Technical Documentation

This documentations provides insights of various modules implemented, with in-depth analysis of the functions used in the program.

Modules

1) Mouse Hover Handle: This module handles the changes in the UI when the hovers over the application. Nodes on the map highlights on the hover event and display the name of the node

2) Description and images: This module handles the event of click on a particular node of the map. The node gets highlighted with its name and picture and description of the place is displayed.

3) Search Module: This module handles the search request on the map. Text mining is used to search, for extracting particular keyword from the search text and find matches from the predefined text. Highlight the found place on the map.

4) Shortest Path Module: This module deals with finding the shortest path between two nodes in the map. Graph data structure is used to make the calculations with Dijkstras algorithm.

Functions Used

1) make_invisible(): Once you click on one object on the map, it gets selected and its color becomes prominent and text is displayed. What we want is that if we click on another object, all other objects become transparent and they don't show any text.

   This function does exactly that. First it sets all objects to be unselected, then it removes any previous text that was shown and then it restores the transparency of all the objects.

2) Obj_clk(): This is just the syntax of the functions of this kind. For instance Hospital_clk() will set the object hospital to be selected, it will then change its color to Green and also display the name (i.e Hospital) on the object. It then loads the image of the place and displays it in the pictrueBox2.

3) Obj_click(): This is just syntax of functions of this kind. For instance Hospital_click() will call make_invisible() and then Hospital_clk().

4) Obj_MouseHover(): Example of function: Hospital_MouseHover(). This checks if the object being overed on has been already selected or not. If not then it changes the color of object to blanched Almond and displays the name of the object in the label.

5) Obj_MouseLeave(): Examples of function of this kind: Hospital_MouseLeave(). This functions is called when mouse leaves from a currently hovered position. It checks if the object is selected, if not then it changes the color of the object back to transparent and removes the text shown in its label.

6) search_map(): We first define a string s that contains the lowered form of the text contained in comboBox. Then for each object, we check if this string s contains the name of the object(ex: Kapili Hostel will match with kapili). If yes, then that particular object is selected, its color is changed to green and its information is shown along with its image. We then break out of the loop. If the searched query does not match with any of the names then “No results found” is shown in the same comboBox in which the user was searching.

7) dijkstra(): This function is an implementation of dijkstra’s algorithm wih some modification to store to the parent of each node while calculating the minimum distance of all the nodes from the start node. Weighted nodes are placed over some of the legal routes of the map. Each node is implemented by labels.

8) printpath(): This function displays the minimum distance path found by the dijkstra() function. The labels which are set to invisible from the start are set to visible only in the minimum path traversed. After the query is done, they are again set to invisible so that the next query’s shortest path be set to visible.

9) distancesearch(): This function gets the start and end indexes of the chosen points on the map. It does a string search and gets the label indexes which are used in the dijkstra function.

The selected/not selected property

To store the information whether an object is being currently selected or not, what we do is that we define an array chk. We assign a unique index to each place(ex:44 to Siang hostel).

chk[44]=0 will mean that the object Siang is not being selected at that particular instance and chk[44]=1 will mean it is being selected currently.

Path labels are indexed so that the minimum distance path is accessed easily in an orderly manner.

Declaration of array:

```
array<int>^ chk = gcnew array<int>(65);
```

array represents the category of the item declared. <int> represents the type of data that this array holds, chk denotes the name of the array, gcnew tells that a new item is being declared and (65) represents size of the array.

Note that this is not an ordinary declaration (int chk[50]) because this is a managed type CLR array.

Modularity:

We faced difficulties in modularizing the code, since the program deals with handling various features of UI, more than 150 labels are used in the program, and they all have to be handled in the same form class, as a form cannot be passed as an object to other classes.
However some amount of modularity was achieved through separating the path search module.
Shortest path module is handled by graph_node.h and graph_node.cpp.
graph_node.h is the header file which contains the declaration for dijkstra's algorithm and graphnode constructor.
graph_node.cpp contains the actual implementation of the graphnode constructor which initializes the required variables. Also printpath function is implemented.
There is a struct printpar defined containing the pointer for parent array for sending the data back into the form as parent array cannot be directly returned back into the form.

Files Included

Files included in this project are the images of all the different objects on the map of IIT-G like serpentine lake, Subansiri Hostel, Academic complex etc. All these images have to be included beforehand in the Resources folder.

Most importantly we also had to include the IIT-G detailed map.

Image sources: Google images and intranet.iitg.ernet.in

Software used

Microsoft visual studio professional 2013.

Language used:

Visual C++