Working with DotSpatial Symbology and Attribute Table

**Tutorial (2)**

Purpose of this tutorial: Become familiar with DotSpatial symbology, attribute table filter operations.

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

This step is the same as the first step in the first tutorial.

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

Add the required DotSpatial reference in the visual studio development environment.

Create a new C# application and right click over the project on the solution explorer. On the context menu select the add reference and add the following reference from the DotSpatial folder:

DotSpatial.Controls, DotSpatial.Data, DotSpatial.Data.Forms, DotSpatial.Serialization, DotSpatial.Symbology, DotSpatial.Topology.

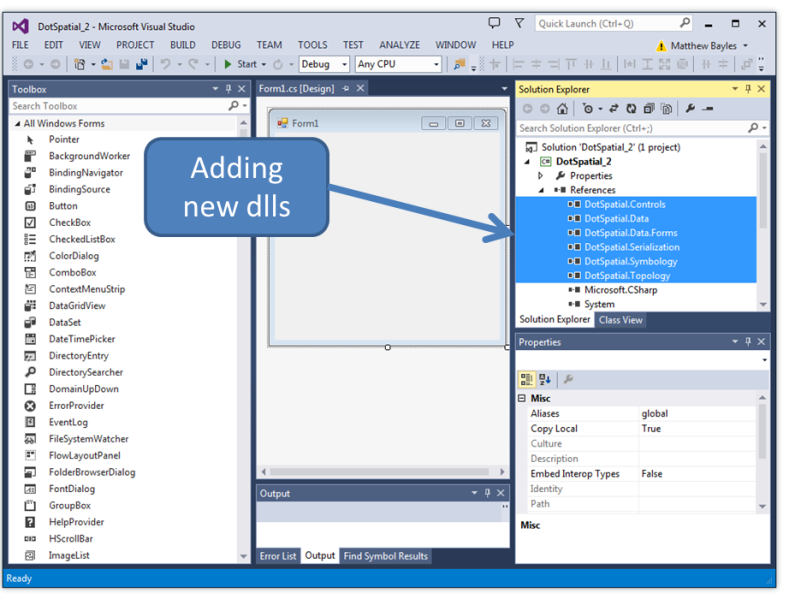


Figure 1: Required dlls

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

This step is the same as in the first tutorial.

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

Design the GUI as follows:

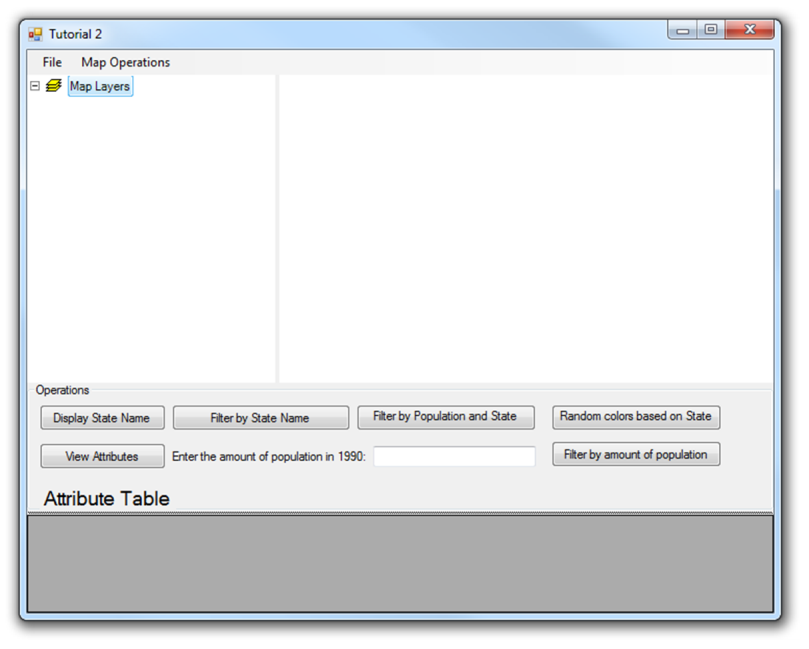


Figure 2: User Interface

Interface design considerations.

1. Add a MenuStrip control on the form. The menu should contain the following items.

Main menu items: File, MapOperation

Sub menu items:

Under File menu item create 3 sub menu items as follows:

Load Map - Shortcut key should be "Control key + L"

Clear Map - Shortcut key should be "Control key + C"

Exit - Shortcut key should be "Control key + E"

Under Map Operations menu items create 3 sub menu items as follows:

Zoom In - Shortcut key should be "Control key + Up arrow"

Zoom out - Shortcut key should be "Control key + Down arrow"

Zoom Extent - Shortcut key should be "Control key + Z"

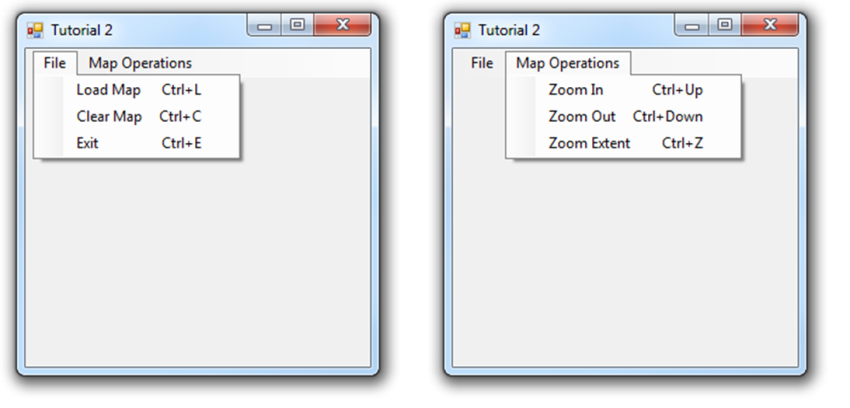


Figure 3: Menu

Note: For implementing shortcut key with alt and letters, you need to add the "&" sign in front of the letters. Example: File menu can access by pressing "Alt"+ F key. Adding "&" sign in front of letter F is used implement this feature.

2. Add the SpatialDockManger controls on the form.

Add a SpatialDockManger control on the form. Set the property of the SpatialDockManger control as follows:

Name: sdmMapLegend, Orientatio: Vertical, Dock : Top.

Add one more SpatialDockManger control in the same form on the empty space (below the above SpatialDockManger control. Set the property of the second SpatialDockManger control as follows:

Name: sdmDataOperation , Orientation : Horizontal , Dock : Fill

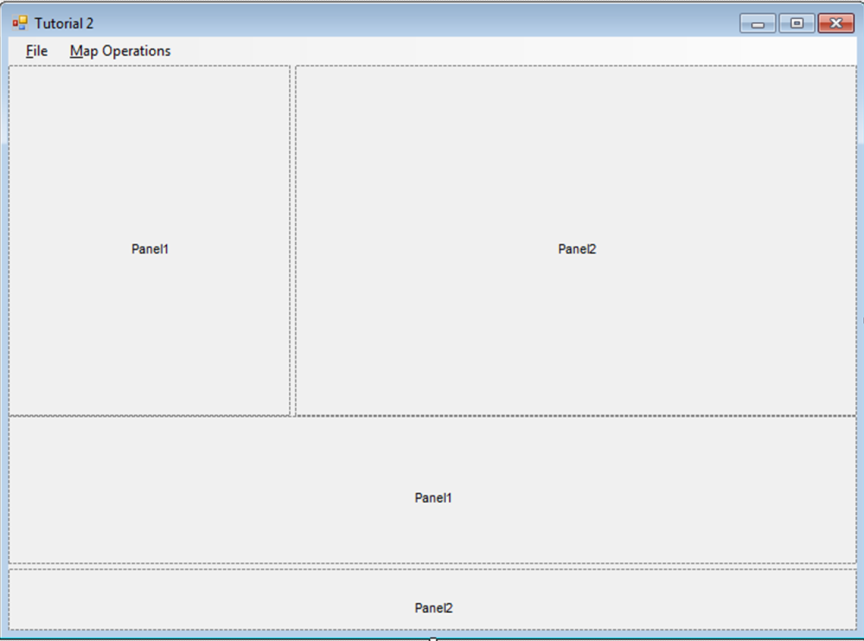


Figure 4: SpatialDockMangers

2. Add the Map and legend controls on the form.

Add the map control on the splcMapLegend control's right side and add legend control on the left side. The Map and Legend control are located on the Toolbox under DotSpatial controls.

Set the map properties as follows: Dock = Fill, Legend = Legend1, FunctionMode = Pan

Set the legend properties as follows: Dock = Fill

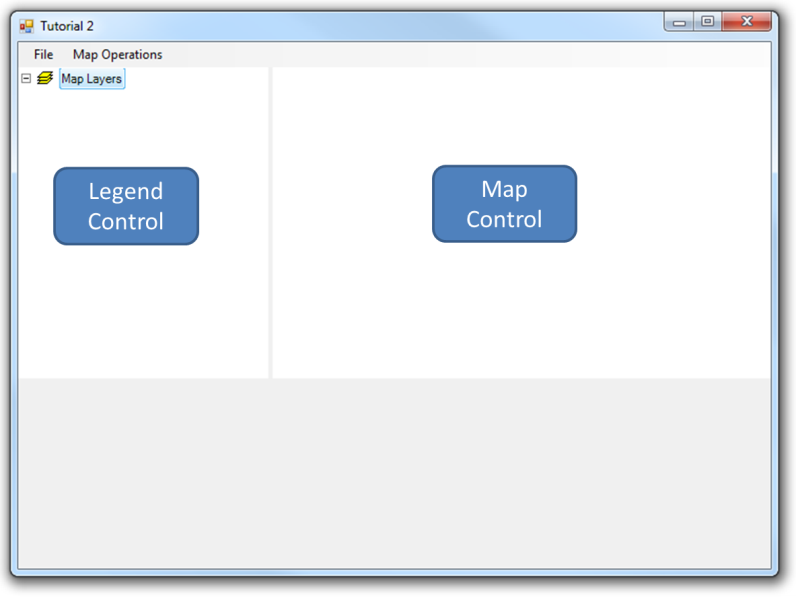


Figure 5: Map and Legend Controls

3. Add a DataGridView and button controls on the form.

Add a DataGridView on the form: DataGridView is located under Data tab of the toolbox window. Add a DataGridView on the splcDataOperation control's panel2 and set its property as follows:

Name = dgvAttributeTabl , Dock = fill

Add a GroupBox control on the splcDataOpeation control's panel1 and set its property as follows:

Name = gbOperations, Text = Operations, Dock = Fill

Add the buttons and other controls.

|  |  |  |
| --- | --- | --- |
| Controls | Name | Text |
| Button | btnDisplayStateName | &Display State Name |
| Button | btnFilterByStateName | Filter by &State Name |
| Button | btnFilterByPopState | Filter by &Population and State Name |
| Button | btnRandomColors | &Random colors based on State Name |
| Button | btnViewAttributes | View &Attributes |
| Button | btnFilterByPopulation | Filter by amount of &population |
| Label | lblPopulation | Enter the amount of population in 1990: |
| TextBox | txtPopulation |  |
| Label | lblTitle | Attribute Table |

Final GUI should look as follows:

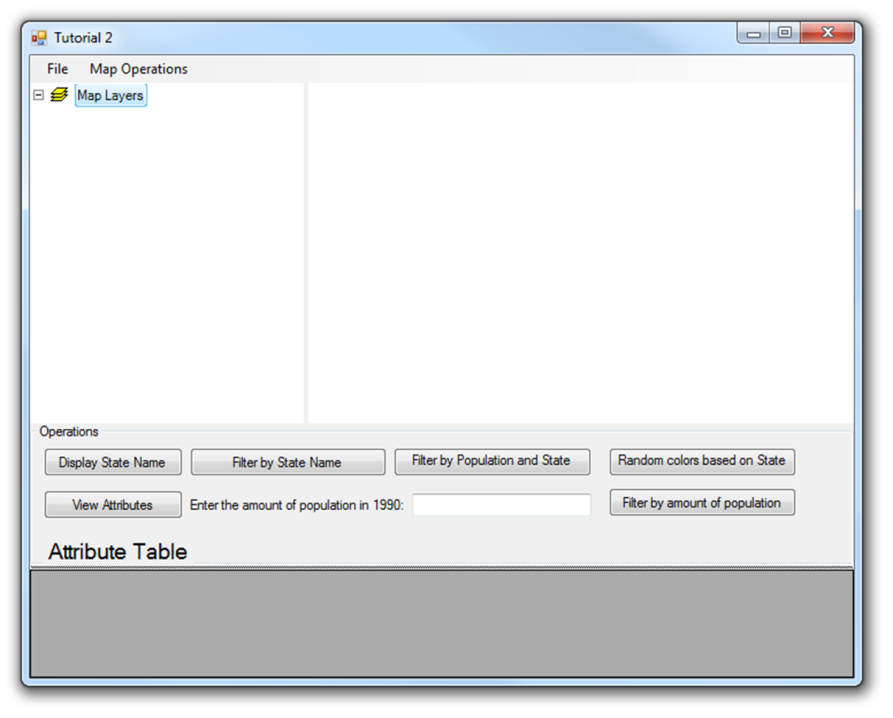


Figure 6: Final GUI

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

Add the following namespaces.

// Required namespaces

using DotSpatial.Controls;

using DotSpatial.Symbology;

Menu Item code implementation

Load a map

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

{

//AddLayer() method is used to add a shape file in the MapControl.

map1.AddLayer();

}

Clear the map

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

{

//Clear() method is used to clear the layers from the map control.

map1.Layers.Clear();

}

Exit the application

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

{

// Confirm with users that they are ready to close application or not with the help of message box.

if (MessageBox.Show("Do you want to close this application?", "Admin", MessageBoxButtons.OKCancel) == DialogResult.OK)

{

//Close() method is used to close the application.

this.Close();

}

}

Zoom in, Zoom Out, ZoomToMaxExtent

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

{

map1.ZoomIn();

}

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

{

map1.ZoomOut();

}

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

{

map1.ZoomToMaxExtent();

}

Implementing the attribute table filter operations and symbology

Displaying the attribute values on the shape file.

This example is used to display the state names on the shape file.

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

{

//Check the number of layers from MapControl

if (map1.Layers.Count > 0)

{

//Declare a MapPolygonLayer

MapPolygonLayer stateLayer = default(MapPolygonLayer);

//TypeCast the first layer from MapControl to MapPolygonLayer.

//Layers are 0 based, therefore 0 is going to grab the first layer from the MapControl

stateLayer = (MapPolygonLayer)map1.Layers[0];

//Check whether stateLayer is polygon layer or not

if (stateLayer == null)

{

MessageBox.Show("The layer is not a polygon layer.");

}

else

{

//add StateName as labels on the stateLayer

stateLayer.AddLabels("[STATE\_NAME]", new Font("Tahoma", (float)8.0), Color.Black);

}

}

else

{

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

}

}

Displaying a specific state on the Map based on "**SelectByAttribute**" method. In this example Idaho State will be highlighted on the map.

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

{

//Check the number of layers from MapControl

if (map1.Layers.Count > 0)

{

//Declare a MapPolygonLayer

MapPolygonLayer stateLayer = default(MapPolygonLayer);

//TypeCast the first layer from MapControl to MapPolygonLayer.

//Layers are 0 based, therefore 0 is going to grab the first layer from the MapControl

stateLayer = (MapPolygonLayer)map1.Layers[0];

//Check whether stateLayer is polygon layer or not

if (stateLayer == null)

{

MessageBox.Show("The layer is not a polygon layer.");

}

else

{

//SelectByAttribute method is used to implement the filter operations.

//In this example, STATE\_NAME is used implement the filter operation

//We can see the IdhoSate on the map

stateLayer.SelectByAttribute("[STATE\_NAME] = 'Idaho'");

}

}

else

{

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

}

}

Implementing a filter operation for more than one attribute from the attribute table. In this example, two attributes will be filtered and the result will be highlighted on the map.

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

{

//check the number of layers from map control

if (map1.Layers.Count > 0)

{

//Delacre a MapPolygonLayer

MapPolygonLayer stateLayer = default(MapPolygonLayer);

//Type cast the FirstLayer of MapControl to MapPolygonLayer

stateLayer = (MapPolygonLayer)map1.Layers[0];

//Check the MapPolygonLayer ( Make sure that it has a polygon layer)

if (stateLayer == null)

{

MessageBox.Show("The layer is not a polygon layer.");

}

else

{

//!!!-------------------- this line is necessary otherwise the code doesn't work------------------------!!!!!!!!!!!!!!!!!!!!

//this will load the attribute table of the layer into memory.

stateLayer.DataSet.FillAttributes();

//Create a new PolygonScheme

PolygonScheme scheme = new PolygonScheme();

//Create a new PolygonCategory

PolygonCategory category = new PolygonCategory(Color.Yellow, Color.Red, 1);

//Declare a filter string

//[POP1990],[STATE\_NAME] are attributes from the attribute table of the given shape file.

string filter = "[POP1990] > 10000000 OR [STATE\_NAME] = 'Idaho'";

//Set/Assign the filter expression to PolygonCategory

category.FilterExpression = filter;

//Set the Legend Text

category.LegendText = "population > 10 Million";

//Add the PolygonCategory in to the PolygonScheme

scheme.AddCategory(category);

//Set the scheme in to the MapPolygonLayer's symbology

stateLayer.Symbology = scheme;

}

}

else

{

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

}

}

This code will set the random colors based on the unique values from an attribute table's column.

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

{

//check the number of layers from map control

if (map1.Layers.Count > 0)

{

//Delacre a MapPolygonLayer

MapPolygonLayer stateLayer = default(MapPolygonLayer);

//Type cast the FirstLayer of MapControl to MapPolygonLayer

stateLayer = (MapPolygonLayer)map1.Layers[0];

//Check the MapPolygonLayer ( Make sure that it has a polygon layer)

if (stateLayer == null)

{

MessageBox.Show("The layer is not a polygon layer.");

}

else

{

//Create a new PolygonScheme

PolygonScheme scheme = new PolygonScheme();

//Set the ClassificationType for the PolygonScheme via EditorSettings

scheme.EditorSettings.ClassificationType = ClassificationType.UniqueValues;

//Set the UniqueValue field name

//Here STATE\_NAME would be the Unique value field

scheme.EditorSettings.FieldName = "STATE\_NAME";

//create categories on the scheme based on the attributes table and field name

//In this case field name is STATE\_NAME

scheme.CreateCategories(stateLayer.DataSet.DataTable);

//Set the scheme to stateLayer's symbology

stateLayer.Symbology = scheme;

}

}

else

{

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

}

}

This code will filter the data from shape file during the run time. The user can enter the filter amount of population via the interface.

/// <summary>

/// This method is used filter the attribute table of the shapefile based on the population in 1990.

/// </summary>

/// <param name="population">Amount of population in 1990</param>

/// <remarks>No return value</remarks>

private void filterbyPopulation(int population)

{

if (map1.Layers.Count > 0)

{

MapPolygonLayer stateLayer = default(MapPolygonLayer);

stateLayer = (MapPolygonLayer)map1.Layers[0];

if (stateLayer == null)

{

MessageBox.Show("The layer is not a polygon layer.");

}

else

{

//!!! this line is necessary otherwise the code doesn't work

//this will load the attribute table of the layer into memory.

stateLayer.DataSet.FillAttributes();

PolygonScheme scheme = new PolygonScheme();

PolygonCategory category = new PolygonCategory(Color.Yellow, Color.Red, 1);

string filter = "[POP1990] < " + population + "";

category.FilterExpression = filter;

category.LegendText = "population < " + population.ToString();

scheme.AddCategory(category);

stateLayer.Symbology = scheme;

}

}

else

{

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

}

}

Call the above method in the btnFilterByPopulation button click event.

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

{

double number;

//Validating the textbox input.

if (string.IsNullOrEmpty(txtPopulation.Text) || !double.TryParse(txtPopulation.Text, out number))

{

MessageBox.Show("Please enter a valid value", "Admin", MessageBoxButtons.OK, MessageBoxIcon.Warning);

txtPopulation.Text = "";

}

else

{

//Call the filterbyPopulation method.

//We need to pass an interger intput paramter,

// Therefore, I just implemented integer typecasting.

filterbyPopulation(Convert.ToInt32(txtPopulation.Text.ToString()));

}

}

Displaying the attribute table on a DataGridView.

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

{

//Declare a datatable

DataTable dt = null;

if (map1.Layers.Count > 0)

{

MapPolygonLayer stateLayer = default(MapPolygonLayer);

stateLayer = (MapPolygonLayer)map1.Layers[0];

if (stateLayer == null)

{

MessageBox.Show("The layer is not a polygon layer.");

}

else

{

//Get the shapefile's attribute table to our datatable dt

dt = stateLayer.DataSet.DataTable;

//Set the datagridview datasource from datatable dt

dgvAttributeTable.DataSource = dt;

}

}

else

{

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

}

}

Implementing DataGridView selection event

Based on the selection of the state from the DataGridView, the state is highlighted on the map. First of all, select the DataGridView on the GUI and select its property window. On the property window select the event tab. On the event window, double click over the SelectionChanged event.

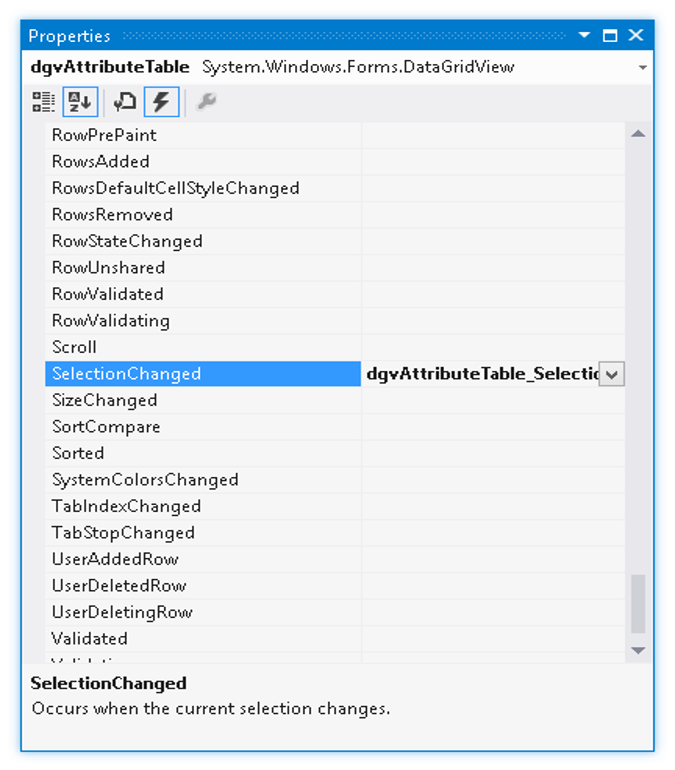


Figure 7: SelectionChanged

Under the dgvAttributeTable\_SelectionChanged write the code as follows:

private void dgvAttributeTable\_SelectionChanged(object sender, EventArgs e)

{

foreach (DataGridViewRow row in dgvAttributeTable.SelectedRows)

{

MapPolygonLayer stateLayer = default(MapPolygonLayer);

stateLayer = (MapPolygonLayer)map1.Layers[0];

if (stateLayer == null)

{ MessageBox.Show("The layer is not a polygon layer."); }

else

{

stateLayer.SelectByAttribute("[STATE\_NAME] =" + "'" + row.Cells["STATE\_NAME"].Value + "'");

}

}

Final output will be as follows:

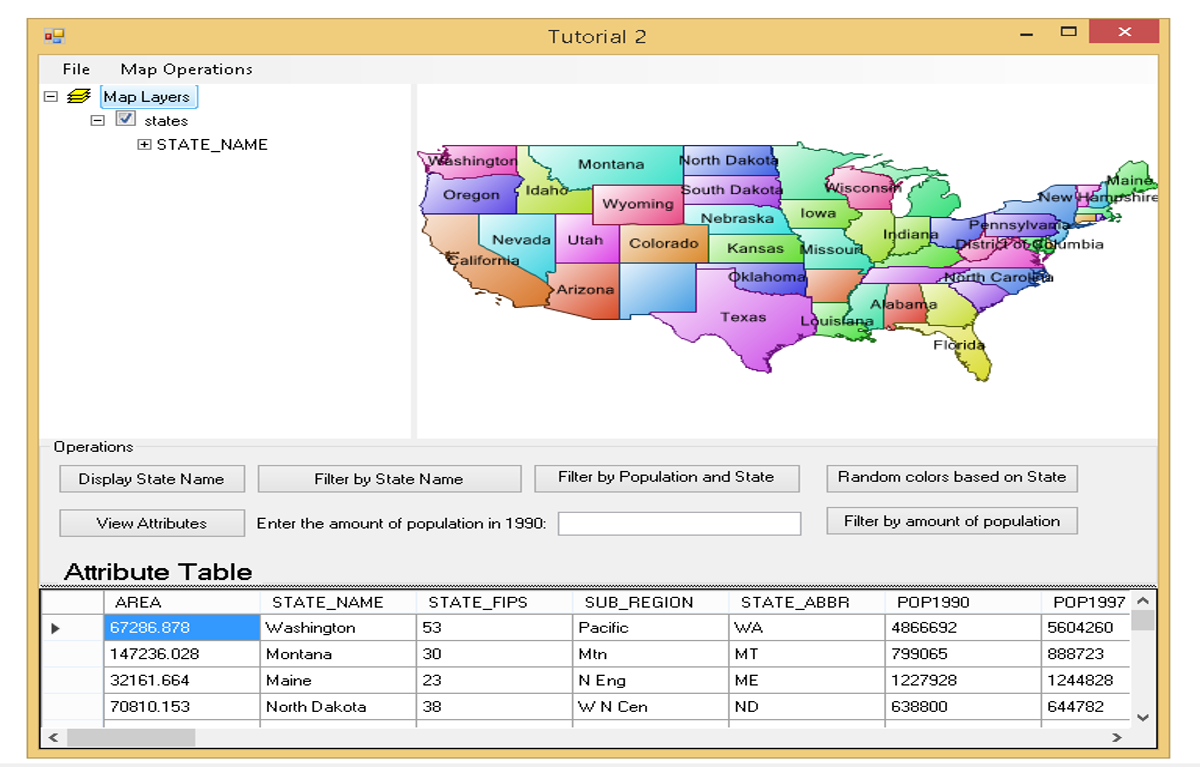


Figure 8: Final Output