SOIL PROFILER

April 16, 2015

brigham younng university

PROVO, UTAH

Technical and Design Documentation

Technical Specifications

# Software Details

This standalone desktop application was built for creating an interactive user interface to show the soil profile of any drawn cross section. The application has been designed using C# and DotSpatial GIS library. The Microsoft Visual Studio 2012 was used as the development environment. The application was developed by Sam Ruggles and Sarva Pulla. It can be downloaded at https://github.com/SarvaPulla/SoilProfiler

# Compatibility and Limitations

The application is limited to users with windows operating systems. However, it can be used in other operating systems by installing implementation software such as Mono for Mac. The availability of soil raster data is extremely limited. We hope that more data will be available in the future, thus allowing the application to be used to its full potential. For any issues or troubleshooting assistance, please contact Sam Ruggles ( [sruggles@byu.edu](mailto:sruggles@byu.edu)) or Sarva Pulla ([sarva@byu.edu](mailto:sarva@byu.edu)).

# License

Soil Profiler by Sam Ruggles & Sarva Pulla is licensed under a [Creative Commons Attribution­NonCommercial­ShareAlike 3.0 Unported License](http://creativecommons.org/licenses/by-nc-sa/3.0/deed.en_US)

software design

# Purpose of Design

The purpose of this application is to serve as a standalone desktop application for providing a soil profile of multiple raster layers. The multiple raster layers are projected onto a single map and there are two tools main tools. One to draw a line/polyline and another one to graph the elevation of the drawn polyline. Limiting the number of tools and providing only the bare minimum options has made the application very straightforward to use

All the features and functionality have been included in the main form, so that a user doesn’t have to browse through multiple menus/options. Thus, making the application user-friendly and easy to navigate. A single menu has been used to make the application simple and straightforward.

# Header and Interface Elements

The main menu (See Figure 1) is simple and just has there elements Layers, View and Tools dropdown menus. This allows the application to be neat and look professional. Clicking on any of the three items will show you a list of the tools that are available.

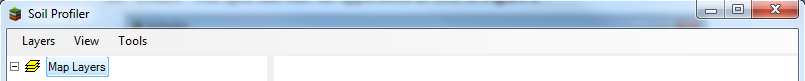


Figure 1: Main Menu

The following is a list of tools that each button will display (See figure 2). Clicking on Layers will give the option of loading a raster layer. Clicking on view you have option of zooming in or out of the loaded raster layer. Clicking on tools will provide you with the option of drawing a line and viewing the soil profile.

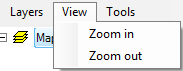
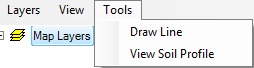


Figure 2: Dropdown Menus

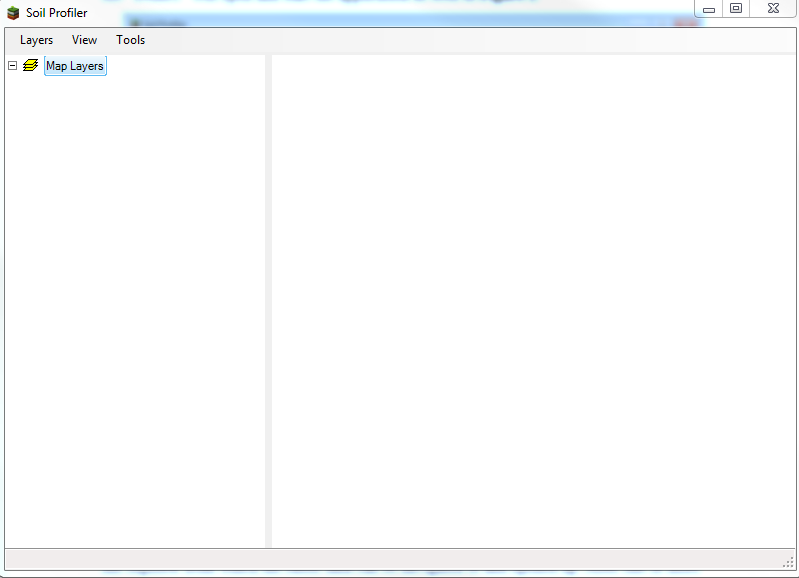


Figure 3: Panels

Figure 3, is how the program appears to a user. The panel on the left contains the legend while the empty panel on the right is where the loaded raster data will be displayed. Once the view soil profile is clicked then another form with the graph will be displayed (See figure 4)

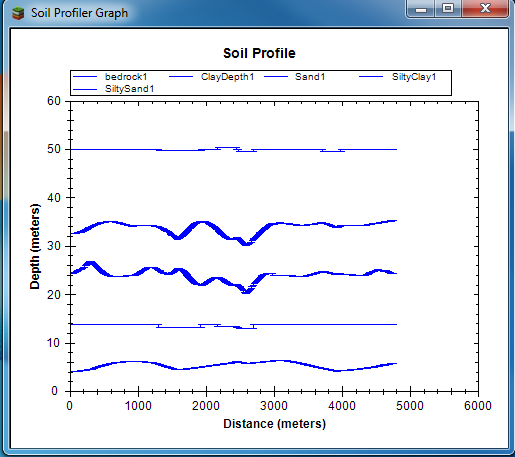


Figure 4: Soil Profiler Graph

# Main Files

The main files of these application are Form1. cs, frmGraph.cs, and PathPoint.cs. The GeoDll, ZedGraph and DotSpatial extensions play a crucial rule in executing the application. Most of the folders were automatically generated by Visual Studio. See Figure 5 for a visual of how the website folders were organized.

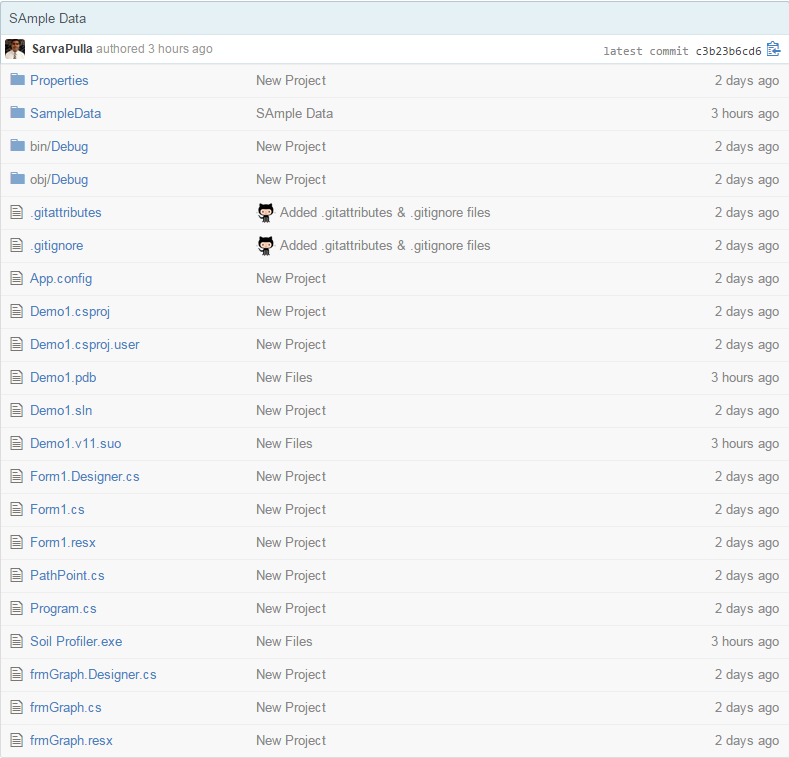


Figure 5: Working Directory

The extensions and all other main files are located in the debug folder. One can add more references via Visual Studio and they will be added automatically to the bin/debug folder. Also, the SampleData folder contains raster data that we have generated for testing. Using this rasater data, the user can see how they soil profile is generated.

# Code Structure

The basic structure of the code is represented in the flowcharts below. The custom classes are what make this application robust and very useful. The flowcharts cover the classes that are responsible for creating the polyline and extracting elevation values for generating the soil profile chart.

Figure 6 is the flow chart for how the line is being created and executed while figure 7 is the flow chart for the elevations are being extracted along these lines. Once the elevation values are extracted using the ZedGraph library and the frmGraph class we are able to plot a comprehensive Soil Profile plot.

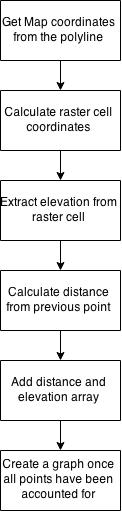


Figure 7: Extract Elevation

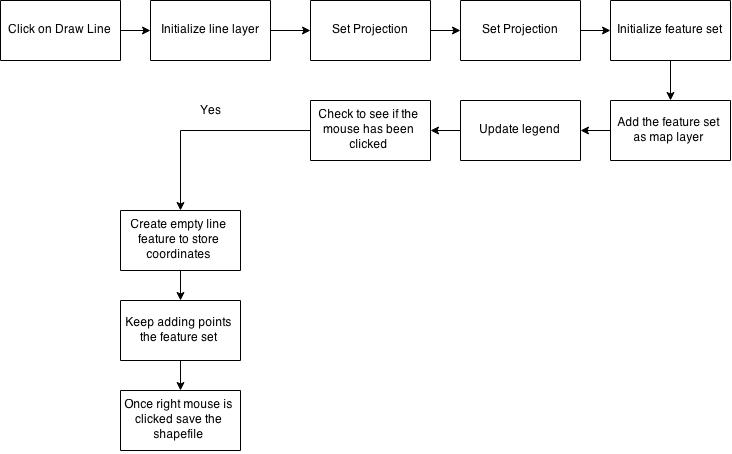


Figure 6: Flowchart for creating a polyline

# Key Functions

The following table contains all the custom classes that were used in building this application. They are presented in the order of their appearance.

|  |  |
| --- | --- |
| **Class Name** | **Related Action** |
| loadToolStripMenuItem\_Click | Responsible for adding the raster layer |
| appManager | Responsible for executing the DotSpatial extensions |
| drawToolStripMenuItem\_Click | Initializes the drawing capabilities of the program. This is lined with extract elevation and map1\_Mousedown |
| map1\_Mousedown | Starts digitizing the polyline and saves the coordinates based on user input. Will stop digitizing on right mouse click. |
| viewElevationToolStripMenuItem\_Click | Initiates the draw graph class and is responsible for calculating the distances between two neighboring points. It gets the elevation. |
| ExtractElevation | Responsible for creating points and extracting their elevation. Makes use of the PathPoint class to achieve its objective |
| zoomInToolStripMenuItem\_Click | Zooms in to the raster layer. In case the user is very particular about getting the profile of small area |
| zoomOutToolStripMenuItem\_Click | Zooms out of the raster layer |
| frmGraph | Captures the elevation values form the viewElevationToolStripMenuItem and initializes the ZedGraph process. Using the PathPoint class and the points from ExtractElevation we are able to populate the graph. Once that is done, we use ZedGraph controls to produce the Soil Profile graph |
| PathPoint | A custom class that is created to store points and variables that are responsible for storing them. |