University Name

Doctoral Thesis

Thesis Title

Author:
John Smith

Supervisor:
Dr. James Smith

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

Research Group Name
Department or School Name

January 2014

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Every person in his life at least once leaves his home city for any reason, whether it is a business trip or just a vacation. Taking into account that modern technologies nowadays allow people to move from one point to another in a few hours, such trips becomes much easier to perform. Moreover it gives an opportunity to investigate almost every point on the globe.

That is why maps become indispensable attribute of any trip. There are two main categories of maps: paper and electronic maps. Classical paper maps can help in any situation but in comparison with electronic map they have one big disadvantage – they are not interactive. Electronic maps in its turn are also separated on "online" and "offline" maps. "Online" maps are interactive, fast and usually not limited to one city. Furthermore there is more than one "online" map, so user can use any on his taste.

But what to do if for any reason there is no internet connection? For this case "offline" maps can be used, especially if person appears in unknown city. And the main problem of any map that covers large area in this case can be absence of any important information like presence of grocery store or bakery or laundry for small cities. That is why implementation of maps for small areas and cities is very important, because it contains specific information about this particular region.

So the goal of our project was implementation of the map for city Le Creusot with basic set of data and functions like finding way between two points or showing main public facilities. It can be useful for any person whether it's a tourist, visitor on a business trip or a student. Especially we hope that this map can be useful for next *ViBOT/MsCV* promotions for simplifying their life in Le Creusot.

Chapter 2

Project management

In this chapter we want to present how our group organized and managed this project. Group members are:

- Ozan
- Oksana
- Klemen
- Natalia

As we all had a bit of experiences in software design, we knew how important a proper plan and research before the actual implementation is. Unfortunately, we were also aware that no matter how good the plan is, we will be forced to adjust it during the implementation. Either because of the things we overlooked while planning or because some assumptions we made were wrong. We decided to follow the *iterative and incremental development model*, which enabled us to adjust our plans after each of the implementation iterations. A schematic representation of the iterative and incremental development model can be seen on figure 2.1.

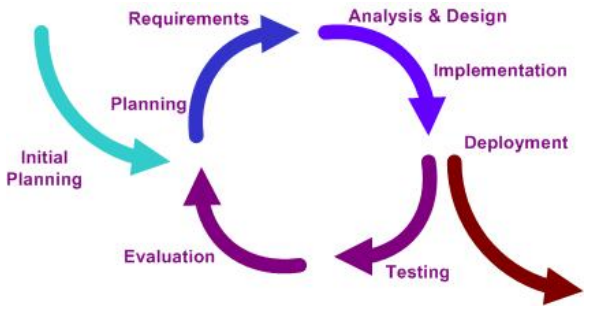

FIGURE 2.1: An iterative development model (image taken from [1])

2.1 Basic building blocks

On our first meeting we decided to split the project into four main parts to be able to fully manage the whole project at all times. This parts are:

- User interface
- Database and data structure
- Map representation
- Path algorithms

2.1.1 UI - User Interface

Part of the application that is responsible for user interaction with the application. Main goal is to make the interface as user-friendly as possible. In the beginning we made a rough sketch of how the interface should look (figure []), which has, as expected, changed during the process of later design and implementation. The UI is presented more in details in chapter [UI USER MAUNAL].

2.1.2 Database and data structure

Our database consists of two parts. The static part, having the information about the roads in Le Creusot (described in []) and the dynamic part with the information about the points of interest (POI). In chapter[] we address the facts we took into account while considering different database options and data structures.

2.1.3 Map representation

We separated map representation from other parts of user interface, because we think it is the most important of UI, so we will give special attention to it.

2.1.4 Path algorithms

In the different algorithms we adopted and developed, we do all the searching for paths, calculating distance and travel times and optimize the search results as much as possible. Our main goal is to make the algorithms work as quickly as possible, taking into account all the restrictions road networks has (oneway streets, footways etc.). We also want to

make the algorithms as reusable as possible, to reduce the redundancy in development and minimize the possibilities of errors.

2.2 Softwares used for project management

Github, skype etc.

2.3 Meetings

some bullshit about that

Chapter 3

Initial planning

3.1 User requirement analysis

As mentioned in the previous chapter, we decided to use *the iterative and incremental development model*. Before starting the iterations we spend quite a lot of time on the initial planning, with special attention to user and general project requirement analysis. From the project's instructions we were able to identify the following major user requirements and construct use case diagram (figure 3.1):

- User should be able to enter start and end point either by:
  - mouse click;
  - specifying latitude and longitude;
  - selecting a point of interest.
- Find shortest path from point A to B (by foot or by car);
- Find the shortest way to all POIs of a certain category in a radius from point A (by foot or by car);
- Construct an itinerary from point A to B with visiting POIs in between (with max distance limit);
- View, edit and add POIs

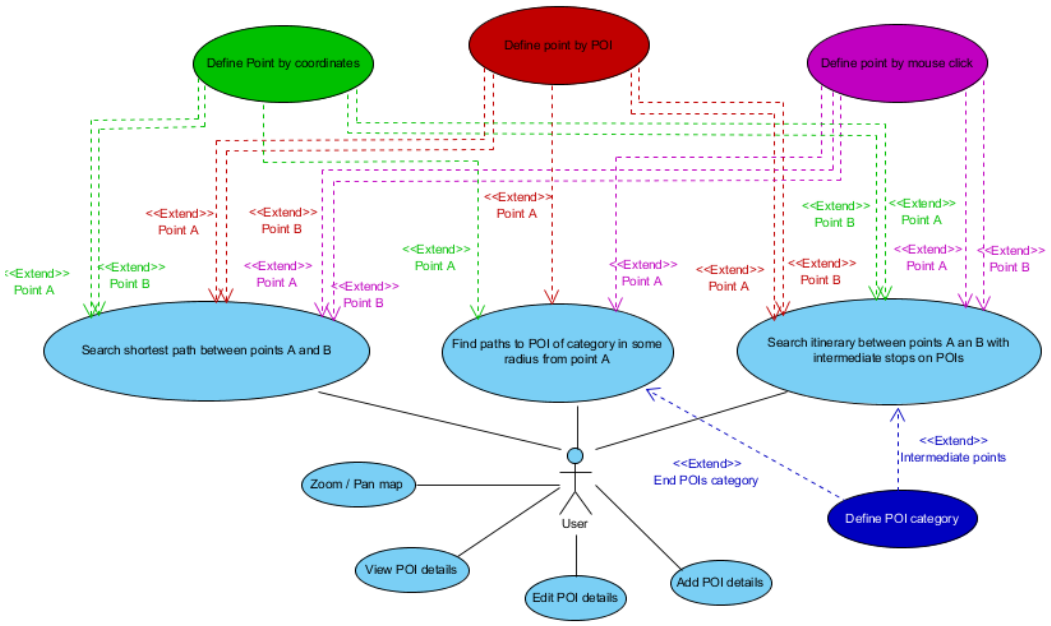

FIGURE 3.1: Use case diagram

3.2 Plan of implementation

After having the user requirements, we decided to make the global plan of implementation. We took into account the fact that we have to develop the applications in **C++** and **Matlab**. As described in section 2.1, we have already split the project into four main parts. Because the user interface and map representations require completely different approaches in C++ and Matlab, we decided to split them into two parallel projects, one almost independent from another. On the other hand, we decided to use the same algorithms for both projects. The main reason is the minimization of the redundancies in the research and development stage and reducing the possibility of errors. This way we could have the same algorithms for both, differing only in pure implementation details.

Each of was delegated to oversee one of the parts (as shown in figure []). We really want to emphasis that this does not mean each of us solely did that part or that is responsible for it. No matter the delegation, we all worked on all parts of the project, either in research, design or implementation stage, so the delegation was just for preparing the material for the meetings.

3.3 MAYBE TIME PLAN?

Chapter 4

Data sources

Our application uses two main types of data. **Static** map data describing the streets and buildings and **dynamic** data about the different points of interest (POIs). The difference between static and dynamic data lies in the users ability to add and edit POI, while the map information is not meant to be changed by the user.

The acquisition of this data was not the essence of the project, so we were allowed to use any data source, with appropriate licence and collaborate with other groups.

4.1 Map data - OpenStreetMap

Acquisition of map data (roads, road types, buildings etc.) can be extremely time consuming and can easily produce unreliable results. At the same time, there are a lot of different open source data sources available on-line. Together with the other groups we decided to use OpenStreetMap data by the OpenStreetMap Foundation ([3]).

OpenStreetMap is an open source project providing free map data of the world. Data is published under the *Open Database License (ODbL) from Open Data Commons (ODC)*, which enables us to freely produce the works from the database, modify, transform and build upon the database.

4.1.1 Data format

OpenStreetMap data consists of four core elements:

- **Nodes** - basic point of location with longitude and latitude information. They can be used to mark a single point (POI) or in a list of nodes (way, relation);

- **Ways** - ordered list of nodes, representing a street or an area like a lake, forest etc.;
- **Relations** - ordered list of nodes and ways which can be in a relation (many different roads can be part of a long motorway);
- **Tags** - key-value pairs which are used to store information about different objects (nodes, ways, relations).

Figure 7.1 shows the difference between different core elements

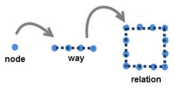

FIGURE 4.1: Representation of basic OSM elements (image taken from [2])

4.2 Points of Interest (POI)

One of the main requirements for our application is to enable user to search using different points of interest. For acquisition of this points, all of the groups again collaborated, each marking all the points in one part of Le Creusot. For each point we decided to acquire:

- name;
- location;
- address;
- photo.

In the end we also categorized all the points into 26 different categories.

Chapter 5

Database and data structure

5.1 Database

As described, our data consists of two parts. *Static* map data and *dynamic* points of interests data. For the beginning, we were deciding between using relational database and XML based database. With each of them having some benefits, we took the following facts intro account during the consideration:

- Most of the data is static - information about the roads in Le Creusot will not change;
- Dynamic data will rarely change - user will not often update information about the points of interest;
- Relatively small amount of data - Le Creusot is a small city, so there is not a lot of information about the roads. This gives us the opportunity to read all the information to memory at the beginning, reducing the latency caused by queries to the database;
- Easiness of install and mobility - using relational database, requires installation of different software (SQL server, connectors, etc.) on the clients computer, which we wanted to avoid.

In the end we decided to use **XML based database**. We kept the two parts of the data split into separate XML files. We kept the OpenStreetMap data in the OSM file and created another XML file for POI data. This way, we could easily change either part of the data without compromising the other.

5.1.1 Disadvantage

Using XML based database has one big disadvantage, which we have to take into account. XML based database is saved into a file. This does not provide the ability to easily update POI data. Unfortunately, every time we update this data, we have to rewrite the whole file. This can in some occasions be the source of problems, as the writing to the file can be disrupted or cancelled, making the file unusable. However, we do not anticipate the user to update the data frequently, so we took this compromise.

5.2 Data structure

We have designed out class structure with the A* search algorithm and OpenStreetMap (OSM) file structure in mind. We followed the OSM structure, having a basic class called **Node**. In figure 5.1 we show the different types of classes we use with a small graphic representation for better understanding.

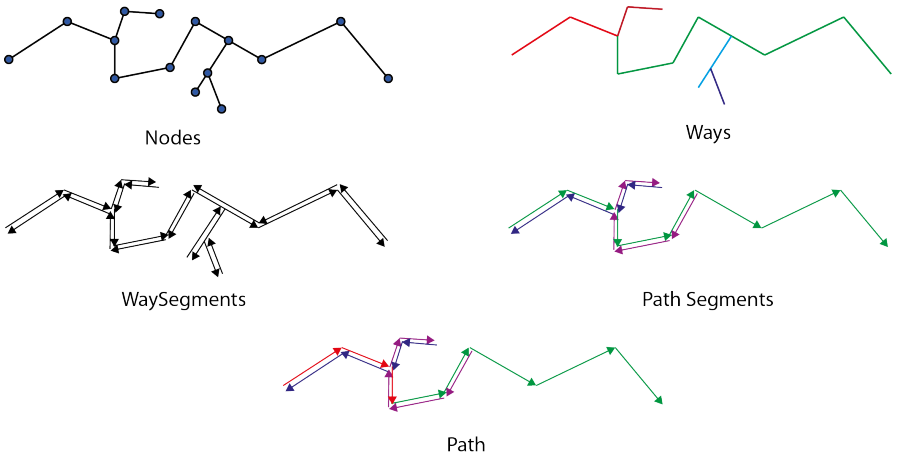

FIGURE 5.1: Different objects in our data structure

5.2.0.1 Node class

It is used to represent a single point on the map. This can either mark a point of interest (POI), or can be a part of a Relation such as *WaySegment, Building or Way.* It contains location information, some A* algorithm informations (weights, visited flags, etc.) and a vector of pointers of WaySegments to which we can traverse from this Node. This is important especially for A* algorithm, as it significantly reduce the time needed to find all possible and subsequently the correct next move.

5.2.0.2 Way class

Class implemented as in OSM file, to store the properties of each of the ways. It us usually used to represent whole streets, or part of the street that have the same properties. Each way is assigned a type (primary, secondary, etc.), direction (one-way, bidirectional) and access privileges (private or public).

5.2.0.3 Relation class

Relation is a base class from which we extend different relation classes (WaySegment, Building). It only contains a vector of pointers to Nodes that are in one specific relation and type of the relation.

5.2.0.4 WaySegment class

WaySegment is the main class for our path algorithms. WaySegment connects two nodes in a specific direction. This means that if the road between two nodes is bidirectional, we will create two WaySegment objects, one for each direction. We also have a pointer to an object Way, which belongs to the road traversing. This gives us the option of getting information about the road at any time.

5.2.0.5 Building class

Is a simple class containing all the nodes representing one building.

5.2.0.6 PathSegment class

It is a class that is used to store the result of a *shortest path search*. It contains all the pointers to WaySegment objects we need to traverse in order to get from point A to B.

5.2.0.7 Path class

Object that contains vector of PathSegment pointers. Path is the end result of any search. If the search is solely path from point A to B, the path is going to have only one PathSegment pointer. On the other hand, if we search for itinerary, Path will include multiple PathSegments, one for each pair of middle points.

5.2.0.8 Database class

Database is an main class for the whole database. It contains functions for correct parsing of XML files and also has containers *(std::map)* with pointers to each of the classes created (Node,Way,Bulding,WaySegment). We use this to quickly retrieve the classes based on their ID, and to be sure to delete all created objects in the destructor of the class. We also used a boost implementation of a *rtree* data structure to hold all the WaySegment objects to be able quickly do a spatial filtering. This is very useful in implementation of finding the close WaySegments.

In figure 5.2 we can see the whole class diagram of the database part of the application in C++.

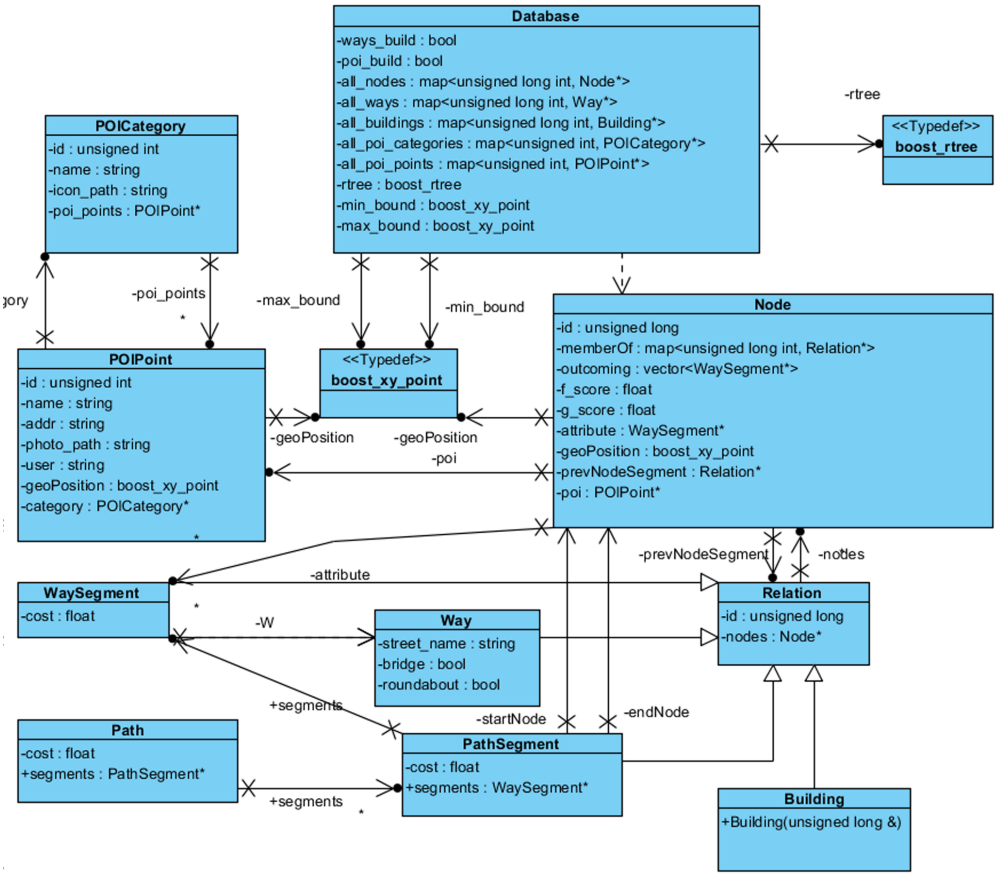

FIGURE 5.2: Class diagram

Chapter 6

GUI

6.1 C++

C++ Part

6.2 Matlab

Graphical user interface is main connection link between user and the program. By the correctly representation of data in GUI user can get all information that he needs. That is why it is so important to implement user interface that from the one hand will be easily understandable by user, and on the another hand can represent all possibilities of the program. In this chapter we will represent GUI for C++ and Matlab part, discuss its main features and difficulties.

6.2.1 MATLAB GUI

6.2.1.1 Creating GUI with GUIDE Layout Editor

For creation GUI in MATLAB the GUIDE Layout Editor was used. It allows design graphically user interface and then generates code that corresponds to each element of the user interface. GUI can be consists of standard elements like axes, toggle buttons, push buttons, panels and others.

Yomap MATLAB GUI is represented on figure 721 and consists from next elements:

- figure
- axes
- panels
- textboxes
- checkboxes
- toggle buttons
- push button
- static text
- pop-up menus

Here will be " Figure 721 – Yomap GUI representation in GUIDE Layout Editor". Natalia will provide.

The final result of GUI is represented on figure 722 and has such functionality:

- Zooming map in
- Zooming map out
- Pan map
- Plotting map layers
- Picking points on map with mouse click
- Choosing categories and points of interest by choosing from pop-up menu
- Choosing categories and points of interest with mouse click
- Enter name and address of the points of interest
- Enter coordinates of the points of interest
- Enable/disable middle point
- Hide/show search panel
- Show point of interest information
- Swapping points

- Choosing type of transport

Here will be "Figure 722 - Yomap GUI"

For zooming in, zooming out and pan map standard MATLAB functions were used. For others features callback functions were used. User is allowed to enter name or coordinates of point of interest manually or with mouse click. In case of typing all information into textboxes it is verified and in case of error corresponding message is appear. If user prefers use mouse click coordinates are checking on being in range of the map. Another feature of the Yomap GUI is hiding/showing search panel (figure 723). Because sizes of the map are not allowed to show map and search panel together on the screens with small resolutions we decided to hide a panel for better map representation. In case if button "Hide All" is pressed the search panel became invisible and panel with instructions moves to the bottom of graphic window to show the whole map. After pressing button again all panels appear back on their initial places.

Here will be "Figure 723 – Yomap GUI search panel"

For user convenience button "swap" was implemented (figure 724). In case of the same type of point of interest representation after pressing button "swap" all information between two points changes.

Here will be "Figure 724 – Yomap GUI "swap" button"

All possible information about points and way is represented in instruction panel (figure 725).

Here will be "Figure 725 – Yomap GUI instruction panel"

6.2.1.2 Map Layers Plotting

Map by itself is complicated graphical structure that contains luck of information. That is why one of the biggest problems can appear in map representation is its overloading with information. So to prevent this and for user convenience we decide to use layers for plotting map. Each layer represents only one type of information. In our GUI we have 4 main layers:

- Map Layer
- Roads Layer
- Way Layer

- Category Layer

Map layer (figure 726) is a *.png image of map of Le Creusot that loads in the beginning of the program.

Here will be "Figure 726 – Map layer"

Roads layer (figure 727) is a vector of location of nodes that we traverse in order to have way from point A to point B.

Here will be "Figure 727 – Roads layer"

Way layer (figure 728) is a vector that shows the shortest way from point A to point B.

Here will be "Figure 728 – Way layer"

Category layer (figure 729) is a representation of points of interest on the map.

Here will be "Figure 729 – Category layer"

Each layer can be shown either by itself either in a combination with others layers. Because MATLAB allows draw any graphics on each other in a row the function "map_Show_Result" was implemented. Any time if one of the layers buttons is pressed the function is called and conditions of all layers buttons are sent to it. Before any layer appeared function "prepareMap" is called. It allows preparing axes before drawing layers and after this depending on buttons value corresponding layer is drawn. Way layer can be drawn only in case if route between point A and B is found.

6.2.1.3 Mouse Click And Dots Plotting

As was already mentioned user is allowed to pick any point on the map using mouse. To pick any point MATLAB functions "WindowButtonDownFcn" and "get(handles.mapDraw,'currentpoint
are used. Result of these functions returns coordinates of the axes at the place where mouse button was pressed (figure 7210).

Here will be "Figure 7210 – Mouse click"

In case if user chooses "show on map" as a point representation then he allows to take any point on the map even if "show category" button pressed. In case if "show category" button pressed but user choose "category search" or "search by enter" then he can pick only category points, otherwise point could not be plotting and data could not be saved. Any points that picked on the map are checked on being not out of range (figure 7211).

Here will be "Figure 7211 – Out of range checking"

Dots' plotting after mouse click is the most difficult and "unstable" layer. The problem is in plotting dots for every point with saving map layers. We should always take into account how many points do we have, which of them are already plotted, which we need to plot and etc... To solve this problem we used flag and handle to each point. Flag allows us to check if point already plotted or not and handle is used for deleting/plotting dot. For plotting dots function "draw_point" was implemented. To avoid problems with plotting points and layers algorithm that shown on figure 7212 was developed. It allows to draw any point on any layer independently.

Here will be "Figure 7212 – Dots plotting algorithm"

Chapter 7

C++ vs MATLAB

First of all the main difference between C++ and MATLAB is its cost. There are a lot of free and commercial implementations of C++ for different platforms. MATLAB on its term is only in charge. Of course student version of MATLAB exists, but it does not have some of toolboxes that can be very useful.

Beside this as we repeatedly noticed MATLAB has extremely high CPU and memory usage in comparison with C++. So it takes a lot of time for huge data calculation.

From other point of view MATLAB is more flexible tool. Even if it takes more resources for data calculation representation of the data is more "free". It is much easier to work with matrices, strings and other data in MATLAB than in C++. There is no need in variables declaration in MATLAB like in C++, we can use any variable in any part of program without caring of its type.

Another MATLAB particularity is its global variables. In C++ global variable that declared in the beginning of the program became global for all program cycle. In MATLAB declared global variable is global only inside function. To use it in another function we need also declare it as a global in this function.

One more difference between MATLAB and C++ is a huge amount of toolboxes in MATLAB. It allows to make different experiments without big effort. For example plotting graphs, made difficult calculations.

Other difference is graphical user interface creation. C++ provides an extensive choice of methods and tools for GUI implementation. MATLAB in its turn has limited set of tools for this purpose, that does not give such big opportunities like in C++.

Thus we can conclude that both C++ and MATALB are powerful tools, but must take into account that they should be chosen correctly depends on the existing problem.

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

[1] Iterative and incremental development. 2014. URL http://en.wikipedia.org/wiki/Iterative_and_incremental_development.

[2] Openstreetmap data overview. 2014. URL http://wiki.snowflakesoftware.com/display/GLDOC/OpenStreetMap+Data+Overview.

[3] Openstreetmap fundation. 2014. URL http://wiki.osmfoundation.org/wiki/Main_Page.