Browser-based Online Multiplayer Roleplaying Game

Final Report for CS39440 Major Project

Author: David Field (dvf9@aber.ac.uk)

Supervisor: Dr. Hannah Dee (hmd1@aber.ac.uk)

April 16, 2014

Version: 0.1.1 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G401)

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 .....

Acknowledgements

I am grateful to...

I'd like to thank...

Abstract

Include an abstract for your project. This should be no more than 300 words.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

Games are interesting projects to take on; they have a history of being difficult to make and pushing technology to its limits. There are a great many systems and features that can be included in them—AI, physics, graphics, audio, UI, multiplayer and more—and a great many ways to implement each, from simple to very complicated depending on the needs of the project.

Games also have a history of aiming for too many of these features in too little time. In most cases, every one of the features thought up would enhance the final product in some way, from a significant improvement that changes the way the game is played to a minor enhancement that makes things just a little more pleasant for the user. Many of these features may be considered mandatory for the game to be worth making at all. For example, a single-player chess game would probably not be very good if there were no AI to play against.

It is not just players who require features, however; developers of games need tools to implement the game design and a good engine to hang the design off of. A lot of game projects build custom tools that let developers and designers implement things quickly. Many game engines even provide methods for scripting and modding them after their release to players.

It is clear that, given the sheer enormity of the possible things that can be put into any one game, there is not enough time in this project to implement even half of them without a great deal of previous experience and skill. Every one of the major systems mentioned can be extremely complicated, requiring a lot of research, time and effort to make them work.

This chapter will discuss what the project is, why it was worth taking on and how a minimal system was devised that would satisfy enough of the game design requirements to be playable but also be implementable in the time given.

1.1 Background

1.1.1 The Project

Before discussing the decisions made about what was doable and why it was interesting, it's useful to know what the project actually is. A full account of the original, idealised, game design can be found in Appendix C.

The name of the project—*Browser-based Online Multiplayer Roleplaying Game*—gives a relatively good hint as to what the game is. "Browser-based" and "multiplayer" are fairly self-evident in meaning: multiple people play together in a game hosted in the browser. "Roleplaying game" is more ambiguous. In this case, it refers to a game in the style of the classic tabletop roleplaying game *Dungeons & Dragons.*

In the context of the project that meant the following things: Firstly, there needed to be two types of players—regular player and Game Master. A regular player plays the game as a character inhabiting the world they happen to be in. For example, they may be a dwarf in a fantasy kingdom, or a space marine on a futuristic space station.

The Game Master is a player responsible for building the world, telling the story and controlling characters that aren't controlled by the players (known as Non-Player Characters or NPCs). Traditionally, the Game Master would also be responsible for enforcing the rules of the world. However, in this project the game was to be responsible for that instead. The Game Master could, however, override or change the rules if he or she wished to do so.

Combat is the most obvious area where the game enforcing the rules comes into effect, as tradtionally combat is the strictest area of tabletop gameplay. Combat is turn-based, with players put onto a grid and given limits on the distance they can move and number of actions they can perform in each turn. When they attempt to do something—such as attack another character or creature or escape from a trap—they have to roll dice, the result of which decides whether they were successful or not, and often how well they succeeded or failed. As an example of the last part, a player failing to attack a creature with their sword could simply miss, or they could throw the sword away accidentally, depending on how badly they failed.

The project design specification called for interactions outside of combat too. For example, a player might be faced with a locked door. To get through a player could attempt to use a key they found. Alternatively, they could attempt to bash the door open with an item, such as an axe, or even their bare hands.

Given that the majority of the game is players—be they regular or Game Master—interacting with each other it was important to allow players to communciate effectively. To this end, a chat system was specified.

How the world is presented to players is important too. Given the style of game a simple textual system may well work. However, in this case graphics were asked for. More specifically, the graphics needed to be 2D isometric tiles.

Finally, there were some technical specifications as well. Firstly, the game was to be run in a

browser. Secondly, the game is multiplayer and needed a server, for which Python was chosen as the language with the goal of gaining experience in it.

1.1.2 Why make the game?

It is important to answer why the project was worth doing, particularly as it was a student-suggested one.

The first answer to this is that games are interesting in general. Most obviously, the final product of a game is (hopefully) something fun to play with appeal to a wide range of people. More relevant to the context of a project, however, is that games are interesting from a software perspective.

Games are made up of a lot of different parts, each one potentially being difficult to implement by itself. In this game, the most challenging individual parts are graphics and multiplayer. More important than just the individual parts, however, is making sure they fit together and work properly. In most cases the game needs to share data between these different places—game logic needs to know what an object is doing so it can perform game functions on it; the renderer needs to know what the object is doing so that it can be drawn to the screen correctly; the networking part needs to know what the object is doing so that it can forward on any relevant information to the server.

The game offers a lot of extendability. Given more time, more features can be added. Each feature, and fitting it together, offers a lot of potential for learning as well. For example, an extra feature could be AI, which is an interesting area in itself that offers a lot of opportunity to learn something new.

1.1.3 Other Games

It is useful to take note of other games in the space that this project inhabits, both to see whether the project is worth doing and whether there is anything to be learned from the work of others.

1.2 Analysis

With the knowledge that the scope of games can expand to encompass a ridiculous area, and that everything within that scope is likely to be time consuming to implement, it is important to do some research into each feature that one would like to include.

Taking into account the problem and what you learned from the background work, what was your analysis of the problem? How did your analysis help to decompose the problem into the main tasks that you would undertake? Were there alternative approaches? Why did you choose one approach compared to the alternatives?

There should be a clear statement of the objectives of the work, which you will evaluate at the end of the work.

In most cases, the agreed objectives or requirements will be the result of a compromise between

what would ideally have been produced and what was felt to be possible in the time available. A discussion of the process of arriving at the final list is usually appropriate.

1.3 Process

You need to describe briefly the life cycle model or research method that you used. You do not need to write about all of the different process models that you are aware of. Focus on the process model that you have used. It is possible that you needed to adapt an existing process model to suit your project; clearly identify what you used and how you adapted it for your needs.

Chapter 2

Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

2.1 Overall Architecture

2.2 Some detailed design

2.2.1 Even more detail

2.3 User Interface

2.4 Other relevant sections

Chapter 3

Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

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.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

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.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

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.

As an example, you might include a definition such as:

Apache POI library The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client s existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [2]. The library is released using the Apache License [1]. This library was used without modification.

Appendix B

Code samples

2.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [11].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*-------------------------------------------------*/
  /* Minimum Standard Random Number Generator        */
  /* Taken from Numerical recipies in C              */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity   */
  /* Always call with negative number to initialise  */
  /*-------------------------------------------------*/
```

```
int j;
long k;
static long idum2=123456789;
static long iy=0;
static long iv[NTAB];
double temp;

if (*idum <=0)
{
  if (-(*idum) < 1)
  {
    *idum = 1;
  }else
  {
    *idum = -(*idum);
  }
  idum2=(*idum);
  for (j=NTAB+7;j>=0;j--)
  {
    k = (*idum)/IQ1;
    *idum = IA1 *(*idum-k*IQ1) - IR1*k;
    if (*idum < 0)
    {
      *idum += IM1;
    }
    if (j < NTAB)
    {
      iv[j] = *idum;
    }
  }
  iy = iv[0];
}
k = (*idum)/IQ1;
*idum = IA1*(*idum-k*IQ1) - IR1*k;
if (*idum < 0)
{
  *idum += IM1;
}
k = (idum2)/IQ2;
idum2 = IA2*(idum2-k*IQ2) - IR2*k;
if (idum2 < 0)
{
```

```
    idum2 += IM2;
  }
  j = iy/NDIV;
  iy=iv[j] - idum2;
  iv[j] = *idum;
  if (iy < 1)
  {
    iy += IMM1;
  }
  if ((temp=AM*iy) > RNMX)
  {
    return RNMX;
  }else
  {
    return temp;
  }
}
```

Appendix C

Original Proposal

An online, multiplayer roleplaying game similar to *Dungeons & Dragons* hosted in the browser. The game will have a Game Master and several players in a group. The players move around the world and interact with it and the people/creatures in it and the Game Master guides the players, sets rules and manages the experience. They will be able to manipulate the game world and do things players can't do. For example, they could teleport a player to another location, spawn new monsters or change the weather.

It will include isometric graphics and aim to be as permissive as possible in what players are allowed to do (for example, interact with the environment using items they have, such as burning down a house with a torch). This means that there will be several systems, including combat, movement, character/creature creation, interacting with other characters (players and non-players, with chat as well as game system rules) and the environment (things like objects having heat and a burning point, in the fire example).

The nature of the design means that the project is open-ended, allowing for a basic game with a few simple systems and a single dungeon and character type to a game with several systems, full environmental interaction, a map and character editor and the ability to upload your own artwork.

It will require a server to manage the clients and let them interact, relay chat and act as an authority in the game world to keep clients in sync. It will also have to keep track of where players are in the world and whether they can see and interact with each other. There will be a graphical front end client that the players use and, in theory, multiple different clients could be made, such as a mobile app.

Appendix D

Project Outline Specification

4.1 Project description

An online, multiplayer game based in the browser, utilising HTML5 canvas and JavaScript for the client and Python for the server. It will be played similarly to the table-top roleplaying game *Dungeons & Dragons*, with a Game Master and several players.

The players will interact with the world in two modes or states. The first is a free-roaming state that operates in real-time. In this state, players will be allowed to interact with the world and the objects and characters within it with relative freedom. The rules that govern them will be loose and mostly describe the interactions of objects with other objects. For example, a magic spell that creates fire would have a heat attribute, and a door made of wood would have a burning point attribute. When they met, if the heat was greater than the burning point, the door would be set on fire.

The other state is the combat state. Players in this state will be locked into a turn-based system where they can only move and perform actions in a limited amount during their turn. Opponents can be creatures or other characters—both player and non-player—and will take their own turns

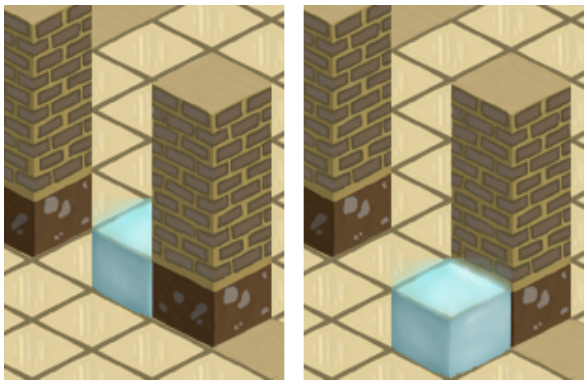

Figure D.1: This image shows a very early build of the game with 2D isometric graphics. The blue block represents a player. On the left, the player is obscured by an object; on the right, the player is in front of it.

to perform actions. Non-player characters and creatures will be operated by the Game Master. All characters and creatures will have attributes, such as health and mana (for magic), as well as a set of abilities that they can perform. If a character or creature reaches zero health it will have been killed and removed as an active element.

Each player will operate a single character and their view of the world and abilities within it will be defined by that character's location and abilities. If a character is too far away to see another character, then the player will also not see the other character. If a character cannot use magic spells, then the player will not be able to use them.

Player characters who are killed, either in battle or in some other way (perhaps they are killed by falling rocks in the free-roam state) are no longer playable. Players who lose their character may be removed from the game, become a spectator, be given an already existing character previously controlled by the Game Master or be allowed to make a new character.

The Game Master is not a player but rather the controller of the world. They will be able to interact with the world without limitations and be responsible for creating the world via a map editor, guiding the players around it, operating non-player characters and creatures in the world and set up events for the players. The Game Master will even be able to override the normal rules of the world. In the example of the fire spell and the door, the Game Master will be able to say that the door is not set on fire, even if it otherwise would have been.

The world will be presented to users using tile-based, 2D isometric graphics (as seen in Figure D.1). It will consist of a planar terrain with objects, items and characters on top of it. Movement in the world will be done in 8 directions: up, down, left, right and diagonal. Characters and creatures will move from tile to tile, with only one able to be in a tile at any given time. However, each tile can hold many items (such as weapons, money, clothes). Objects will be varied, with those such as walls and pillars taking up a tile by themselves but objects such as chairs or chests of treasure that can be interacted with by characters may coexist in a tile with characters.

The players and Game Master will be able to communicate throught a textual chat system. In its most basic form, this will be a global chat that all users in the game can see. However, the ability to restrict chat to a local context or to an individual user would be nice.

Extra features that would enhance the project but are not mandatory include: voice chat system; multi-levelled maps with varying heights; random map generator; random creature/character generator; visual representation for character items (such as weapons, clothes, etc); and the ability for users to upload their own artwork for use in their games.

4.2 Proposed tasks

My proposed tasks to achieve a basic version of the project are as follows:

- Create basic client-side graphics engine, allowing a map to be rendered and a player to move around.

- Create basic multiplayer functionality, setting up the server and allowing multiple players to exist in the same map and move around within it.
- Add Game Master, who can select different characters to control.
- Add items and attributes to characters, allowing them to carry things and setting things up for the next task.
- Add combat state, allowing users to do more than walk around the world.
- Add attribute interactions in the free-roam state, allowing for the fire spell and wooden door interaction.
- Add map editor for the Game Master.

4.3 Project deliverables

Game Client, final 'production' version. This is what the users of the game will interact with, via a browser, allowing for both regular players and a Game Master, who is given the ability to create maps for the game.

Game Server, final 'production' version. This will be responsible for syncing the game between all the players and providing authoritative state to clients to help prevent cheating.

Documentation. Basic guides for users that instruct them on how to operate the game from the Game Master and player perspectives.

Final Report. Report detailing the system; the process of the system's creation from beginning to end; differences between the proposal and final system and explanations for those differences; full bibliography.

Annotated Bibliography

[1] Apache Software Foundation, "Apache License, Version 2.0," http://www.apache.org/licenses/LICENSE-2.0, 2004.

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

[2] ——, "Apache POI - the Java API for Microsoft Documents," http://poi.apache.org, 2014.

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

[3] J. Bose, *Creating Isometric Worlds: A Primer for Game Developers*, May 2013. [Online]. Available: http://gamedevelopment.tutsplus.com/tutorials/creating-isometric-worlds-a-primer-for-game-developers--gamedev-6511

Tutorial series used to understand the basics of rendering isometric graphics.

[4] H. M. Dee and D. C. Hogg, "Navigational strategies in behaviour modelling," *Artificial Intelligence*, vol. 173(2), pp. 329–342, 2009.

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

[5] S. Duckworth, "A picture of a kitten at Hellifield Peel," http://www.geograph.org.uk/photo/640959, 2007, copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

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

[6] Mozilla, L. Workshop, F. Lecollinet, and G. Lecollinet, "BrowserQuest Source Code," GitHub, 2012. [Online]. Available: http://github.com/mozilla/BrowserQuest

BrowserQuest source code.

[7] ——, "Mozilla's BrowserQuest Game," 2012. [Online]. Available: http://browserquest.mozilla.org/

The BrowserQuest game itself.

[8] M. Neal, J. Feyereisl, R. Rascunà, and X. Wang, "Don't touch me, I'm fine: Robot autonomy using an artificial innate immune system," in *Proceedings of the 5th International Conference on Artificial Immune Systems*. Springer, 2006, pp. 349–361.

This paper...

[9] B. Nystrom. Game Programming Patterns. Online. [Online]. Available: http://gameprogrammingpatterns.com/index.html

Online book detailing common game programming patterns, as well as examples on how to use more general design patterns in the context of games.

[10] N. G. Obbink, *Javascript tile engine tutorials*, 2012. [Online]. Available: http://nielsgrootobbink.com/wokflok/jte/

Series of tutorials going through the creation of a simple RPG in JavaScript.

[11] W. Press *et al.*, *Numerical recipes in C.* Cambridge University Press Cambridge, 1992, pp. 349–361.

This is my annotation. I can add in comments that are in **bold** and *italics and then other content.*

[12] Various, "Fail blog," http://www.failblog.org/, Aug. 2011, accessed August 2011.

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

[13] L. Workshop. (2012) Little Workshop Information on BrowserQuest. [Online]. Available: http://www.littleworkshop.fr/browserquest.html

Web page by the authors of the BrowserQuest game giving some information on it.