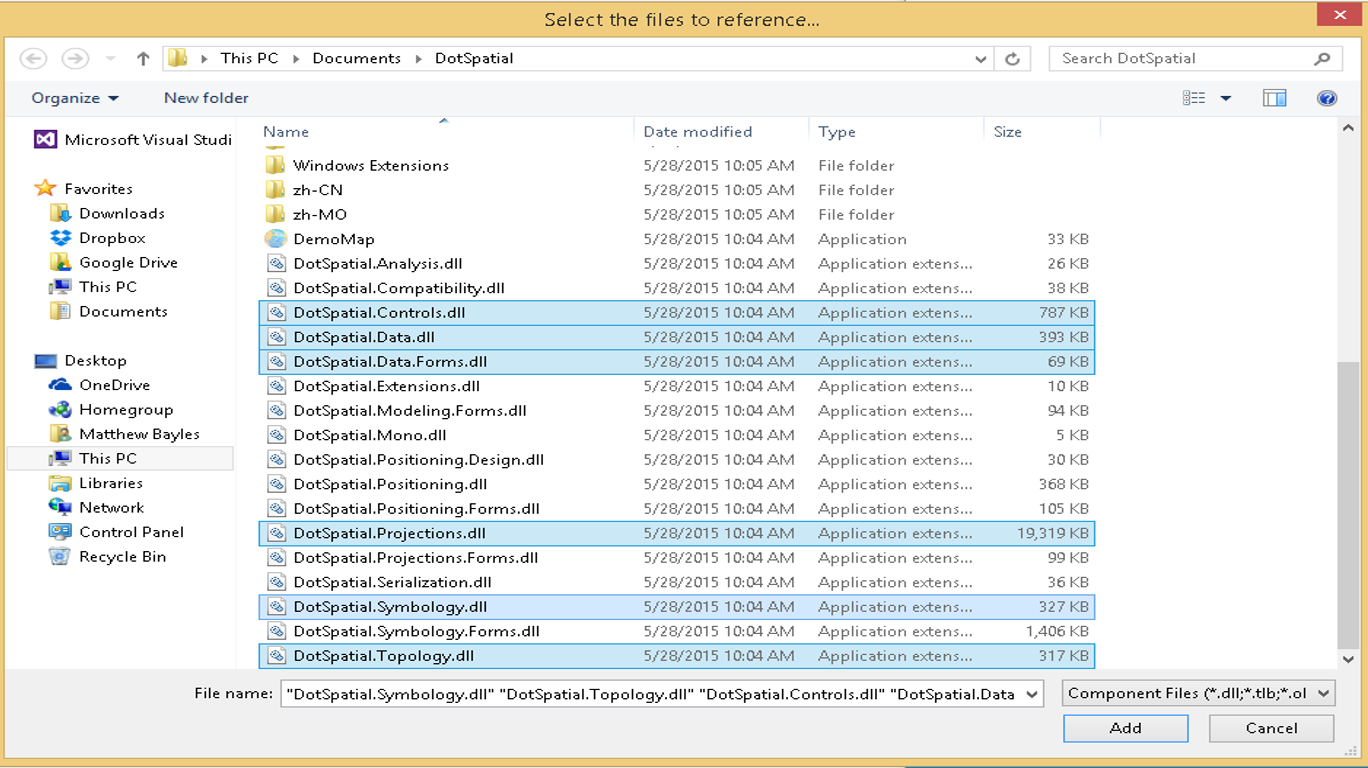
**Step 1**: Download the DotSpatial class library

This step is the same as Tutorial #1 step 1.

**Step 2:** Add the DotSpatial reference

1.1) Adding the references.

DotSpatial.Data.Forms.dll, DotSpatial.Symbology.dll, DotSpatial.Controls.dll, DotSpatial.Projections.dll, DotSpatial.Data.dll, DotSpatial.Topology.dll



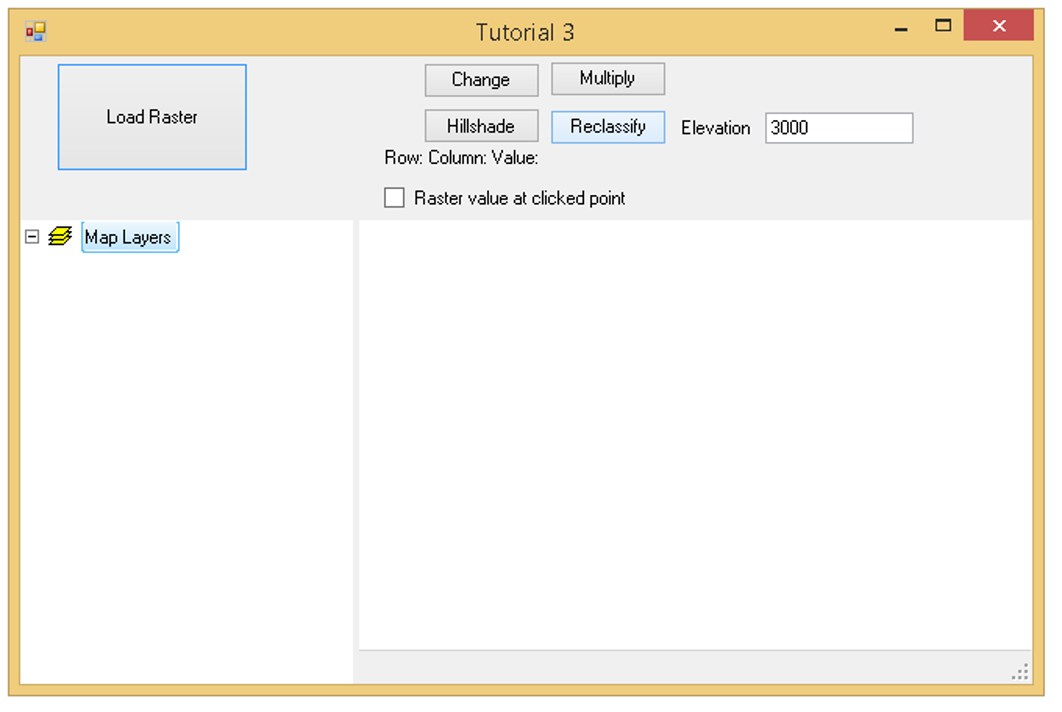
**Figure 1: Required References**

***Step 3:*** *Add the DotSpatial Controls into the Visual Studio Toolbox.*

This step is the same as the Tutorial # 1 step 3.

**Step 4**: Design the GUI

Design the GUI as follows:



**Figure 2: Final GUI**

Interface design considerations.

1. Add a panel control and the SpatialDockManger. The control's properties should be as follows:

|  |  |  |
| --- | --- | --- |
| Properties | Panel 1 | DockManger |
| Name | pnlOperations | sdmLegendMap |
| Dock | Top | FIll |

2. Add five buttons. Button properties should be as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Properties | Button1 | Button2 | Button3 | Button4 | Button5 |
| Name | btnLoadRaster | btnHillshade | btnChangeColor | btnMultiplyRaster | btnReclassify |
| Text | &Load Raster | &Hillshade | Change | &Multiply | &Reclassify |

3. Add two label controls. Labels' properties should be as follows:

|  |  |  |
| --- | --- | --- |
| Properties | Label1 | Label2 |
| Name | lblElevation | lblRasterValue |
| Text | Elevation | Row: Column: Value: |

4. Add a checkbox and it should have the following properties.

Name: chbRasterValue Text: Raster value at clicked point

5. Add a textbox and it should have the following features.

Name: txtElevation Text: 3000

6. Drag a “Legend” control from the DotSpatial tab under toolbox and drop it on left side of the SpatialDockManger. Legend properties should be as follows:

Name: Legend1 Dock: Fill

7. Drag a “Map” control from the DotSpatial tab under toolbox and drop it on the right side of the SpatialDockManger. Map properties should be as follows:

Name: map1 Dock: Fill Legend: Legend1

**Step: 5** Loading raster data.

6.1) Import the following namespaces in the coding window.

//Required namespaces

using DotSpatial.Symbology;

using DotSpatial.Controls;

using DotSpatial.Data;

using DotSpatial.Topology;

6.2 ) btnLoadRaster click event should be as follows:

private void btnLoadRaster\_Click(object sender, EventArgs e)

{

//AddRasterLayer() method is used to load the raster layers on the map

map1.AddLayer();

map1.ZoomToMaxExtent();

}

**Implementing the hill shade.**

btnHillshade\_Click event should have the following code.

private void btnHillshade\_Click(object sender, EventArgs e)

{

if (map1.Layers.Count > 0)

{

//IMapRasterLayer layer = (IMapRasterLayer)map1.Layers[0];

IMapRasterLayer layer =map1.Layers[0] as IMapRasterLayer;

if (layer == null)

{

MessageBox.Show("Please select a raster layer");

return;

}

layer.Symbolizer.ShadedRelief.ElevationFactor = 1;

layer.Symbolizer.ShadedRelief.IsUsed = true;

layer.WriteBitmap();

}

else

{

MessageBox.Show("Please add a layer to the map.");

}

}

**Change the color.**

btnChangeColor click event should have the following event.

private void btnChangeColor\_Click(object sender, EventArgs e)

{

if (map1.Layers.Count > 0)

{

//change the color of raster

//typecast the first layer to MapRasterLayer

IMapRasterLayer layer = map1.Layers[0] as IMapRasterLayer;

if (layer == null)

{

MessageBox.Show("Please add a raster layer.");

return;

}

//set the color scheme

//create an instance for a colorscheme

ColorScheme scheme = new ColorScheme();

//create a new category

ColorCategory category1 = new ColorCategory(2500, 3000, Color.Red, Color.Yellow);

category1.LegendText = "Elevation 2500 - 3000";

//add the category to the color scheme

scheme.AddCategory(category1);

//create another category

ColorCategory category2 = new ColorCategory(1000, 2500, Color.Blue, Color.Green);

category2.LegendText = "Elevation 1000 - 2500";

scheme.AddCategory(category2);

//assign new color scheme

layer.Symbolizer.Scheme = scheme;

//refresh the layer display in the map

layer.WriteBitmap();

}

else

{

MessageBox.Show("Please add a layer to the map.");

}

}

**Multiply the raster.**

btnMultiplyRaster should have the following event.

private void btnMultiplyRaster\_Click(object sender, EventArgs e)

{

if (map1.Layers.Count > 0)

{

IMapRasterLayer layer = map1.Layers[0] as IMapRasterLayer ;

if (layer == null)

{

MessageBox.Show("Please select a raster layer");

}

IRaster demRaster = layer.DataSet;

string[] rasterOptions = new string[1];

IRaster newRaster = Raster.CreateRaster("multiply.bgd", null, demRaster.NumColumns, demRaster.NumRows, 1, demRaster.DataType, rasterOptions);

//Bounds specify the cellsize and the coordinates of raster corner

newRaster.Bounds = demRaster.Bounds.Copy();

newRaster.NoDataValue = demRaster.NoDataValue;

newRaster.Projection = demRaster.Projection;

for (int i = 0; i <= demRaster.NumRows - 1; i++)

{

for (int j = 0; j <= demRaster.NumColumns - 1; j++)

{

if (demRaster.Value[i, j] != demRaster.NoDataValue)

{

newRaster.Value[i, j] = demRaster.Value[i, j] \* 2;

}

}

}

//save the new raster to the file

newRaster.Save();

//add the new raster to the map

map1.Layers.Add(newRaster);

}

else

{

MessageBox.Show("Please add a layer to the map.");

}

}

**Reclassify the raster.**

btnReclassify should have the following event.

private void btnReclassify\_Click(object sender, EventArgs e)

{

//typecast the selected layer to IMapRasterLayer

IMapRasterLayer layer = map1.Layers.SelectedLayer as IMapRasterLayer ;

if (layer == null)

{

MessageBox.Show("Please select a raster layer.");

}

else

{

//get the raster dataset

IRaster demRaster = layer.DataSet;

//create a new empty raster with same dimension as original raster

string[] rasterOptions = new string[1];

IRaster newRaster = Raster.CreateRaster("reclassify.bgd", null, demRaster.NumColumns, demRaster.NumRows, 1, demRaster.DataType, rasterOptions);

newRaster.Bounds = demRaster.Bounds.Copy();

newRaster.NoDataValue = demRaster.NoDataValue;

newRaster.Projection = demRaster.Projection;

//reclassify raster.

// values >= specified value will have new value 1

// values < specified value will have new value 0

double oldValue = 0;

//get the specified value from the textbox

double specifiedValue = Convert.ToDouble(txtElevation.Text);

for (int i = 0; i <= demRaster.NumRows - 1; i++)

{

for (int j = 0; j <= demRaster.NumColumns - 1; j++)

{

//get the value of original raster

oldValue = demRaster.Value[i, j];

if (oldValue >= specifiedValue)

{

newRaster.Value[i, j] = 1;

}

else

{

newRaster.Value[i, j] = 0;

}

}

}

newRaster.Save();

map1.Layers.Add(newRaster);

}

}

**Get the mouse clicked point values on the raster data layer.**

Select the chbRasterValue's "CheckedChanged" event. chbRasterValue\_CheckedChanged event should have the following events.

private void chbRasterValue\_CheckedChanged(object sender, EventArgs e)

{

if (chbRasterValue.Checked)

{

IMapRasterLayer rasterLayer = map1.Layers.SelectedLayer as IMapRasterLayer ;

if ((rasterLayer != null))

{

//set the map cursor to cross

map1.Cursor = Cursors.Cross;

}

else

{

//if no raster layer is selected, uncheck the checkbox

MessageBox.Show("Please select a raster layer.");

chbRasterValue.Checked = false;

}

}

else

{

//change map cursor back to arrow

map1.Cursor = Cursors.Arrow;

}

}

Select the Map1's mouse up event.

map1\_MouseUp event should have the following code.

private void map1\_MouseUp(object sender, MouseEventArgs e)

{

if (chbRasterValue.Checked)

{

//get the layer selected in the legend

IMapRasterLayer rasterLayer = map1.Layers.SelectedLayer as IMapRasterLayer;

if ((rasterLayer != null))

{

//get the raster data object

IRaster raster = rasterLayer.DataSet;

//convert mouse position to map coordinate

Coordinate coord = map1.PixelToProj(e.Location);

//convert map coordinate to raster row and column

RcIndex rc = raster.Bounds.ProjToCell(coord);

int row = rc.Row;

int column = rc.Column;

//check if clicked point is inside of raster

if ((column > 0 & column < raster.NumColumns & row > 0 & row < raster.NumRows))

{

//get the raster value at row and column

double value = raster.Value[row, column];

//show the row, column and value in the label

lblRasterValue.Text = string.Format("row: {0} column: {1} value: {2}", row, column, value);

}

else

{

lblRasterValue.Text = "outside of raster";

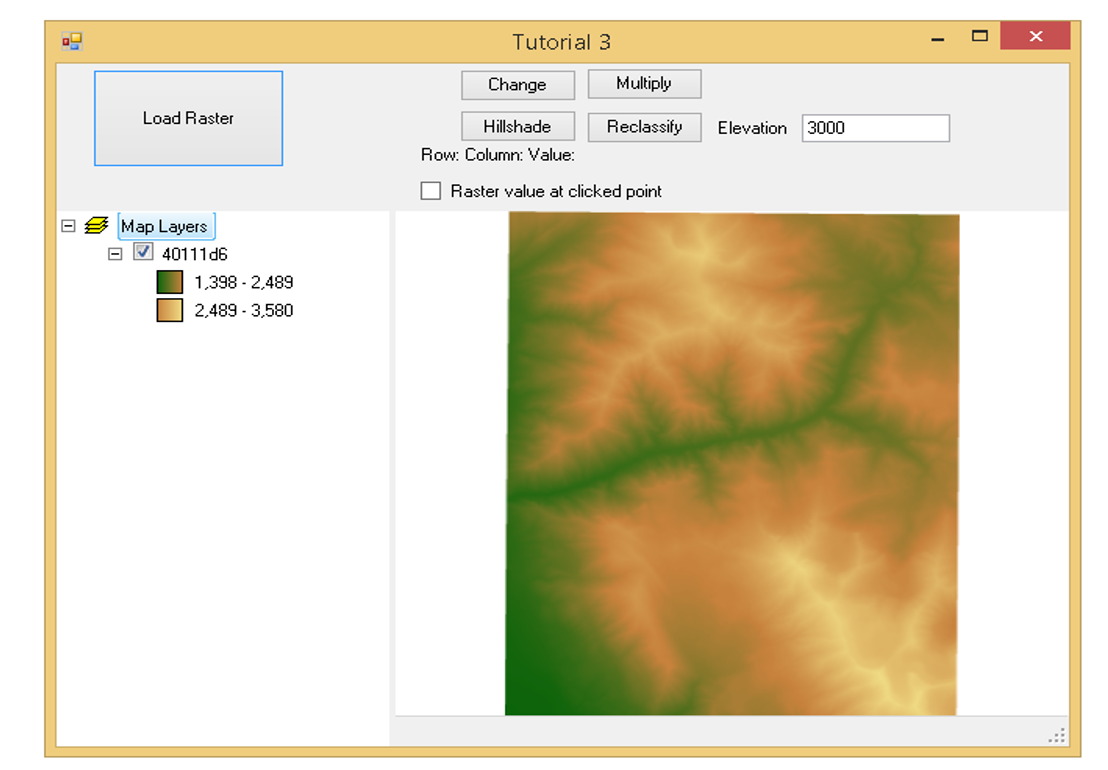
}

}

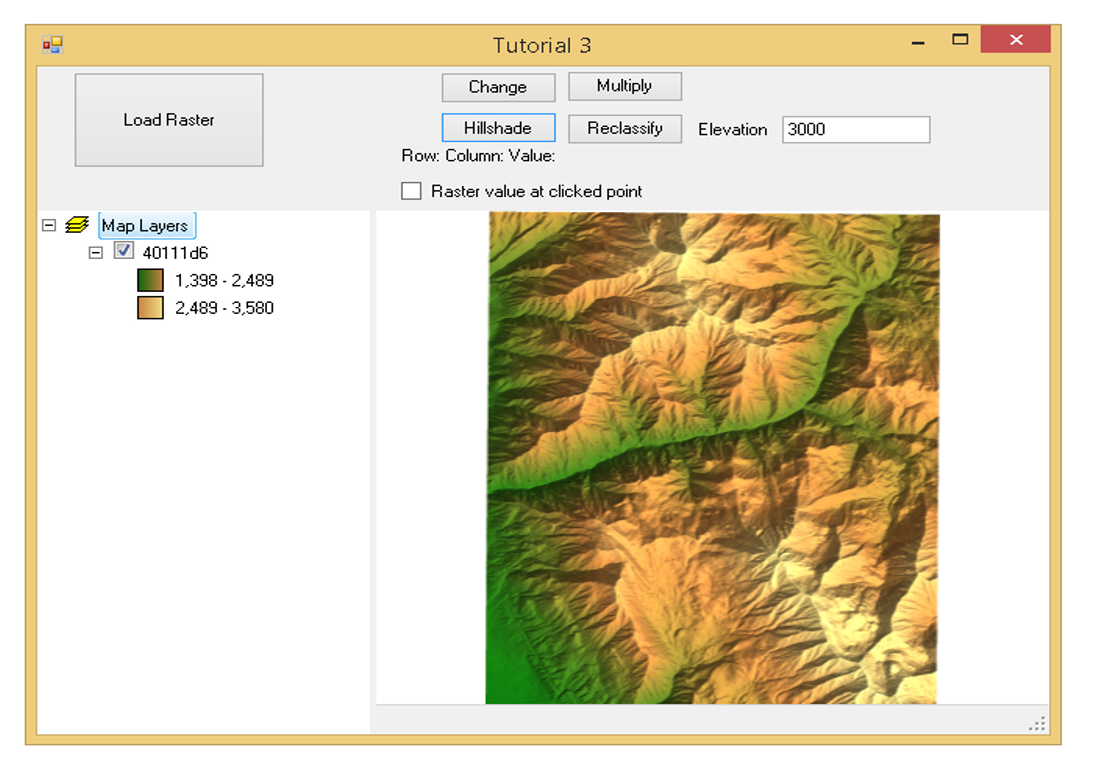
}

}

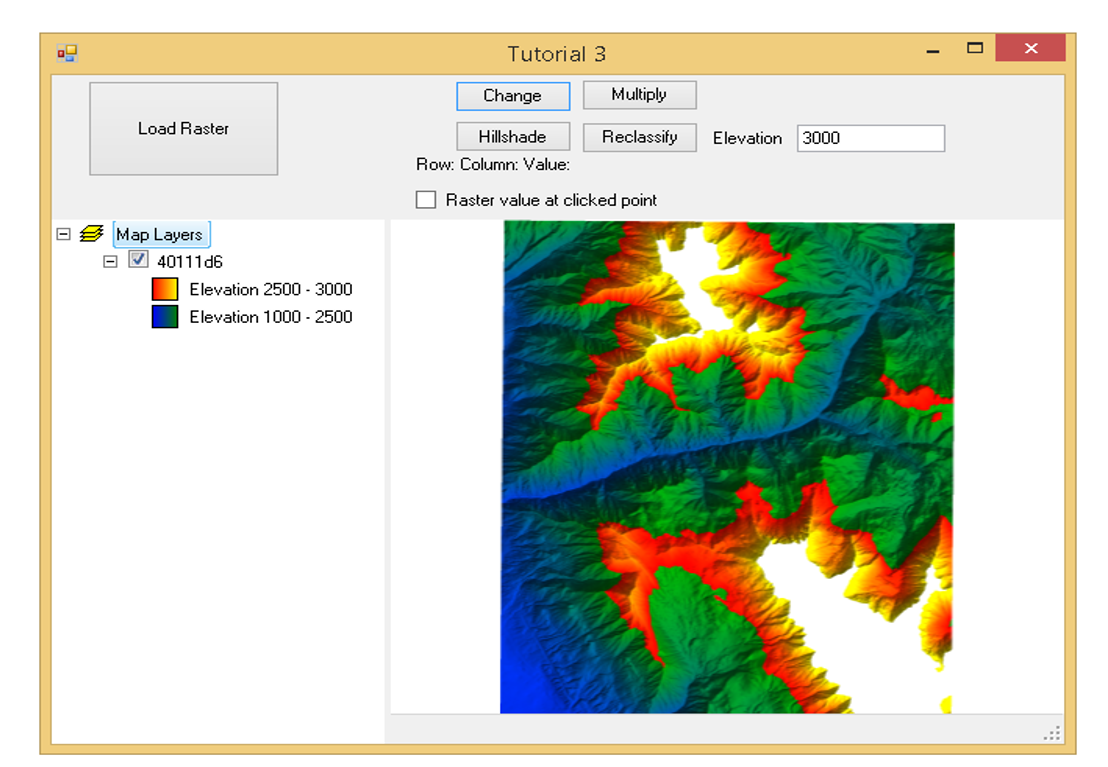
***Output screen shot for different operations.***



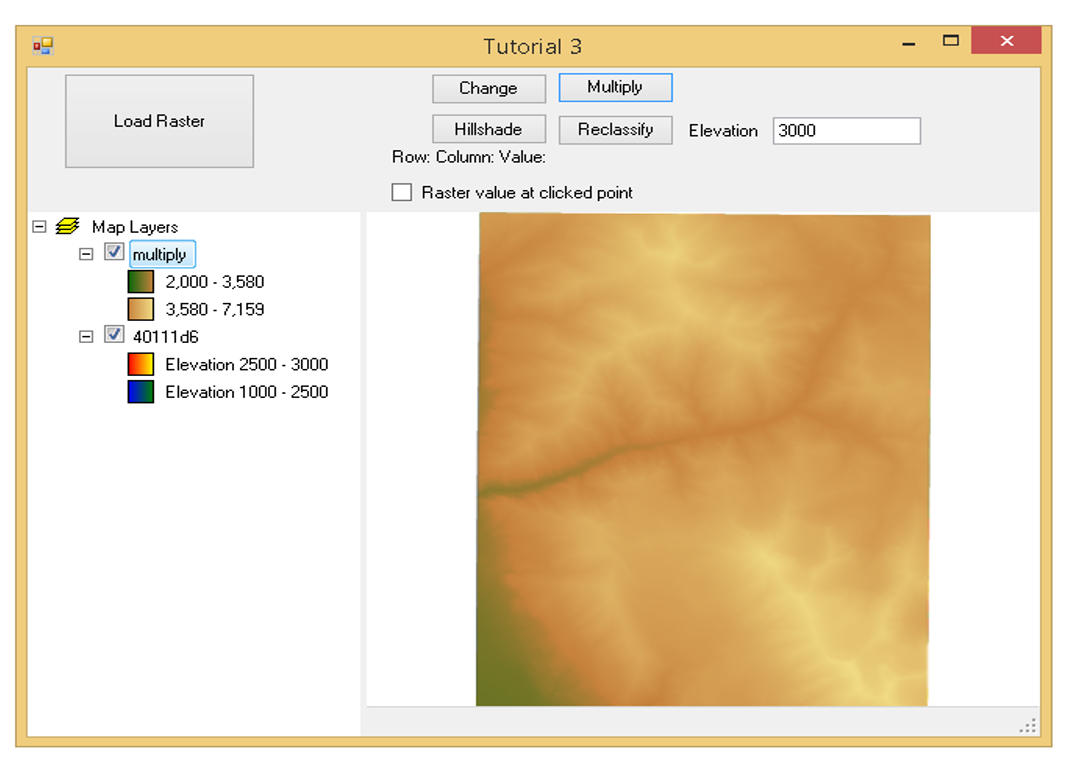
**Figure 3: Loading Raster Data**



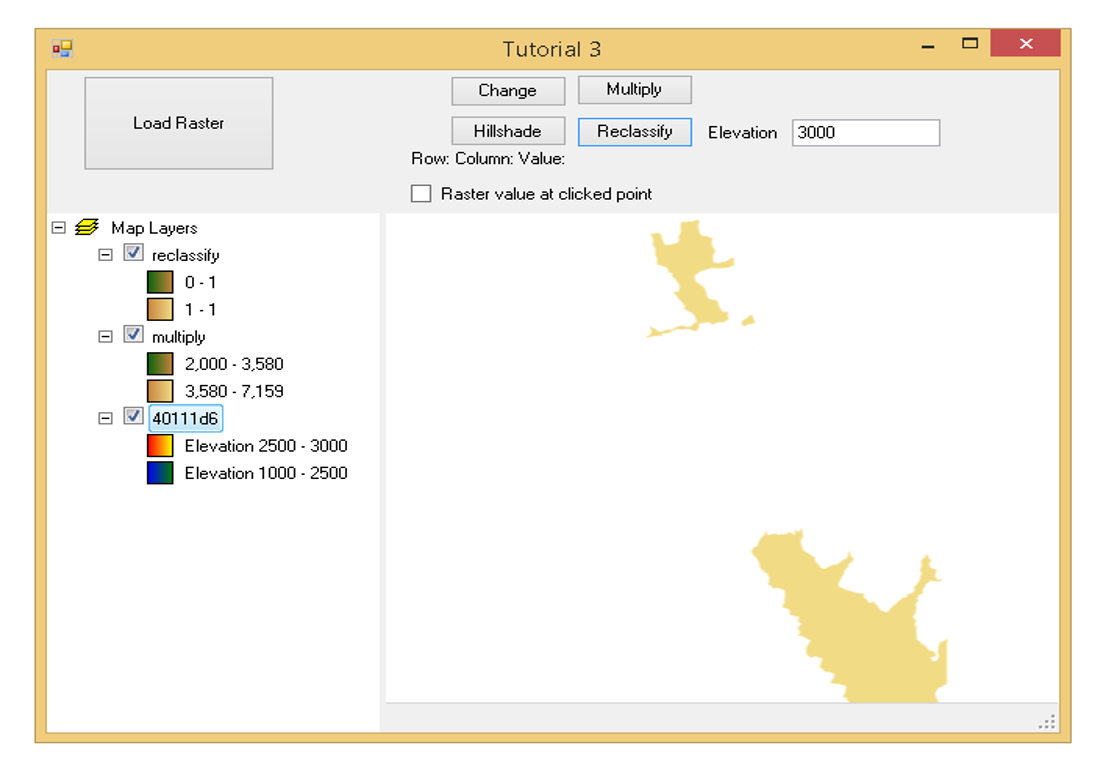
**Figure 4: Hill Shade Operation**



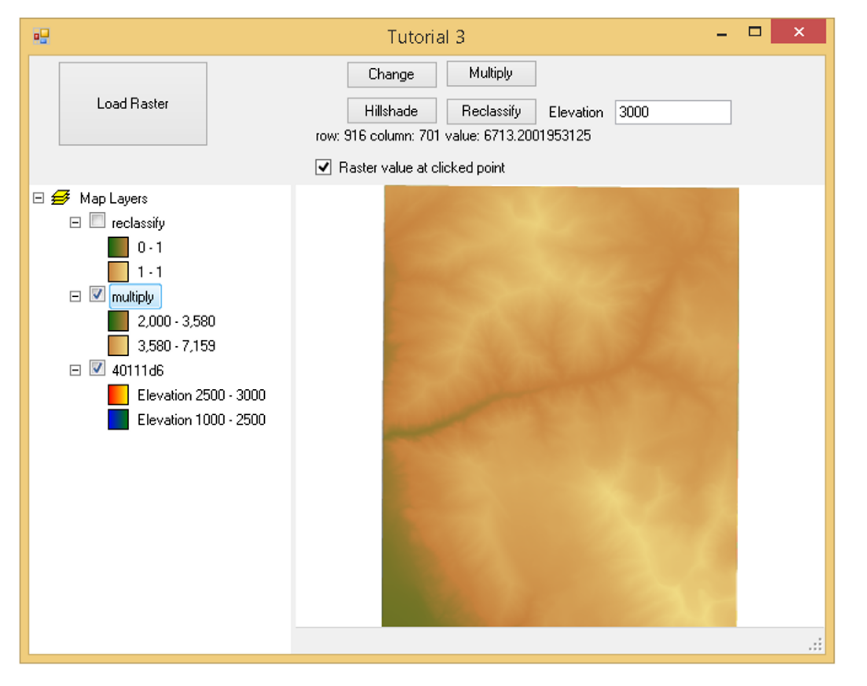
**Figure 5: Color Change Operation**



**Figure 6: Raster Multiplication**



**Figure 7: After Reclassify Operation**



**Figure 8: Getting Point Values on Raster**