MapWars: Location-Aware Multi-Player Mobile Game

Final Report for CS39440 Major Project

Author: Luke Ward (luw9@aber.ac.uk)
Supervisor: Reyer Zwiggelaar (rzz@aber.ac.uk)

25th March 2013
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G400)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature .....

Date .....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature .....

Date .....

Abstract

MapWars is the culmination of an investigation into the feasibility of creating a location-aware, multi-player real-time strategy (RTS) game for mobile platforms. Combing strategic game play against multiple players in a persistent environment, where game play is based on map of Earth and users are restricted in game to their real world location. Giving each user the capability to create their own army and battle against other players within their local area. Consideration while protecting each users privacy.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Overview

1.1 Background

RTS games have a huge market on desktop environments, but have yet to make the break through into both the console and the mobile gaming markets. This is mostly attributed to the complex control mechanisms that need to be executed precisely. A mobile devices form factor restricts the number of controls that can be presented to the user at any given time as well as the precision in which these commands can be issued.

RTS games are generally designed around large, expansive maps that cover a large range of environments and landscapes. The game map shapes how the game will be played, how involved the player feels and ultimately the engagement of players. These maps can take years to develop and result in one of the largest costs within the development process. So when coming up with an environment why not take advantage of a ready made one? That of planet Earth.

Building a game where the game play takes place on a map of Earth has a number of benefits. First is the scale of a map covering 510 million square kilometers of varying terrain and features. This scale also contains a huge level of detail that could not be achieved by a team of designers. Along side this there is the feeling of familiarity, unlike when starting a new game and having to teach the player about the environment, the user will already have a well formed model in their own head. Along with highly detailed knowledge about certain areas especially those within close proximity of their current location.

Mobile devices have a whole host of unique features and sensors that set it apart from other gaming platforms. One feature that has been widely adopted in a huge range of different applications is location. A users location can be easily determined with a number of different components present on most modern smart phones, these are a GPS chip and the mobile GSM network. Although location has been used extensively in applications it has yet to be utilized effectively as a key metric within a game. This extra information about the user would enable a game set around the physical world to be able to be integrated with the user. Instead of placing new users randomly on a unfamiliar map they can be placed in their real location. Thus giving them a chance to use their own knowledge of their surroundings to help them within the game environment.

1.1.1 Market

A worldwide RTS game that combines a map of the world and data gathered about the user combine to create a unique gaming experience and could add an entirely new dimension to the genre. This can also reduce the learning curve and thus shorten the time between installation and engagement. With the sheer number of applications available on a mobile platform combined with the

ease and minimal to no cost of installing new applications, this is important to reduce the chance of the user simply finding an alternative.

1.1.2 Platforms

The Google Android smartphone platform, as of February 2013, had over 51% market share in the US [2]. Closely followed by the original and most established app ecosystem of Apple iOS. These two platforms are the most appealing platforms for mobile application developers.

1.2 Objectives

MapWars is going to attempt to combine the the tactical aspects of these systems with a mobile experience tailored to the users surroundings. Adding in multiplayer gameplay will allow each user to interact, attack and defend against enemies as well as affect the landscape they are playing in. Taking as much of the the tactical aspects of other systems with a higher emphasise on resource gathering and long term tactical game play. Try to remove some of the fast paced rushing game play that can be adopted in some RTS games. Rushing is a tactic in which a team will attack an opponent as quickly as possible as to surprise and overwhelm them. Hoping that the opponent will not have create adequate defences to counter the attack. One of MapWars main focuses in regard to strategy and game play is to have on continuous game, with many opponents. This will encourage players to concentrate on gathering resources and fortifying their base before they can have the fire-power to take on an enemies forces. As resource gathering is a major aspect of the game play, time permitting, I intend to position these resources based on real world locations. For example allowing users to gather wood or other building materials from woodland and forests that exist in the real world. Resource will deplete as users remove them but they will have a fairly short time scale in which they replenish.

Chapter 2

Development Process

Due to a strict deadline and extensive possibilities the project offered it is important to choose a development life cycle that could both quick and flexible. It needed to be able to allow for rapid development in a controlled manor that would reduce the need for rewriting and refactoring of code.

My development method will consist of taking the project and breaking it into smaller segments that can be tackled individually. Each segment will contain a planning, development and analysis stage. The development will take place iteratively building on prototypes to produce the final component. This will avoid making throwaway prototypes and cut down on development time. These segments will build upon the ones before, building up a complete working application. This combines different aspects from both the Spiral and Rapid application development (R.A.D) methodologies.

A weekly plan has been set out that is currently up until mid-January 2013. It covers the major aspects and milestones of the project as well due dates for each hand in. At the time of writing the first two sections, mapping solutions and location services have both been completed comfortably within time. The next stage is to research, prototype different client-server communication methods. Writing up my findings and implementing the basis of my chosen method. After getting the basic client-server interactions working it will be a case of adding features to each and adding the required supporting system to the other. Adding features until a basic working prototype is ready for testing which I plan to be at the end of November during the last week of term. This basic prototype aims to locate each user and display their location on every users device. It will be an important test on such things as responsiveness, battery usage and server stability and load. Testing will take the form of distributing my test application to a number of testers who will spread out and walk around the local area. Detailed logs will be take on each device storing different hardware events, such as GPS updates and battery statuses, as well as all sent and received data from the server. Also the server and proxy will log all the connections received and responded to as well as their loads at various intervals.

2.1 Introduction

The development life cycle chosen is that of Rapid Application Development (RAD). This process reduces the need to have a detailed specification or design at the beginning of development. Instead specification, design and implementation are all simultaneous. This life cycle follows more closely to the Spiral Model than that of a more traditional waterfall type life cycle. The Spiral Model, as defined by Barry Boehm [1], depicts an evolutionary style of development which still keeps control

over the project. Unlike the waterfall model which requires a stringent and detailed design that needs to be followed throughout implementation. Only the highest-priority features are considered for each iteration. The chosen feature is defined, designed and implemented during the cycle. Each cycle results in a prototype of a portion of the final system that can be tested. From this prototype ideas can be tweaked and changed then defined and implemented. Then the next feature can be defined and so on until the desired system is completed.

The RAD life cycle is a further streamlining of the spiral model and only has four distinct stages. Firstly there is the requirements gathering stage where all parties agree on the scope and requirements for the overall project. This is followed by user design where users interact with the developers and create a set of prototypes that are then evaluated by the users. This step is continuous phase that sees the prototypes change and adapt to the users requirements. This phase works in tandem with the construction phase which sees the prototypes be integrated into the final application and tested. With the user still actively involved they can suggest changes and improvements as the application takes shape. After the application is completed the final stage sees its testing, integration and user training.

2.2 Modifications

The spiral and RAD models are defined as an evolutionary method therefore each iteration prototype should be delivered to the stakeholders. These stakeholders can then give their feedback which should be utilized in the next cycle. Seeing as this project does not have any stakeholders this crucial step can not be completed and therefore the model needed to be adapted from a evolutionary to an incremental one. To achieve this the stages were mapped out before implementation began resulting in a slightly increased amount of design. These stages were purposefully left as broad as possible to accommodate necessary changes that would present themselves as development began.

As well as removing the of reworking code by cutting out the feedback loop for each prototype, the user design and construction phases are combined. This single step would see prototypes being worked directly into the current code base. In this way extra development time would not need to be used to integrate the prototype back into the master branch. This combination by itself would slowly reduce the code quality seen across the application as ideas are tested, for this reason spike work is encouraged to be carried out before each iteration. These short spike work sessions should show up any problems and highlight more efficient ways that part of code could be implemented. This will then carry over into the main code when the feature is implemented. It is important to keep this spike work short and focused and full solutions are not the ideal outcome from it.

These modifications were made as a way to streamline development, reducing time spent in design and coding while not reducing code quality or flexibility.

2.3 Development Environment

Development on the Android platform restricts development to Java using the Android SDK. It was decided to use the Eclipse IDE as the main editor partly due to it's popularity and abundance of information and also it's improved integration with the SDK. Google provide an Eclipse plugin called Android Development Tools (ADT) which provide a set of tools to streamline the development process.

The server portion of the project was written in Python this was mainly to speed up development and minimize unforeseen problems. Python is a general purpose high-level programming

language that has a small learning curve and generally produces concise code.

Chapter 3

Design

3.1 Overall Architecture

3.2 Mapping

The central component of the application was going to be a map. For this reason it was vitally important that an appropriate mapping solution was used.

3.2.1 Google

Google provides a simple, easy to use interface to their own maps making it the obvious choice for any Android application. Their maps are accurate, up-to-date and very detailed.

Google Maps were used for early prototype development.

Unfortunately there are a number of restrictions in place stopping their use in a number of situations. The most relevant of which is that they can not be used in an application that is not freely available to the public. Therefore restricting it's use in a paid-for application, such as MapWars may become. As the future of the application is uncertain it seemed desirable to steer clear of as many possible restrictions as possible. For this reason it was important to find a comparable alternative.

3.2.2 OpenStreet Map

OpenStreet map is an INSERT DESCRIPTION OF OSM HERE. It's growing popularity means that INSERT STATS ABOUT AREAS COVERED. With an acceptable level of detail combined with it's open SOMETHING(ethos?) made it the next most obvious source.

OSM has an API that allows it to be easily embedded into webpages but no native android SDK. A number of 3rd party libraries are available. The most complete and popular is that provided by MapQuest.

MapQuest are a mapping company that combines proprietary data and OSM data to create their own maps. They offer an Android SDK that gives you the option of which tile source to use. There are obviously restrictions to the proprietary data but if you opt for the free tiles then the same license is used as with OSM. The Android SDK available was designed to mirror the API available for the Google Map SDK. This made swapping out the Google Maps code and replacing it with the MapQuest code was trivial and problem free.

At the point in time of implementing MapQuest the design had called for the option to switch between satellite and road maps. MapQuest's main drawback, and more widely OSM itself, was

it's lack of detail. The level of zoom supported was a number of levels less than that of Google Maps. These extra zoom levels would have made unit manipulation easier on smaller devices. Satellite images were the main concern as they were not available at the level of zoom required to make game play comfortable.

MapQuest was used as the mapping solution for a large portion of development and offered a stable platform. Once more of the functionality was in place user testing presented a number of problems with the map tiles being used. Most significant of which was a difficulty in being able to locate units among the details presented with the map. The sprites and colours being used to represent units were experimented with but none were clearly visible. The problem was with the design of the tiles being used and not necessarily the zoom levels present, although this may have helped alleviate the problems.

3.2.3 MapBox

One option available was to use a tile creator and host the map tiles on a server. This would be a costly and difficult solution to the problem. Hosting tiles is not a trivial task and require large amounts of storage and bandwidth.

MapBox offer beautiful hosted tiles. They also have their own software called TileMill which allows the creation of bespoke tiles based on any data source which can then be hosted and distributed via their network. TileMill was based on a CSS style syntax allowing you to customise any visual aspect, from line widths, colours, strokes. It also had the ability to import data from any source giving the ability to build up rich tiles with as much detail as required. For MapWars only the most basic detail was required while using a simple colour pallet. The idea was to make any unit stand out against the map while still presenting all the information required to orientate the user with their surroundings.

Tiles could be loaded from MapBox using a standard URI syntax used by the most tile vendors. This allowed it to integrate easily into any mapping framework.All that was needed was an SDK that allowed custom tile sources. Such functionality was found in OSMDroid. Like with MapQuest, OSMDroid followed the same pattern as Google Maps allowing it to be easily placed into the application without only one substantial problem. OSMDroid was missing one function that was supported by both Google Maps and MapQuest. These function was key in selecting units so had to be reimplemented ... which was not difficult but took time. Assumption was made it would be as effortless as the previous transition. After integration was complete plugging in the URI to my generated tiles was simple and worked straight of the bat.

MapBox did not offer satellite imagery but the beauty and simplicity of the maps being used made up for this. It was also decided that the complexity of such maps would just present the same images as found with the default OSM tiles. Satellite images could always be added to OSMDroid by simply finding a tile source and using that and would have no affect on the functionality of the application.

3.3 User Interface

When developing any application for a small form factor it is important to consider how the user interface can be minimized, and to not clutter the display making precise controls difficult. This is especially important when trying to fit all the functionality of a RTS game within the small form factor of a mobile device. To get around this problem only the essential controls will be included with only a portion of these being accessible from the main game screen.

Figure 3.1: Insert design of game map

Figure 3.2: Insert design of game map menu

Figure 3.2

Chapter 4

Implementation

The project was split into two entirely separate parts, the server and the client.

4.1 Client Implementation

4.1.1 Mapping & Location

The initial investigation into both mapping and determining the users location.

4.1.2 Communication

4.1.3 Units

4.2 Server Implementation

4.2.1 Echo

4.2.2 Units

4.2.3 AI

Chapter 5

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Have you tested your system on 'real users'? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

5.1 Overall Approach to Testing

5.2 Automated Testing

5.2.1 Unit Tests

5.2.2 User Interface Testing

5.2.3 Stress Testing

5.2.4 Other types of testing

5.3 Integration Testing

5.4 User Testing

Chapter 6

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

The critical evaluation can sometimes be the weakest aspect of most project dissertations. We will discuss this in a future lecture and there are some additional points raised on the project website.

Appendices

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

Annotated Bibliography

[1] B. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[2] comScore, "comScore Reports February 2013 U.S. Smartphone Subscriber Market Share," http://www.comscore.com/Insights/Press_Releases/2013/4/comScore_Reports_February_2013_U.S._Smartphone_Subscriber_Market_Share, 2013.

This is my annotation. I should add in a description here.

[3] MapQuest, "Android Maps API: Developer's Guide," http://developer.mapquest.com/web/products/featured/android-maps-api/documentation.

This is my annotation. I should add in a description here.