Shapefiles, Attribute Table Operations and Printing map in DotSpatial

**Tutorial (4)**

Purpose of this tutorial: Become familiar with shapefiles, the attribute table, and printing operations in DotSpatial.

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

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

**Step 2:** Add the DotSpatial reference

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

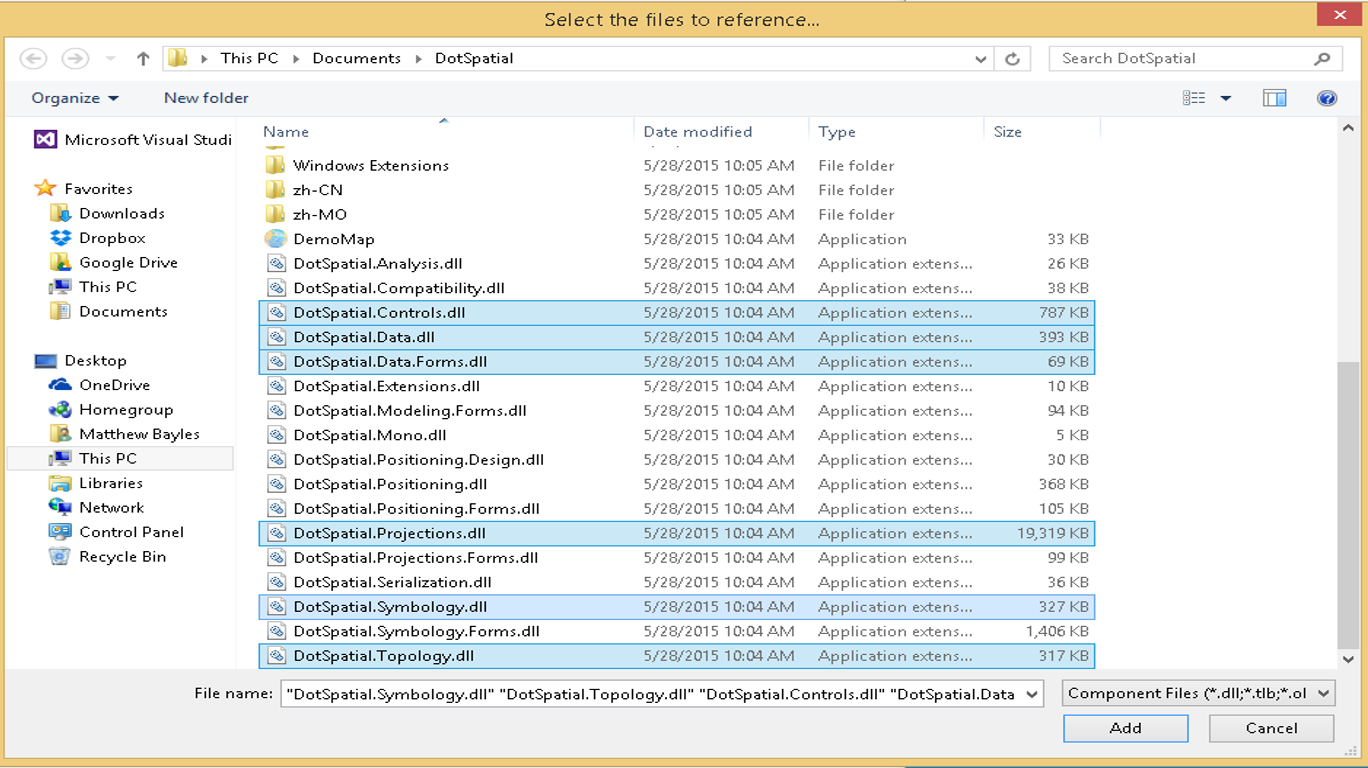


Figure 1: Required References

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

This step is the same as in the first Tutorial.

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

Design the GUI as follows:

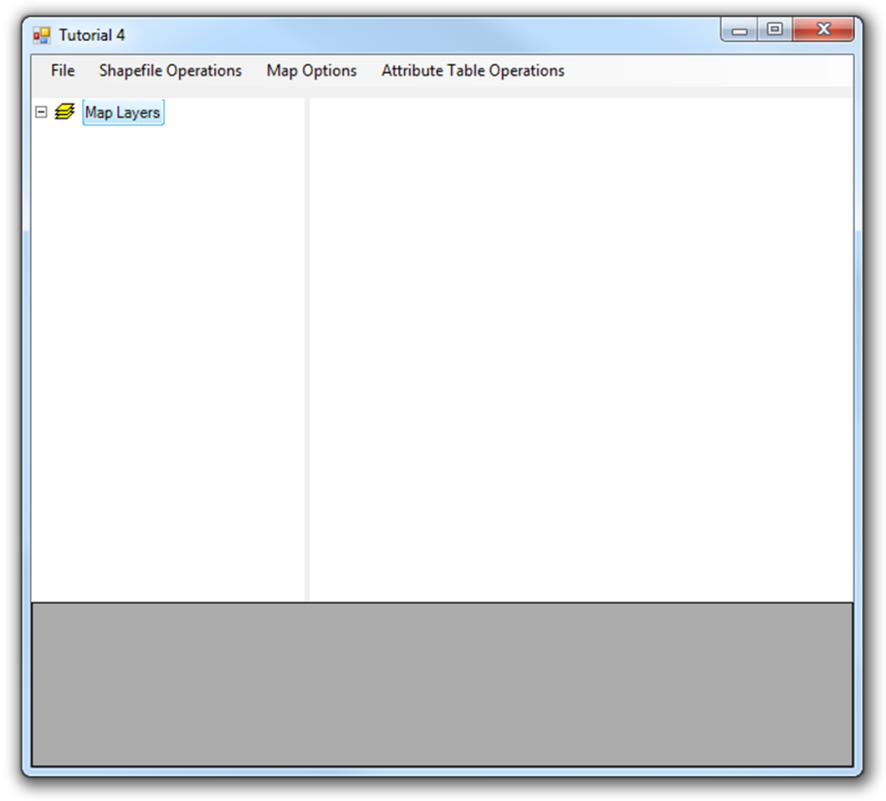


Figure 2: FInal GUI

*Interface design considerations.*

1. Add two panel controls and a SpatialDockManger. Panel control's properties should be as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Properties | Panel1 | Panel2 | SpatialDockManger |
| Name | pnlOperations | pnlAttribute | sdmLegendMap |
| Dock | Top | Bottom | Fill |

2. Add a menu strip control on the pnlOperations. Menu should be contains the following menu items.

File, Shapefile Operations, Map Options, and Attribute Table Operations

Sub menu items:

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

Load, Clear, Exit

Those implementations can be found in Tutorial 2.

Under the Shapefile Operations create the following sub menu items:

Point, Polyline, Polygon

Under the Point sub menu item create the following Sub menu items:

Create Point Shapefile, Save Point Shapefile.

Under the Polyline sub menu item create the following Sub menu items:

Create Polyline Shapefile, Save Polyline Shapefile.

Under the Polygon sub menu item create the following Sub menu items:

Create Polygon Shapefile, Save Polygon Shapefile.

Under the Map Options menu item create the following sub menu items:

Information, Zoom In, Zoom Out, Zoom to Extent, Pan, and Print Map

Under the Attribute Table Operations menu item create the following sub menu items:

View Attribute Table, Add a Column, Save Column, Delete a Column, Update attribute Table, Export Attribute table to Excel.

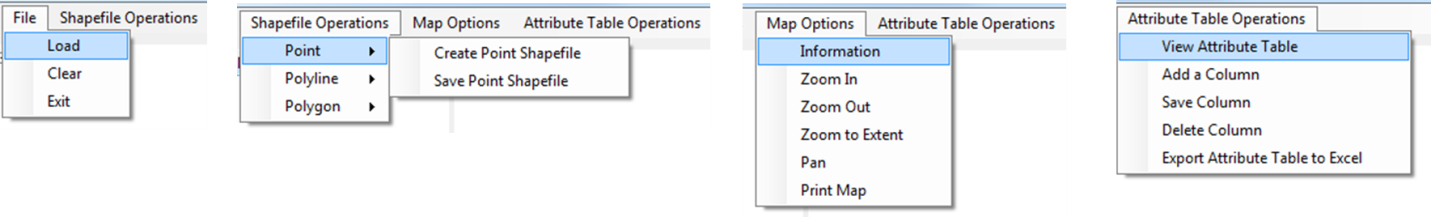


Figure 3: Menu Items

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

3.1) Add the map control to the right container of the DockManger. Add the legend control to the left container of the DockManger

Map and Legend control are located on the Toolbox under DotSpatial controls.

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

Set the legend properties as follows: Dock = Fill

4. Add a DataGridView on the pnlAttribute.

Set its properties as follows: Name : dgvAttributeTable , Dock = Fill

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

5.1) Import the following namespaces:

using DotSpatial.Controls;

using DotSpatial.Data;

using DotSpatial.Symbology;

using DotSpatial.Topology;

**Create a Point shapefile.**

Declare the following class level variables.

//which type of shapefile is created

string shapeType;

#region Point ShapeFile class level variable

//the new point feature set

FeatureSet pointF = new FeatureSet(FeatureType.Point);

//the id of point

int pointID = 0;

//to differentiate the right and left mouse click

bool pointmouseClick = false;

#endregion

Write the following code in the "CreatePointShapeFileToolStripMenuItem\_Click" event

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

{

//Change the cursor style

map1.Cursor = Cursors.Cross;

//set the shape type to the classlevel string variable

//we are going to use this variable in select case statement

shapeType = "Point";

//set projection

pointF.Projection = map1.Projection;

//initialize the featureSet attribute table

DataColumn column = new DataColumn("PointID");

pointF.DataTable.Columns.Add(column);

//add the featureSet as map layer

MapPointLayer pointLayer = (MapPointLayer)map1.Layers.Add(pointF);

//Create a new symbolizer

PointSymbolizer symbol = new PointSymbolizer(Color.Red, DotSpatial.Symbology.PointShape.Ellipse, 3);

//Set the symbolizer to the point layer

pointLayer.Symbolizer = symbol;

//Set the legentText as point

pointLayer.LegendText = "point";

//Set left mouse click as true

pointmouseClick = true;

}

Write the following code in the Map mouse down event. Select the map control and press the F4 key for getting its properties window. On the properties window select the event tab and on the event tab double click over the Mouse down.

switch (shapeType)

{

case "Point" :

if (e.Button == MouseButtons.Left)

{

if ((pointmouseClick))

{

//This method is used to convert the screen cordinate to map coordinate

//e.location is the mouse click point on the map control

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

//Create a new point

//Input parameter is clicked point coordinate

DotSpatial.Topology.Point point = new DotSpatial.Topology.Point(coord);

//Add the point into the Point Feature

//assigning the point feature to IFeature because via it only we can set the attributes.

IFeature currentFeature = pointF.AddFeature(point);

//increase the point id

pointID = pointID + 1;

//set the ID attribute

//currentFeature.DataRow["PointID"] = pointID;

//refresh the map

map1.ResetBuffer();

}

}

else

{

//mouse right click

map1.Cursor = Cursors.Default;

pointmouseClick = false;

}

break;

Save a point shape file: Write the following code into SavePointShapeFileToolStripMenuItem\_Click event.

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

{

pointF.SaveAs("point.shp", true);

MessageBox.Show("The point shapefile has been saved.");

map1.Cursor = Cursors.Arrow;

}

Output of the above code:

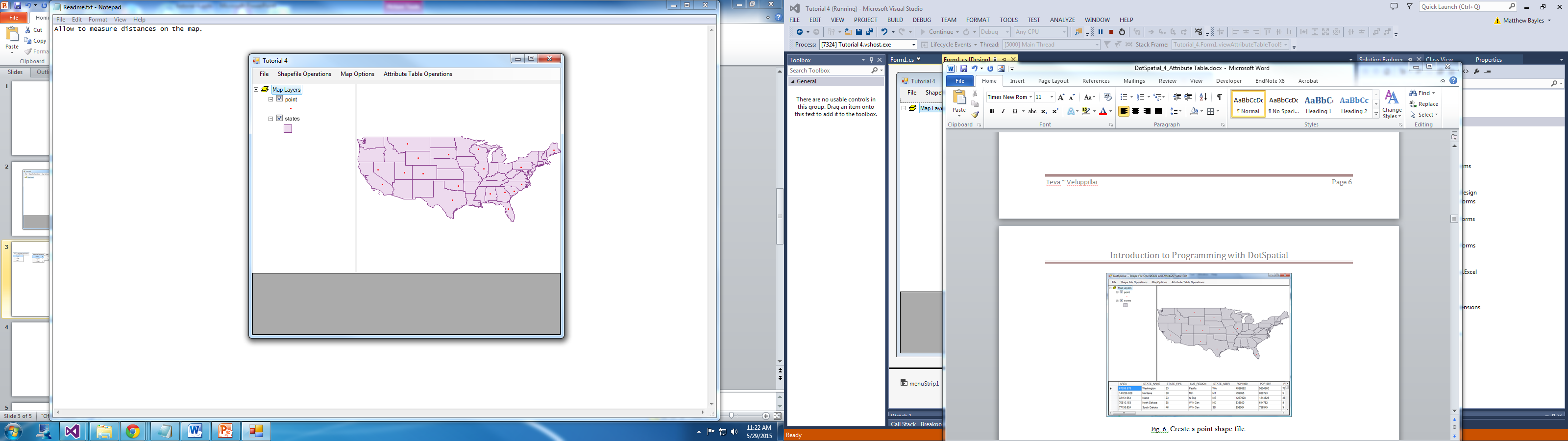


Figure 4: Point Shapefile

**Create Polyline shapefile.**

Declare the following class level variables.

#region Polyline ShapeFile class level variables

MapLineLayer lineLayer = default(MapLineLayer);

//the line feature set

FeatureSet lineF = new FeatureSet(FeatureType.Line);

int lineID = 0;

//boolean variable for first time mouse click

bool firstClick = false;

//It controls the drawing the polyline after the polyline saved operation.

bool linemouseClick = false;

#endregion

Write the following code into CreatePolylineShapeFileToolStripMenuItem\_Click event

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

{

//Change the mouse cursor

map1.Cursor = Cursors.Cross;

//set shape type

shapeType = "line";

//set projection

lineF.Projection = map1.Projection;

//initialize the featureSet attribute table

DataColumn column = new DataColumn("LineID");

if (!lineF.DataTable.Columns.Contains("LineID"))

{

lineF.DataTable.Columns.Add(column);

}

//add the featureSet as map layer

lineLayer = (MapLineLayer)map1.Layers.Add(lineF);

//Set the symbolizer to the line feature.

LineSymbolizer symbol = new LineSymbolizer(Color.Blue, 2);

lineLayer.Symbolizer = symbol;

lineLayer.LegendText = "line";

firstClick = true;

linemouseClick = true;

}

Write the following code in the Map1\_MouseDown event.

case "line":

if (e.Button == MouseButtons.Left)

{

//left click - fill array of coordinates

//coordinate of clicked point

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

if (linemouseClick)

{

//first time left click - create empty line feature

if (firstClick)

{

//Create a new List called lineArray.

//This list will store the Coordinates

//We are going to store the mouse click coordinates into this array.

List<Coordinate> lineArray = new List<Coordinate>();

//Create an instance for LineString class.

//We need to pass collection of list coordinates

LineString lineGeometry = new LineString(lineArray);

//Add the linegeometry to line feature

IFeature lineFeature = lineF.AddFeature(lineGeometry);

//add first coordinate to the line feature

lineFeature.Coordinates.Add(coord);

//set the line feature attribute

lineID = lineID + 1;

lineFeature.DataRow["LineID"] = lineID;

firstClick = false;

}

else

{

//second or more clicks - add points to the existing feature

IFeature existingFeature = lineF.Features[lineF.Features.Count - 1];

existingFeature.Coordinates.Add(coord);

//refresh the map if line has 2 or more points

if (existingFeature.Coordinates.Count >= 2)

{

lineF.InitializeVertices();

map1.ResetBuffer();

}

}

}

}

else

{

//right click - reset first mouse click

firstClick = true;

map1.ResetBuffer();

}

break;

Save the Polyline shapefile

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

{

lineF.SaveAs("c:\\MW\\line.shp", true);

MessageBox.Show("The line shapefile has been saved in C:\\MW\\");

map1.Cursor = Cursors.Arrow;

linemouseClick = false;

} }

Output of the above code

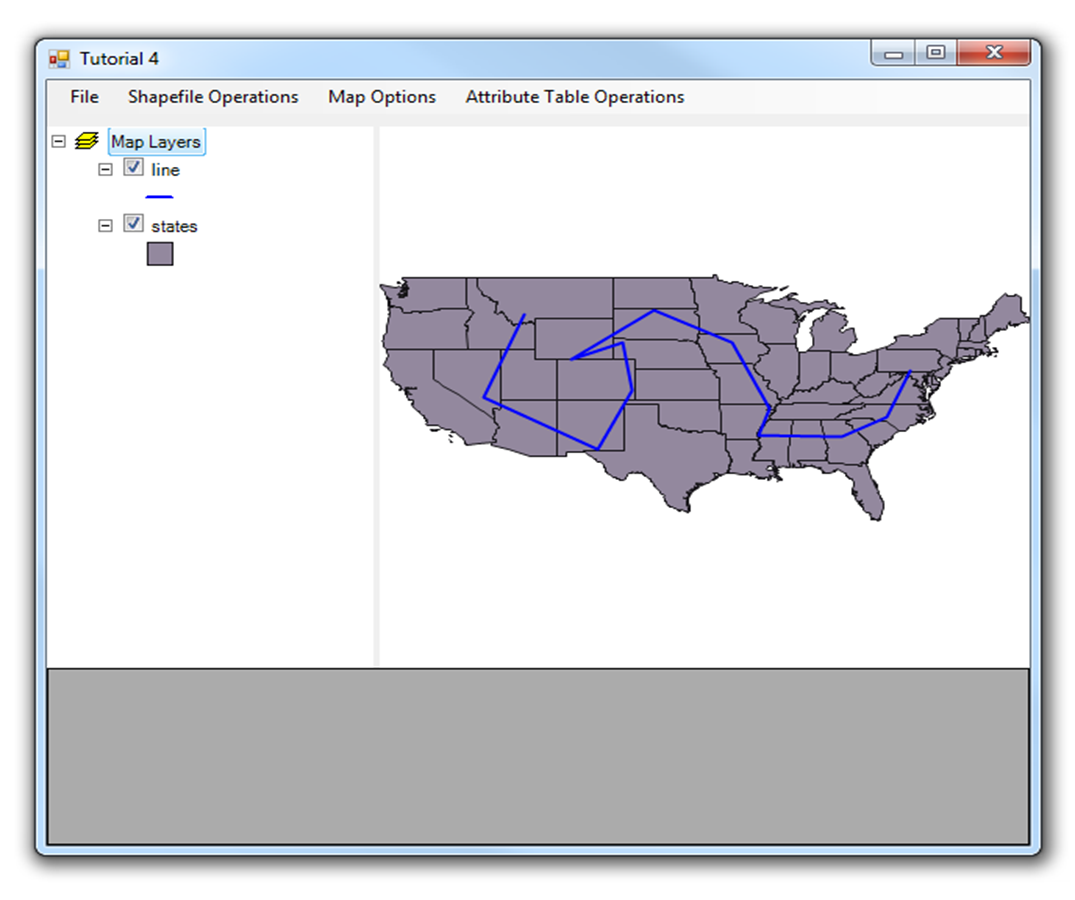


Figure 5: Polyline Shapefile

**Create Polygon shapefile.**

Declare the following class level variables.

#region Polygon ShapeFile class level variables

FeatureSet polygonF = new FeatureSet(FeatureType.Polygon);

int polygonID = 0;

bool polygonmouseClick = false;

#endregion

Write the following code into CreatePolygonShapeFileToolStripMenuItem\_Click event

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

{

//initialize polyline feature set

map1.Cursor = Cursors.Cross;

//set shape type

shapeType = "polygon";

//set projection

polygonF.Projection = map1.Projection;

//initialize the featureSet attribute table

DataColumn column = new DataColumn("PolygonID");

if (!polygonF.DataTable.Columns.Contains("PolygonID"))

{

polygonF.DataTable.Columns.Add(column);

}

//add the featureSet as map layer

MapPolygonLayer polygonLayer = (MapPolygonLayer)map1.Layers.Add(polygonF);

PolygonSymbolizer symbol = new PolygonSymbolizer(Color.Green);

polygonLayer.Symbolizer = symbol;

polygonLayer.LegendText = "polygon";

firstClick = true;

polygonmouseClick = true;

}

Write the following code in the Map1\_MouseDown event.

case "polygon":

if (e.Button == MouseButtons.Left)

{

//left click - fill array of coordinates

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

if (polygonmouseClick)

{

//first time left click - create empty line feature

if (firstClick)

{

//Create a new List called polygonArray.

//this list will store the Coordinates

//We are going to store the mouse click coordinates into this array.

List<Coordinate> polygonArray = new List<Coordinate>();

//Create an instance for LinearRing class.

//We pass the polygon List to the constructor of this class

LinearRing polygonGeometry = new LinearRing(polygonArray);

//Add the polygonGeometry instance to PolygonFeature

IFeature polygonFeature = polygonF.AddFeature(polygonGeometry);

//add first coordinate to the polygon feature

polygonFeature.Coordinates.Add(coord);

//set the polygon feature attribute

polygonID = polygonID + 1;

polygonFeature.DataRow["PolygonID"] = polygonID;

firstClick = false;

}

else

{

//second or more clicks - add points to the existing feature

IFeature existingFeature = (IFeature)polygonF.Features[polygonF.Features.Count - 1];

existingFeature.Coordinates.Add(coord);

//refresh the map if line has 2 or more points

if (existingFeature.Coordinates.Count >= 3)

{

//refresh the map

polygonF.InitializeVertices();

map1.ResetBuffer();

}

}

}

}

else

{

//right click - reset first mouse click

firstClick = true;

}

break;

Save the Polygon shapefile

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

{

polygonF.SaveAs("c:\\MW\\polygon.shp", true);

MessageBox.Show("The polygon shapefile has been saved.");

map1.Cursor = Cursors.Arrow;

polygonmouseClick = false;

}

Output of the above code

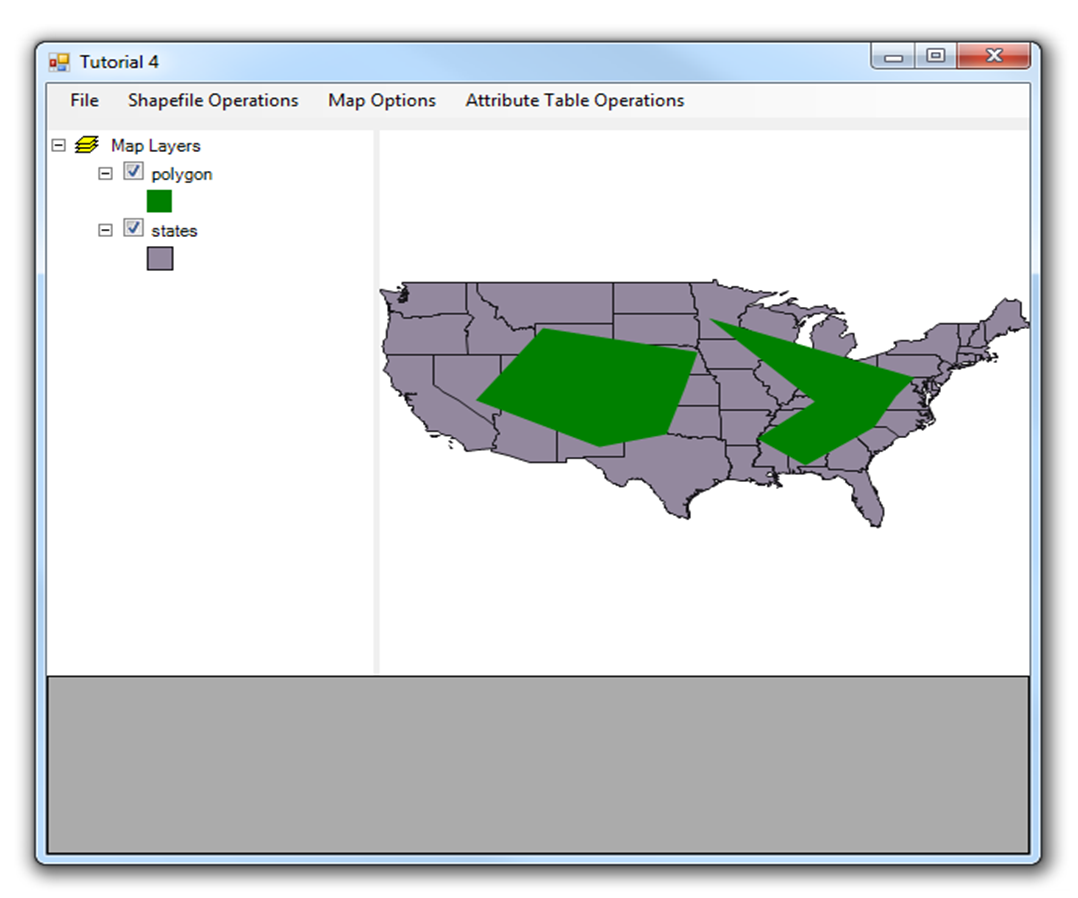


Figure 6: Polygon Shapefile

Map1 mouse down event complete code: In the above examples the same event is used to implement different shape functionalities. So, the following code is the complete code for Map1 mouse down event. The following code shows the all different shape operations.

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

{

switch (shapeType)

{

case "Point" :

if (e.Button == MouseButtons.Left)

{

if ((pointmouseClick))

{

//This method is used to convert the screen cordinate to map coordinate

//e.location is the mouse click point on the map control

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

//Create a new point

//Input parameter is clicked point coordinate

DotSpatial.Topology.Point point = new DotSpatial.Topology.Point(coord);

//Add the point into the Point Feature

//assigning the point feature to IFeature because via it only we can set the attributes.

IFeature currentFeature = pointF.AddFeature(point);

//increase the point id

pointID = pointID + 1;

//set the ID attribute

currentFeature.DataRow["PointID"] = pointID;

//refresh the map

map1.ResetBuffer();

}

}

else

{

//mouse right click

map1.Cursor = Cursors.Default;

pointmouseClick = false;

}

break;

case "line":

if (e.Button == MouseButtons.Left)

{

//left click - fill array of coordinates

//coordinate of clicked point

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

if (linemouseClick)

{

//first time left click - create empty line feature

if (firstClick)

{

//Create a new List called lineArray.

//This list will store the Coordinates

//We are going to store the mouse click coordinates into this array.

List<Coordinate> lineArray = new List<Coordinate>();

//Create an instance for LineString class.

//We need to pass collection of list coordinates

LineString lineGeometry = new LineString(lineArray);

//Add the linegeometry to line feature

IFeature lineFeature = lineF.AddFeature(lineGeometry);

//add first coordinate to the line feature

lineFeature.Coordinates.Add(coord);

//set the line feature attribute

lineID = lineID + 1;

lineFeature.DataRow["LineID"] = lineID;

firstClick = false;

}

else

{

//second or more clicks - add points to the existing feature

IFeature existingFeature = lineF.Features[lineF.Features.Count - 1];

existingFeature.Coordinates.Add(coord);

//refresh the map if line has 2 or more points

if (existingFeature.Coordinates.Count >= 2)

{

lineF.InitializeVertices();

map1.ResetBuffer();

}

}

}

}

else

{

//right click - reset first mouse click

firstClick = true;

map1.ResetBuffer();

}

break;

case "polygon":

if (e.Button == MouseButtons.Left)

{

//left click - fill array of coordinates

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

if (polygonmouseClick)

{

//first time left click - create empty line feature

if (firstClick)

{

//Create a new List called polygonArray.

//this list will store the Coordinates

//We are going to store the mouse click coordinates into this array.

List<Coordinate> polygonArray = new List<Coordinate>();

//Create an instance for LinearRing class.

//We pass the polygon List to the constructor of this class

LinearRing polygonGeometry = new LinearRing(polygonArray);

//Add the polygonGeometry instance to PolygonFeature

IFeature polygonFeature = polygonF.AddFeature(polygonGeometry);

//add first coordinate to the polygon feature

polygonFeature.Coordinates.Add(coord);

//set the polygon feature attribute

polygonID = polygonID + 1;

polygonFeature.DataRow["PolygonID"] = polygonID;

firstClick = false;

}

else

{

//second or more clicks - add points to the existing feature

IFeature existingFeature = (IFeature)polygonF.Features[polygonF.Features.Count - 1];

existingFeature.Coordinates.Add(coord);

//refresh the map if line has 2 or more points

if (existingFeature.Coordinates.Count >= 3)

{

//refresh the map

polygonF.InitializeVertices();

map1.ResetBuffer();

}

}

}

}

else

{

//right click - reset first mouse click

firstClick = true;

}

break;

}

}

**Attribute table related operations**

View the attribute table from the US States shapefile.

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

{

//Declare a datatable

System.Data.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.");

}

}

5.1.1) Output of the above code

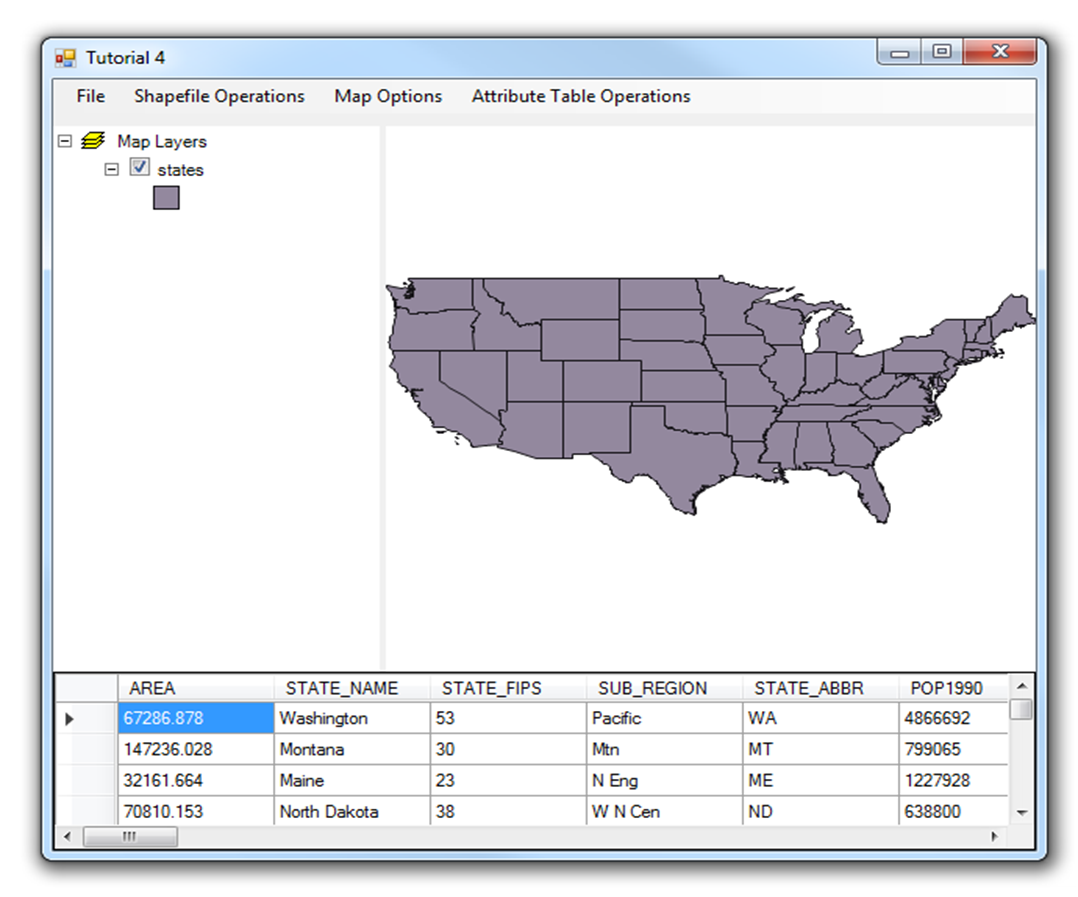


Figure 7: Attribute Table

Add a new column into the attribute table. In the following example the new column name is "PercentMales"

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

{

//Declare a datatable

System.Data.DataTable dt = null;

//check the layers

if (map1.Layers.Count > 0)

{

//Declare a mappolygon layer

MapPolygonLayer stateLayer = default(MapPolygonLayer);

//Assign the mappolygon layer from the map

stateLayer = map1.Layers[0] as MapPolygonLayer;

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;

//Add new column

DataColumn column = new DataColumn("PercentMales");

dt.Columns.Add(column);

//calculate values

foreach (DataRow row in dt.Rows)

{

double males = Convert.ToDouble(row["males"]);

double females = Convert.ToDouble(row["females"]);

double malesPercentage = (males / (males + females)) \* 100;

row["PercentMales"] = malesPercentage;

}

//Set the datagridview datasource from datatable dt

dgvAttributeTable.DataSource = dt;

}

}

else

{

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

}

}

Output of the above code.

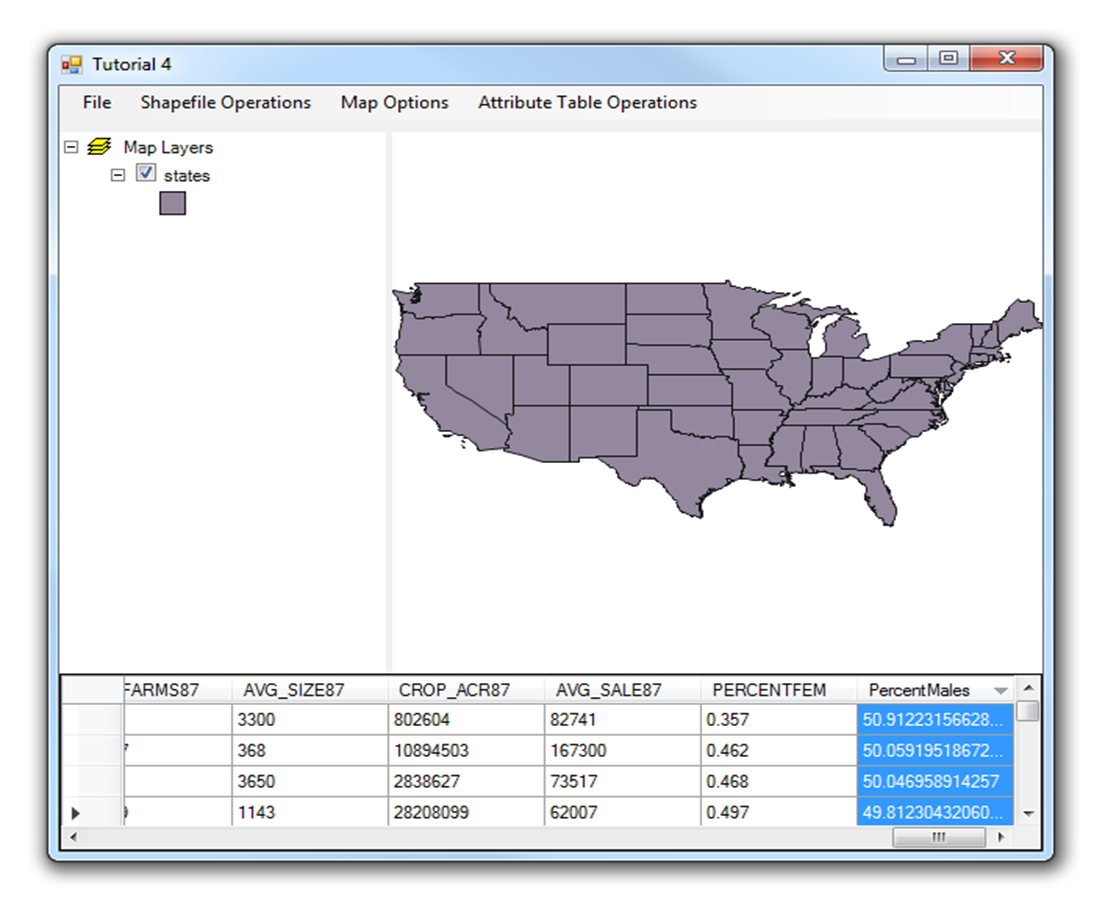


Figure 8: New Column

Update the shapefile's attribute table.

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

{

if (map1.Layers.Count > 0)

{

MapPolygonLayer stateLayer = default(MapPolygonLayer);

stateLayer = map1.Layers[0] as MapPolygonLayer;

if (stateLayer == null)

{

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

}

else

{

stateLayer.DataSet.Save();

}

}

else

{

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

}

}

Delete the attribute column from the attribute table. In the following example "PercentMales" column will be deleted.

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

{

//Declare a datatable

System.Data.DataTable dt = null;

if (map1.Layers.Count > 0)

{

MapPolygonLayer stateLayer = default(MapPolygonLayer);

stateLayer = map1.Layers[0] as MapPolygonLayer;

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;

//Remove a column

dt.Columns.Remove("PercentMales");

//Set the datagridview datasource from datatable dt

dgvAttributeTable.DataSource = dt;

}

}

else

{

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

}

}

Export an attribute table to Excel datasheet.Right click over the project on the solution explorer. On the context menu select the add reference and add the .NET reference: Microsoft.Office.Interop.Excel

Write the following code into the ExportAttributeTableToExcelToolStripMenuItem\_Click event.

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

{

//Declare a datatable

System.Data.DataTable dt = null;

if (map1.Layers.Count > 0)

{

MapPolygonLayer stateLayer = default(MapPolygonLayer);

stateLayer = map1.Layers[0] as MapPolygonLayer;

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;

//Call the sub ExportToExcel

//This sub procedure expects a datatable as an input

ExportToExcel(dt);

}

}

else

{

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

}

}

Write the following sub method for getting an excel datasheet output from an attribute table.

/// <summary>

/// This sub method is used to create an excel worksheet from the attribute table

/// </summary>

/// <param name="objDT">attribute table as a datatable input</param>

/// <remarks>Click the COM tab of the Add Reference dialog box, and find Microsoft Excel 14 Object Library.</remarks>

private void ExportToExcel(System.Data.DataTable objDT)

{

//excel = new Excel.Application();

Microsoft.Office.Interop.Excel.Application xlApp = new Microsoft.Office.Interop.Excel.Application();

string strFilename = null;

int intCol = 0;

int intRow = 0;

//path for storing excel datasheet

string strPath = "C:\\2009 Falls\\";

if (xlApp == null)

{

MessageBox.Show("It appears that Excel is not installed on this machine. This operation requires MS Excel to be installed on this machine.", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

return;

}

try

{

//var \_with1 = Microsoft.Office.Interop.Excel.Application();

xlApp.SheetsInNewWorkbook = 1;

xlApp.Workbooks.Add();

xlApp.Worksheets[1].Select();

xlApp.Cells[1, 1].value = "Attribute table";

//Heading of the excel file

xlApp.Cells[1, 1].EntireRow.Font.Bold = true;

//Add the column names from the attribute table to excel worksheet

int intI = 1;

for (intCol = 0; intCol <= objDT.Columns.Count - 1; intCol++)

{

xlApp.Cells[2, intI].value = objDT.Columns[intCol].ColumnName;

xlApp.Cells[2, intI].EntireRow.Font.Bold = true;

intI += 1;

}

//Add the data row values from the attribute table to ecxel worksheet

intI = 3;

int intK = 1;

for (intCol = 0; intCol <= objDT.Columns.Count - 1; intCol++)

{

intI = 3;

for (intRow = 0; intRow <= objDT.Rows.Count - 1; intRow++)

{

xlApp.Cells[intI, intK].Value = objDT.Rows[intRow].ItemArray[intCol];

intI += 1;

}

intK += 1;

}

if (strPath.Substring(strPath.Length - 1, 1) != "\\")

{

strPath = strPath + "\\";

}

strFilename = strPath + "AttributeTable.xls";

xlApp.ActiveCell.Worksheet.SaveAs(strFilename);

System.Runtime.InteropServices.Marshal.ReleaseComObject(xlApp);

xlApp = null;

MessageBox.Show("Data's are exported to Excel Succesfully in '" + strFilename + "'", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

// The excel is created and opened for insert value. We most close this excel using this system

System.Diagnostics.Process[] pro = (System.Diagnostics.Process[])System.Diagnostics.Process.GetProcessesByName("EXCEL");

foreach (System.Diagnostics.Process i in pro)

{

i.Kill();

}

}

**Printing a map in DotSpatial**

Write the following code into the PrintMapToolStripMenuItem\_Click event

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

{

DotSpatial.Controls.LayoutForm frm = new DotSpatial.Controls.LayoutForm();

frm.MapControl = map1;

frm.Show();

}

Output of the above code is follows:

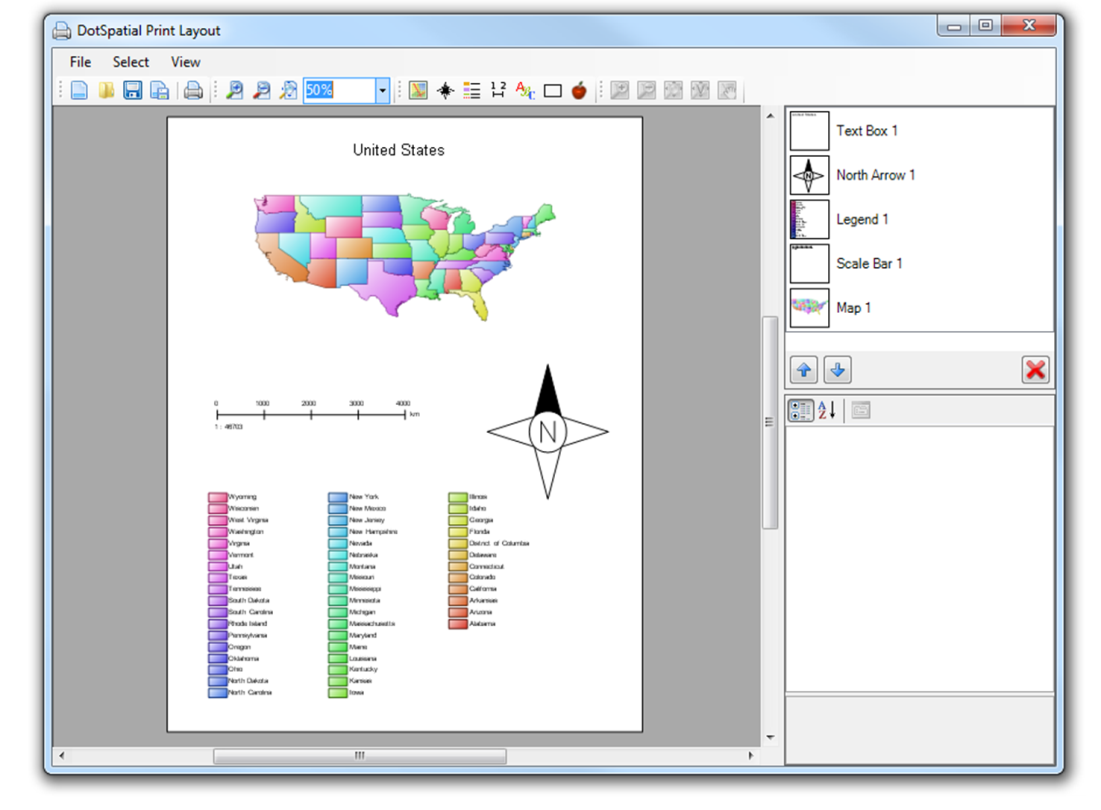


Figure 9: Print Layout