TaskGIS
Software Design Specification

Author

Srividya Subramanian
B.E. IT Third Year,
Fr. C. Rodrigues Institute of Technology, Vashi, 400703

Technical Guide

Shashank Saindane
Scientific Officer,
Bhabha Atomic Research Center, Mumbai, 400085

Overview

TaskGIS is a QGIS tool that can be used to query places and find the shortest path (with recalculation of the next simplest shortest path) for the given shape file. The design of the tool is Object Oriented as it inherits from existing QGIS map tools. TaskGIS also consists of a minimalistic User Interface built using PyQt5 and it uses QGIS Processing Algorithms at its core. This document covers the Functional Description, the Software Requirements and Settings, the Code Design and the Milestones accomplished throughout the development process.

Table of Contents

Chapter 1

Functionality and Requirements

1.1 Functional Description

TaskGIS can be used to perform two tasks -

1. **Query Places within Sectors of a Circle**

   This functionality allows the user to select a point on the map canvas. Upon selection the user will be prompted to enter the radius of the circle to be drawn in kilometers. Once the radius is entered a Circle is drawn with 16 Sectors (numbered 1 -16). The user is now offered a simple selection tool which will list all the places that lie within a particular Sector upon clicking it. The user may also change the center point of the Circle using the provided Keyboard shortcut.

2. **Find Shortest Path with Recalculation**

   Using this tool the user can select 2 locations, a Source and a Destination. Upon selection of these points the tool uses the Road network to find the shortest path between those two locations. If this path has an impedance on it, the user may use the provided keyboard shortcut to recalculate a new path. This can be done as many times as the user wants and as long as a simple shortest path exists between the source and the destination. If there is no new path then the user will be notified of the same.

Along with both these tools the user is also provided the functionality to Pan, Zoom In and Zoom Out over the Map Canvas and conveniently switch between all these tools using the provided Keyboard Shortcuts.

1.2 Software Requirements

This tool is platform independent since it runs on the QGIS Python Console and QGIS is available for Windows, Linux, MAC OS and Android. The tool will run on QGIS versions 3.16 (Hannover) and 3.18 (Zurich). However a fresh install

of QGIS 3.18 is highly recommended. Furthermore, QGIS requires a Python 3.7 Interpreter to be installed on the system in order for scripts to be run on the QGIS Python Console. The following points elaborate further on the softwares required:

1. QGIS 3.18 Zurich

QGIS is a user-friendly Open Source Geographic Information System (GIS) licensed under the GNU General Public License. QGIS is an official project of the Open Source Geospatial Foundation (OSGeo).It supports numerous vector, raster, and database formats and functionalities .QGIS provides a continuously growing number of capabilities provided by core functions and plugins. One can visualize, manage, edit, analyse data, and compose printable maps. The process for installing QGIS 3.18 can be found here.

2. Python 3.7.X

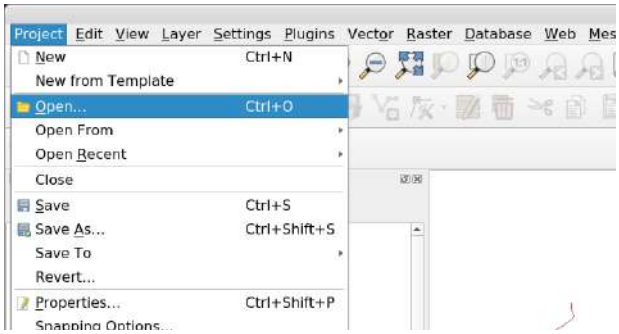
Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python 3.7 provides a Builtin Breakpoint Function, Data Classes, Increased time precision, Performance enhancement to Annotations, Ordered Dictionaries and Miscellaneous Optimisations. The process for installing Python 3.7.X can be found here.

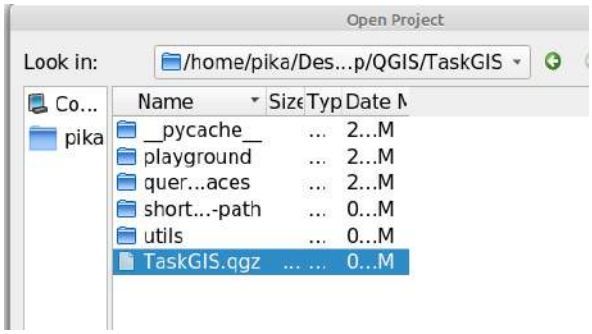
Chapter 2

Prerequisites for Tool Operation

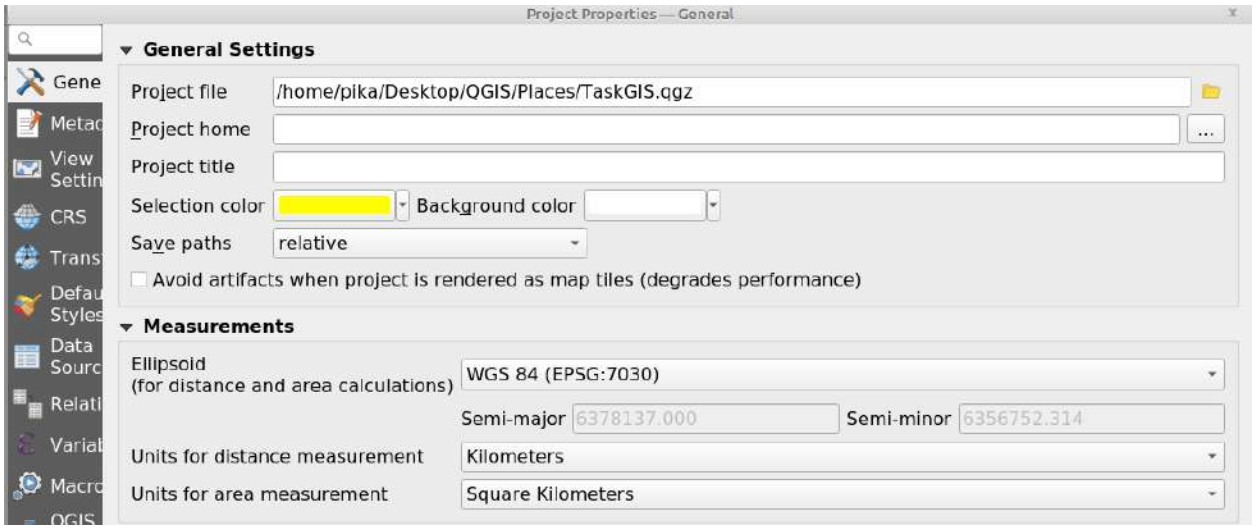
2.1 Initial Settings for QGIS

The first step is to open the QGIS desktop Application and load in the 'TaskGIS.qgz' Project. This can be done by clicking the 'Open' option from the 'Project' Menu as demonstrated in the figure given below.



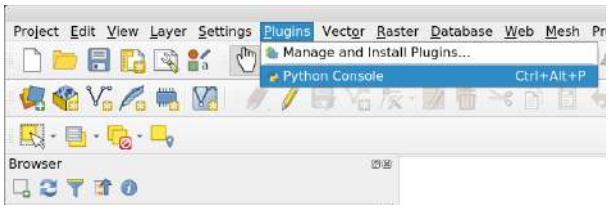


Next, it must be ensured that the Project Units are set to Kilometers. This can be done by navigating to the 'Properties' option in the 'Project' Menu.



The final step requires the Python Console to be open and the necessary scripts to be loaded in. This can be done by clicking the 'Python Console' option in the 'Plugins' menu. Once the Console is visible the script *'main.py'*

should be loaded via the ‘Show Editor’ option followed by clicking on the ‘Open Script’ button as demonstrated below.

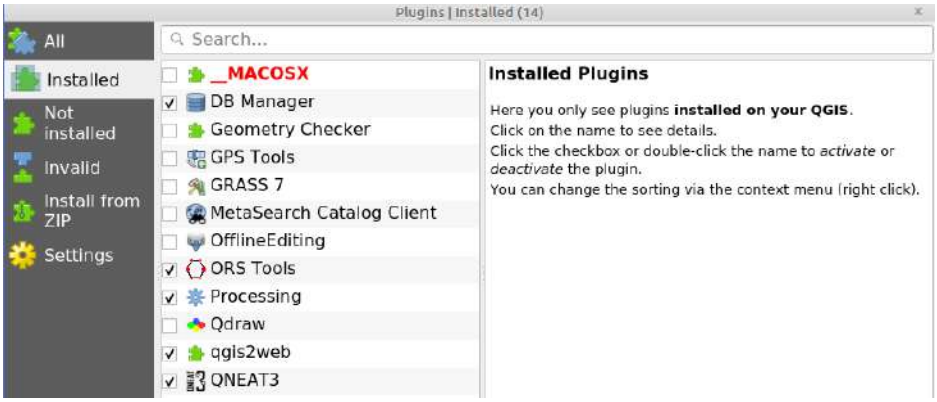


2.2 Installation of Essential Plugins

Certain QGIS Plugins are essential for the functioning of the tool. These are listed below:

1. Processing
2. QNEAT3

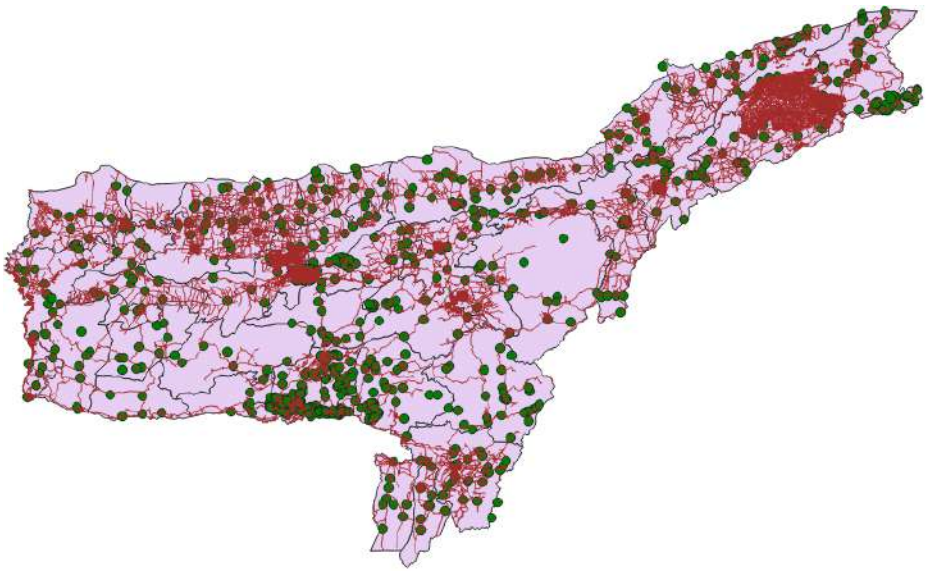
These plugins can be installed via the ‘Manage and Install Plugins’ option in the Plugins menu. The required plugin can be searched for in the Plugins search bar and it can be installed by clicking on the ‘Install Plugin’ button in the bottom right corner. This process is demonstrated below:



2.3 Shape Files

This tool requires a minimum of 2 Shape Files to operate which are provided with the code in a folder named ‘shape-files’ inside the root project directory. The third shape file that is the Base Map for the region is optional. The default Shape Files provided are:

1. Places.gpkg - Point Vector Layer containing all Places in Assam and the Information related to them
2. Roads.gpkg - A MultiLine Vector Layer containing a network of Roads in Assam.
3. BaseMap.gpkg - Base Map of the state of Assam, India



2.3 Alternate Shape Files For Network Analysis

If an alternate Road Network shape file needs to be loaded in it is imperative that all its features of this file have an attribute named **‘maxspeed’** which is set to a value of **5**. This attribute is used for the calculation of the shortest path using Time Optimisation and the Recalculation of the next shortest path. The

input for these shape files can be provided in the **'shape.config.py'** provided in the root directory. This config file consists of 3 variables and the user is expected to provide an input which will be only the name of the required input shape file. This shape file must be placed in the 'shape-file' folder in the root directory for the tool to be able to access it.

Instructions for input shape file -

1. File type - The file type of the input shape file is highly recommended to be '.gpkg' that is a GeoPackage file. Files with extensions '.shp' (ESRI) may also be used but it should be ensured that all the other required files (corresponding .shx, .dbf, .sbn etc.) are also present in the folder.
2. Invalid Geometries in the road Network- The input network file must not have invalid geometries. This will cause the QNEAT3 Graph Builder to fail and hence the shortest path will not be found. IN order to fix Invalid Geometries, the layer can be fed to the Check Validity tool in the Vector Geometry toolbox and all the features with invalid geometries must be removed.
3. Layer CRS - The Coordinate Reference System should be set for all the layers. An accepted and tested CRS is EPSG:4326. This can be done either in QGIS or via the ogr2ogr utility provided by the gdal-bin package in Linux.

Chapter 3

Code Design

3.1 Directory Structure

This section covers the directory structure of the code. The following image shows how the various files are organised.

1. ***main.py*** - The main driver code which provides a menu to the user and initialises tools on the map canvas accordingly.
2. ***playground*** - This folder contains code written for learning purposes.
3. ***query-places*** - This folder contains all files related to the Query Sector Places functionality.
    a. ***draw_circle.py*** - This file contains code responsible for drawing Circles and their Sectors on the map canvas.
    b. ***query_sector.py*** - This file contains code responsible for querying places from a selected sector.

4. ***shape.config.py*** - This file consists of the input for the various Shape files for the tool.
5. ***shape-files*** - This folder consists of the various shape files required for running the tool.
6. ***shortest-path*** - This folder contains all files related to the Shortest Path functionality.
    a. ***find_path.py*** - This file contains code responsible for inputting two points and finding the shortest path between them.
7. ***TaskGIS.qgz*** - QGIS Project file
8. ***utils*** - This folder contains utility functionality which will be used across all files.
    a. **initialise_speeds.py** - This file is used to initialize all the 'maxspeed' attribute values of the Roads ShapeFile to 5.
    b. ***load.py*** - This file is used to load in the various Shape Files.
    c. ***switch_tools.py*** - This file contains the Pan and Zoom Tools which have a switching functionality added to them.

3.2 Design

Both of the tools are made up of Classes that Inherit from the QgsMapTool class. These tools override a few methods from their parent and have several new methods defined in them. Each Class (Tool) performs a certain task.

QUERY SECTOR PLACES

DrawSectorCircle - This class is responsible for letting the user select a point on the canvas, prompting the user to enter the radius for the Circle and then drawing the Circle with 16 Sectors and then transferring the control to the *QuerySectorPlaces* Class. The circle is drawn with 8 diameters as a temporary Vector Layer. It has the following instance variables .

```
class DrawSectorCircle(QgsMapTool):
    def __init__(self, canvas, iface):
        self.canvas = canvas
        self.iface = iface
        self.x = 0
        self.y = 0
        self.circle = QgsVectorLayer()
        self.line_layers = []
        self.toolPan = tool_file.switchPanTool(self.canvas, self.iface, 'draw')
        self.toolZoomIn = tool_file.switchZoomTool(self.canvas, self.iface, False, 'draw')
        self.toolZoomOut = tool_file.switchZoomTool(self.canvas, self.iface, True, 'draw')
        QgsMapToolEmitPoint.__init__(self, self.canvas)
```

This class provides the following functions -

- clearCanvas() - used to clear the canvas.
- drawCircle(radius) - used to draw a Circle of the inputted radius.
- drawSectorLines(radius) - used to draw 8 diameters in the Circle for obtaining 16 Sectors.
- canvasPressEvent(e) *[overridden]* - listens for a canvas press event.
- keyReleaseEvent(e) *[overridden]* - listens for a key release event.

QuerySectorPlaces - This class lets the user select a given sector (1-16) and then displays all the Places found in that Sector. It also provides the option to switch back to the *DrawSectorCircle* class and hence change the center point of the Circle. It has the following instance variables -

```
class QuerySectorPlaces(QgsMapTool):        Find related code in QGIS
    def __init__(self, canvas, iface, point, radius, line_layers, circle):
        self.canvas = canvas
        self.iface = iface
        self.center_x = point[0]
        self.center_y = point[1]
        self.x = 0
        self.y = 0
        self.radius = radius
        self.sector_layer = QgsVectorLayer()
        self.circle = circle
        self.line_layers = line_layers
        QgsMapToolEmitPoint.__init__(self, self.canvas)
```

This class provides the following functions -

- clearCanvas() - used to clear the canvas on Quit.
- clearSector() - used to clear a previously selected Sector
- drawSector(n, r)- used to draw a new selected Sector using the Sector number (n) and the radius (r).
- identifySector() - used to find the Sector number from the clicked mouse coordinates by calculating angle from X-axis.
- getNamesInPolygon(n) - used to perform an area join between the Places layer and the selected Sector to find places lying in sector.
- canvasPressEvent(e) *[overridden]* - listens for a canvas press event.
- keyReleaseEvent(e) *[overridden]* - listens for a key release event.

FIND SHORTEST PATH

FindPath - This class is responsible for letting the user select two points on the canvas and mark them using well defined Point Layers. Then using these two points and the Road network the class calculates the shortest path between those two locations. The shortest path can be found even if the source and destination points do not lie on the Road network. It has the following instance variables -

```
class FindPath(QgsMapTool):
    def __init__(self, canvas, iface):
        self.canvas = canvas
        self.iface = iface
        self.click_count = 0
        self.origin_coords = QgsPointXY()
        self.destination_coords = QgsPointXY()
        self.origin_layer = QgsVectorLayer()
        self.destination_layer = QgsVectorLayer()
        self.shortest_path_layers = []
        self.toolPan = tool_file.switchPanTool(self.canvas, self.iface, 'path')
        self.toolZoomIn = tool_file.switchZoomTool(self.canvas, self.iface, False, 'path')
        self.toolZoomOut = tool_file.switchZoomTool(self.canvas, self.iface, True, 'path')
        self.roads_layer = QgsProject.instance().mapLayersByName('Roads')[0]
        self.resetAlteredMaxspeeds()
        QgsMapToolEmitPoint.__init__(self, self.canvas)
```

This class provides the following functions -

- clearCoords() - used for resetting the source and destination coordinates.
- resetAlteredMaxSpeeds() - this is a utility function used to query all the roads which have a maxspeed of 0 as a consequence of recalculation and reset them to 5.
- excludePath() - used for setting the maxspeed of the road feature lying in the previous shortest path to zero. It uses the Join by location Processing Algorithm to find overlap between the Road Network and the previous shortest path.
- grayPreviousPath() - used to set the colour of the previous path to gray.
- shortest_path() - used for calculating the shortest path between the two points.
- make_point_layer(point_type) - used for making source and destination demarcations.
- canvasPressEvent(e) *[overridden]* - listens for a canvas press event.
- keyReleaseEvent(e) *[overridden]* - listens for a key release event.

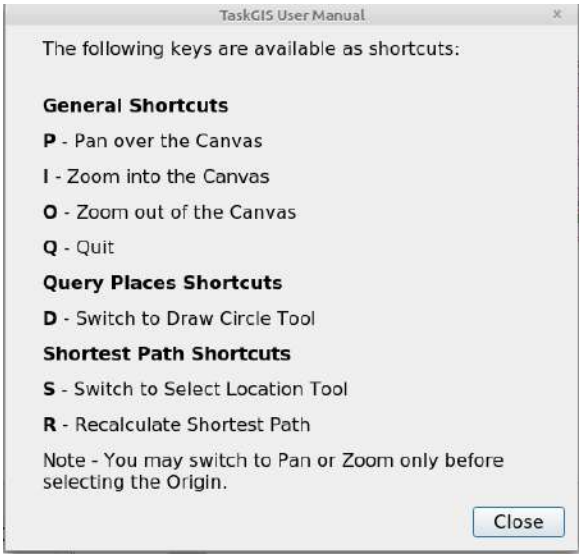
All these classes use two other utility classes which help them provide a Pan and Zoom Functionality. These two classes inherit from the *QgsMapToolPan* and the *QgsMapToolZoom* classes respectively. They provide switching functionality based on the caller tool. This is done so that the functionality of one tool does not get called from another. The ***switchPanTool*** and ***switchZoomTool*** Classes have the following initialisations -

```
class switchPanTool(QgsMapToolPan):
    def __init__(self, canvas, iface, caller_tool):
        self.canvas = canvas
        self.iface = iface
        self.caller_tool = caller_tool
        QgsMapToolEmitPoint.__init__(self, self.canvas)
```

```
class switchZoomTool(QgsMapToolZoom):
    def __init__(self, canvas, iface, inOut, caller_tool):
        self.canvas = canvas
        self.iface = iface
        self.caller_tool = caller_tool
        QgsMapToolEmitPoint.__init__(self, self.canvas, inOut)
```

3.3 User Interface

The User Interface for this tool has a minimalistic design. It has been built using the PyQt5.QtWidgets library and the PyQt5.QtGui libraries. It provides a menu to the user where they can select between the two tools. Furthermore it also provides a manual to the user where they can view all the keyboard shortcuts which can be used for switching between tools.

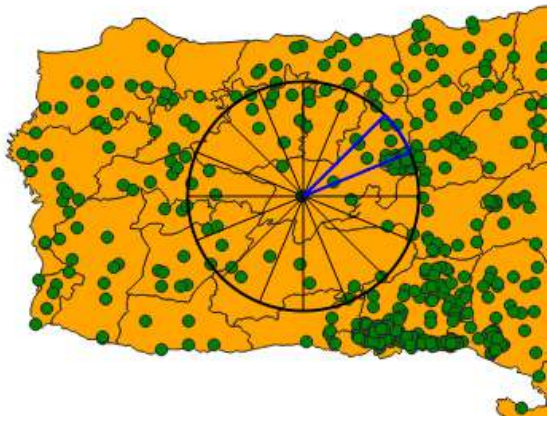


As it is clearly mentioned in the Manual, in the Find Shortest Path Tool the users must Zoom and Pan to the desired location and only then select the origin. Once the origin is selected the Zoom and Pan tools are disabled.

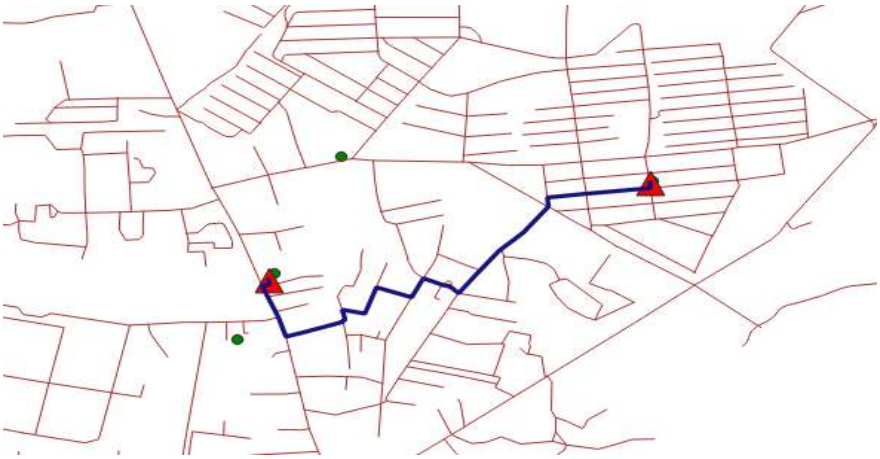
Chapter 4

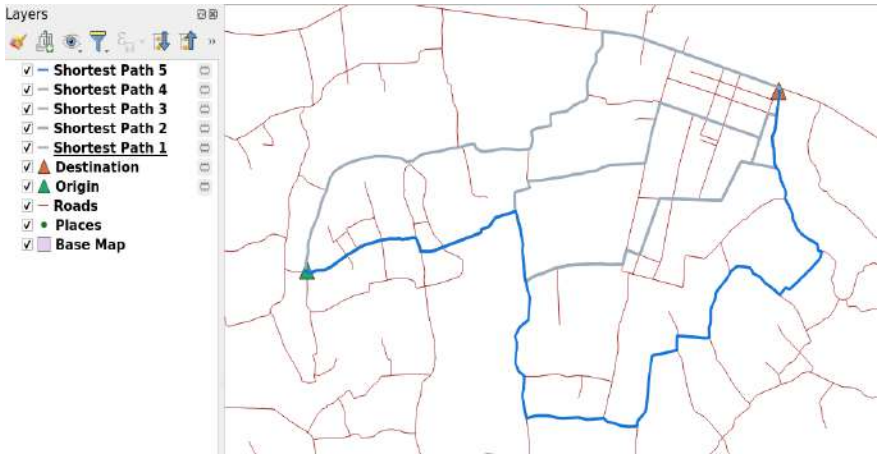
Implementation Results

4.1 Query Sector Places



4.2 Finding Shortest Path





Chapter 5

Conclusion

5.1 Milestones

The following milestones were accomplished through the course of this project.

- The basics of Geographical Information Systems were learnt by the means of QGIS Software.
- Manipulation of map elements via the Python console was implemented successfully.
- Various QGIS Processing Algorithms were explored and used for building various tools.
- PyQt5 functionalities were used for building a User Interface for the tools.
- Using the knowledge gained and research conducted, a map tool for drawing a Circle of a specified radius with 16 sectors and querying the places within the selected sector was built successfully.
- Furthermore, a map tool for finding the shortest path between two selected locations (with recalculation of shortest path) was implemented successfully.