G52GRP Interim Report

EduCraft

Minecraft as a Collaborative Learning Tool

Group `gp13-pxb1`:
Tosin Afolabi (`ooa02u`)
Janos Bana (`jxb12u`)
Eze Benson (`exb02u`)
Ali Kerr (`axk02u`)
Ian Knight (`iak12u`)
Ying Wang (`yxw03u`)

March 31, 2014

Contents

Chapter 1

Introduction and Background

The EduCraft project is concerned with developing a series of extensions[1] for the popular computer game Minecraft, aimed at promoting collaborative learning in primary schools, particularly with reference to numeracy education. In this first section, some of the advantages of collaborative learning are explored in contrast to conventional teaching methods, and the game of Minecraft is introduced in the context of education.

1.1 Collaborative learning

> "Group interaction allows students to negotiate meanings, to express themselves in the language of the subject and to establish a more intimate and dialectical contact with academic and teaching staff than formal methods permit." [4, p. 1]

Group work has long been recognised as a key component in any form of successful education—it enables students to become more involved in their own work, and to engage with it in a different manner from what is normally seen in classroms. Johnson and Johnson observe that "The human species seems to have a *cooperation imperative*", and that cooperation plays a central role in most areas of life [5, p. 12].

Collaborative or cooperative learning is contrasted with two other 'goal structures': competitive learning, and individualistic learning. Competitive learning consists of activities which set the students tasks which some are expected to 'win', by getting the highest score or completing the activity in the shortest time, for example. Individualistic learning is learning where there is no interaction between students at all—no comparisons are made, and students are solely concerned with working by themselves.

Johnson and Johnson argue that each of these goal structures has a role to play in education, but that each is suitable for different purposes [5]. Collaborative learning, specifically, is to be used in situations where the material being taught is especially complex or conceptual. This makes it ideal for use in maths lessons: Yackel, *et al.* describe the benefits of group work in teaching maths to second-grade[2] students in the US [11].

[1] These are commonly called 'mods'. Throughout the report, we will refer to our product as 'the mod'.

[2] Roughly equivalent to year 5 (ages 7–8) in the UK

1.2 Minecraft

> "Minecraft is a game about breaking and placing blocks. At first, people built structures to protect against nocturnal monsters, but as the game grew players worked together to create wonderful, imaginative things." [8]

Minecraft was initially developed in 2009, as a game where players could explore a randomly-generated world and place blocks to build structures. From the outset, Minecraft was not designed as a game that could be 'won'—in game industry terms, it was intended to be a 'sandbox' game where players set their own goals.

There have been a number of articles written exploring the potential use of Minecraft in supporting education [1], [9]. Habgood has already established the usefulness of computer games in general in education [3]; the question is whether Minecraft can be used in the ways that he suggests. This is what EduCraft seeks to establish, at least tentatively.

The official Minecraft Wiki has its own page dedicated to describing possible ways of using Minecraft in education [6], and suggests using the game's in-built system of building items:

> "The crafting system can help in teaching basic math (e.g. I need 3 Sugar Cane for Paper), which transitions to multiplication (I need 3 Paper and 1 Leather for a Book, and 3 Books for a Bookshelf, so I need 9 Paper and 3 Leather all together) and division (When I create Paper I get 3 at once, so $9/3 = 3$ times per Bookshelf I'll have to create Paper)."

We intend to build something more overtly mathematical than this basic concept, and the remainder of the report describes the requirements that have been set for the project, along with a record of our initial design and prototypes.

Chapter 2

Requirements Specification

2.1 Functional requirements

1. The game must cater for users who are currently at level 2 of keystage 2 mathematics.
   - 1.1 There must be a puzzle in the game which challenges the users knowledge of subtraction facts up to 10.
   - 1.2 There must be a puzzle in the game which challenges the users knowledge of number ordering from 0 - 100.
   - 1.3 There must be a puzzle in the game which challenges the users knowledge of number sequencing. This must include odd and even numbers.

2. The game must cater for users who are currently at level 3 of keystage 2 mathematics.
   - 2.1 There must be a puzzle in the game which challenges the users knowledge of the 2,3,4,5 and 10 times table.
   - 2.2 There must be a puzzle which forces a user to use multiplication or division to solve it. The multiplication or division must only be used with whole numbers. When choosing numbers for division numbers with remainders are to be included.
   - 2.3 The game should include puzzles which involves breaking entities into simple fractions.
   - 2.4 Within the game there should be puzzles which require users to equate simple fractions.

3. The game must cater for users who are currently at level 4 of keystage 2 mathematics.
   - 3.1 Puzzles must exist within the game which can only be solved using division by 10 or 100. The entities being divided must be whole numbers.
   - 3.2 A puzzle must exist which tests users upon their knowledge of multiplication facts up to 10x10. This puzzle should then go on to test whether the user can quickly derive correct divison facts.
   - 3.3 Within the game there should be a puzzle which involves adding and subtracting entities which represent decimal numbers to 2 decimal places.
   - 3.4 A puzzle should exist which forces the user to order decimal numbers. These decimal numbers can be up to three decimal places in length.

4. Levels in the game must be constructed in such a way that each user contributes to a problem before all the users can progress. A user must contribute within the game

physically.

5. The game must not be single player and should allow players in groups of 3-4 to play together.
6. The players should be able connect to dedicated servers to play the game.
7. The mod must implement coins as one of the visual representations of numbers.
8. The mod must have custom weapons which can be used to defeat zombies.
9. Coins and other visual representations of numbers must be collected by defeating zombies.
10. Zombies must not drop numbers or coins if the user does not defeat them using the specific mathematical weapon.
11. Zombies must have a number above their head clearly visible to the player indicating what number they drop upon being defeated.
12. Zombies should drop a number or coins equal to the number stated above the zombie's head.
13. Players must be able to craft all mathematical operators using items collected throughout the level. (Mathematical operators being +,-,*,/)
14. Each mathematical operator must be represented visually in a way which is commonly understood by people of the specified age group.
15. Puzzles must be solved in chambers as well as open world spaces.
16. Players should not be allowed to progress until the puzzle they are currently on is completed.
17. Mathematical operators must be usable objects and not weapons.
18. Mathematical operators will be crafted using items which have been ascertained through resource collecting, where resource collecting is the action of getting items in any way but defeating zombies.
19. Equations must be able to be crafted using mathematical operators and numbers collected from defeated zombies.

2.2 Non-functional requirements

1. The game must be implemented in Minecraft through a mod.
2. The mod should work on Minecraft version 1.6.4.
3. The mod must work on the windows operating system.
4. The mod must work on a machine which has the minimum requirements of Minecraft or better.

5. The game should abide by the ethics code in place within the education system for those between the ages of 8 and 11.

Chapter 3

Design and Implementation

3.1 Introduction

In this chapter we present a discussion of the design of the system, and key factors that influenced our design. We will explain the requirement to develop using Java, and our decision to use the MinecraftForge API.

The online tutorials for making MinecraftForge mods informed our decision to create our mod following this system design; examples can be found on the MinecraftForge wiki page [7].

3.2 Key implementation decisions

3.2.1 Programming language

Minecraft is a game written in the Java language, and so it is only logical that when we mod the game it is required that we use Java. This links in with the API we use, which is also dependent on Java and specifically is made with support for Eclipse (in terms for example of online tutorials for setting up and coding in the Eclipse JDK environment).

3.2.2 Application programming interface (API)

Before creating our mod we had to choose the method for modding the game. Our final choice was to use the MinecraftForge API, which is a mod/application layer for Minecraft which lets us create and run Minecraft mods. The decision to use MinecraftForge API was because of its versatility and usefulness.

Furthermore, we decided to use MinecraftForge over other alternatives because, while other tools like ModLoader may have been arguably easier to use, MinecraftForge is much more suited for our purposes namely, creating a more complex mod. This is because MinecraftForge has a lot more possibilities, whereas with ModLoader there is more limited scope for creating complex Minecraft mods. Aside from ModLoader and MinecraftForge, there wasnt much else choice; these two mod tools are the most popular by far in the Minecraft community.

3.3 Initial design of the system

Because of the very nature of MinecraftForge and our design, all modifications are done by creating new classes. Mods in Minecraft are divided into two types: regular mods, which simply add new functionality to the game, and 'core mods' which alter some of the game's existing functionality (changing gravity, for example). Because our mod is a regular mod, our code is placed in a new package separate from the rest of Minecraft's code, called `org.educraft`.

3.3.1 Proxy classes

All mods require proxy classses, which handle the rendering of entities. Given the simple nature of our mod, our proxy classes are currently empty.

3.3.2 Base class

Mods made with MinecraftForge require a base class which provides instances of all blocks and items which the mod adds into the game. For the dummy mod, this is found in `org.educraft.DummyMod`.

3.3.3 Item and entity classes

All items, blocks, and entities that are added into the game extend the appropriate pre-existing Minecraft classes: `net.minecraft.item.Item` in the case of items and `net.minecraft.entity.Entity` in the case of blocks.

All of our custom items and entities were placed in the `org.educraft.dummy` package. As far as possible, we chose to inherit from core classes which provided the functionality we desired, to minimise the amount of new code that needed to be written:

- `DummyCoin` and `DummyCoinPile` both inherit from `net.minecraft.item.Item`, as they are simple items with no extra functions.
- `MathsWand` inherits from `net.minecraft.item.ItemSword`, since it is an item that gets used as a weapon.
- `DummyZombie` inherits from `net.minecraft.entity.monster.EntityZombie`, so that we can reuse the model and animation provided for the existing zombie entity.

3.3.4 Damage handling classes

We also created two classes to handle the requirement that killing a Dummy Zombie with the Dummy Sword / Maths Wand should cause a different item from normal to be dropped. These were `DummyDamageSource`, which defined a new type of damage in addition to the pre-existing types such as 'fire', 'physical', and so on, and `DummyAttackHandler`, which provded a new event handler to ensure that striking a zombie with the Dummy Sword inflicted the new damage type.

3.4 Future development

It is envisaged that the components of the completed mod will be heavily based on the work done in the prototype. A new base class will need to be created, to define the EduCraft mod

(as opposed to the Dummy Mod), and some new items will need to be created, but the basic principles will be carried over: we will extend pre-existing classes wherever possible.

One new feature of the finished product that is not present in our current design is custom texturing: for the dummy mod we used textures build into the game for all of our items. This suffices for a prototype, but in the finished mod we will need to provide our own textures to make it clear to players what items they are using.

Chapter 4

Initial Prototypes

4.1 Introduction

Prototyping is an important part of any software engineering project, as it enables the development team to demonstrate their progress to their client in a way that they can more easily understand, and also enables the client to clarify what they want from the system [2], [10].

In this chapter we describe the prototypes which have been used to demonstrate EduCraft in its early development, along with how they were received by those we showed them to. We first describe the mod which was used in these demonstrations, and then explain the demonstrations themselves.

4.2 The 'Dummy Mod'

Both prototypes involved using something we called the 'Dummy Mod'. This was a simple mod which contained some basic custom items which we believed would serve to demonstrate the goals which we were aiming towards. Neither of the prototypes actually included any elements that would be used in the finished product; rather, they served simply to *illustrate* what we hoped to achieve.

The premise of Dummy Mod was simple: we provided the player with a customised weapon item (a 'Dummy Sword'), and the means to generate a number of modified enemies ('Dummy Zombies'). When a Dummy Zombie was slain with a Dummy Sword, instead of dropping its usual items, instead it dropped a 'Dummy Coin'. We also gave the means to combine nine of these coins into a 'Pile of Dummy Coins', and to split a pile of coins into nine indvidual coins.

These basic implementations demonstrate what we hope to achieve in the finished product, but with much less work: the code used to create the items was minimal, pre-existing textures from within the game were used instead of custom-made ones, and no special mathematical concepts were implemented. The purpose was simply to illustrate that we could, in principle, make a Minecraft mod which allowed players to use special tools to collect particular items from enemies, which could then be combined together.

4.3 Demonstration to supervisor

The first demonstration was carried out in a formal meeting with our supervisor. This demonstration was very basic: we did not intent to present anything approaching a working 'level' of the game, but simply wished to demonstrate that we knew how to work with Minecraft to produce mods.

In the demonstration, we gave the player a Dummy Sword, the means to spawn (generate) a Dummy Zombie, and a crafting bench in order to combine the coins into piles. The objective was very simple: the player spawned and killed as many zombies as they wished, and made and destroyed piles of coins as they wished.

4.3.1 Feedback

Feedback on this prototype was positive: it was clear from what was shown that we knew what we were doing, and were able to produce mods that could be used in a teaching environment.

The main criticisms of the prototype were visual. In an unmodified game of Minecraft, zombies and other 'undead' enemies burn in sunlight, and so for the pursposes of the demonstration the game's time was set to night. Whilst this is perfectly acceptable to an experienced Minecraft player, it was suggested that to someone who had never played the game before, the darkness might make it harder to understand what was going on.

It was also suggested that the perceived violence of the game might make it harder to 'sell' to schools—although the violence was fantasy, our supervisor observed that the game still involved killing monsters with swords. It was recommended that the game's graphics be modified to make the mathematical focus more obvious.

4.4 Demonstration at Southwold Primary School

The second demonstration was much more involved, and was given to the maths co-ordinator of Southwold Primary School, who would be responsible for deciding whether or not to allow us to test our finished product with children later on in the project. For this second demonstration, the feedback received from the first prototype was incorporated and built upon.

Elements of the prototype are shown in Figure 4.1.

4.4.1 Level design

For this demonstration, we created a simple level which used counting as its main objective. The player's goal was to collect twelve dummy coins and deposit them in a hopper; once the player had collected and deposited twelve coins in this way, the door at the far end of the level would open, allowing the player to progress.

The level was designed to be as minimalistic as possible, so as to demonstrate the possibility of creating mathematical puzzles in Minecraft without distracting the teacher with other details of Minecraft gameplay. The level conveyed the idea that we could build areas which players could not leave without first completing some problem—in this case, counting.

Figure 4.1: Elements of the prototype level, clockwise from top left: 1. the outer door, 2. the exit door and coin repository, 3. the means of spawning zombies, 4. the created zombie

4.4.2 Item design

We decided that we could retain zombies in the game, as fighting monsters formed an integral part of playing Minecraft which anyone who had seen the game before would expect. Habgood's PhD work had already established that it is acceptable for this sort of fantasy violence to be depicted in educational games [3], and so we felt that there was no problem in retaining it.

We did redesign the player's 'weapon', however—its texture was changed, so that instead of appearing as a sword it was a simple stick, and it was also called a 'Maths Wand' instead of a Dummy Sword. We hoped this would more clearly convey the mathematical nature of the game.

4.4.3 Feedback

The feedback for this second prototype was overwhelmingly positive. Despite the teacher's complete lack of familiarity with Minecraft, she appeared very excited by the prospect of being able to use it in lessons. She said that she believed the level of violence would be acceptable, provided that the emphasis was very clearly on the maths and not on the killing. This agreed with what we had already heard from our supervisor.

Demonstrating the prototype also provided us with an opportunity to refine our requirements, so that we could better direct our future work.

Chapter 5

Progress and Problems

5.1 Introduction

In this chapter we will describe a discussion of problems faced during the project so far, including both programming and people.

5.2 Research

Everyone did their due diligence and read Jacob Habgoods thesis. Janos managed to find the original thesis in the library which included a cd of the game and source code as well. Unfortunately, we were not able to read the contents of the CD.

The next task was to get familiar with Minecraft. Only Ali and Ian had any previous experience playing it. We tasked ourselves with getting familiar with the game and Janos was very helpful with getting the minecraft environment set up for everyone.

After gaining some understanding of the features and limitations of minecraft. We researched what platform to use to develop our mod. We eventually decided to use MinecraftForge API for reasons discussed in the design/implementation section.

Eze brilliantly took it upon himself to research the KS2 Maths Curriculum from which he developed the first version of our requirements specification.

5.3 Assigning Roles and Team Leader

One of the very first tasks to accomplish was the assigning of roles and who would be team leader. Our group supervisor encouraged that the roles be switched around as the project continues. We decided to do this. Roles assigned for the first iteration are:

- Documentation - Tosin/Janos
- Website - Tosin
- Version Control - Tosin
- Design Ideas - Ying
- Coders - Eze/Ian

- Testing - Ali

During the Formal Group Meetings, the role of chairman and secretary was assigned at the beginning of each meeting.

Choosing a Team Leader was the least trivial decision. No one person was keen on performing the role but Janos put himself up for it. The team has got along well so far so there hasn't been a crucial need for the team leader to use his authority.

5.4 Generation of Ideas

After concluding initial research and deciding roles, the team looked into implementing the actual game. The focus was to keep collaboration at the core of the game design. We took some inspiration from habgood thesis.

We developed a core theme of how to integrate collaboration in the levels. Basically, there would be some sort of game mechanic whereas players would be able to earn mathematical operators and numbers. (E.g killing zombies in a non graphical way). The players would then solve a mathematical problem by combining the numbers and mathematical operators from each player. Upon completion of the problem, they would then all proceed to the next level.

5.5 Web and Repo

The version control system provided by the university is SVN. Most of the team was more familiar with Git and hence the team pushed to used Github. Permission was obtained to use Github as long as we kept some sort of backup with svn.

Tosin looked into converting Git commits and history into svn but it didn't seem very feasible. During this time, unfortunately no backup has been set up on the svn yet. Starting from the 13th of december 2013, a weekly backup of the git repo will be made. This will include the git history allowing us to still revert commits and perform other commands incase the git repo on github ever fails.

The project site is hosted on code.cs.nott.ac.uk. The Project site was completed in due time and the team gained 3 out of a possible 5 marks. One of the main loss of marks was due to Tosins foresight as to status of the project description.

When Tosin was setting up the Project Site, he raised an issue that members of the team should look into contributing some more info to the project site, unfortunately this went unanswered. In hindsight, an easy way to improve the Projects description would have been to add Ezes initial system requirements.

5.6 Prototyping a mod

Before the christmas break, it was planned that we get a simple, one player mod of our project working. Progress on that has been quite good, with Ian contributing quite a bit of code.

We are now on our third prototype and we've had no problems with the actual programming of the prototypes so far. The only problems we have faced when prototyping is coming up with designs that solve the problem of the project. At first it was hard to envision a way of solving the problem in a way which uses Minecraft as a learning tool. Every idea took Minecraft and

attempted to turn it into something else to solve our problem which isn't what we wanted. Eventually we overcame this and created some ideas which seemed to work in the world of Minecraft, the next problem was expanding these to make fully functional parts of a level (we are currently working on that). The reason why we had such trouble with coming up with ideas was because our communication was not great, we didn't convey our ideas to each other in a good way so we couldn't share and expand on ideas well. This left us attempting to build up major ideas alone which for a problem like this is quite difficult.

Chapter 6

Time Plan

Our first meeting was on 7th October 2013. The team met with the supervisor and started planning on the same week. The Gantt chart presented in Figure 6.1 to show the exact time intervals planned for each stage of the project. This is the most up to date plan we have at the moment, but please keep in mind that parts of it will be extended or updated in the future as the need arises.

6.1 Background Research: 07/10/2013 – 25/10/2013

All team members will have to do their own research about Minecraft. Everyone should play the game to get more familiar with it and understand its key concepts. We will look up how to create a mod for the game and find tools and resources to do so. The team will also read the most relevant parts of a thesis suggested by our supervisor. The name of the study is The Effective Integration of Digital Games and Learning Content" written by Jacob Habgood. We must read the "National Curriculum in England, Key Stage 2" in order to understand what mathematical elements to include in our game.

6.2 Initial Decisions: 14/10/2013 – 23/10/2013

The team will identify the most important activities of the project and choose members to be responsible for different duties. All team members should take the Belbin personality test on-line as suggested by our supervisor. The team will decide on which development tools to be used and what software development methodologies and techniques to be followed. Ideas will be discussed during this period. The team will decide what basic concepts and features the game must have during the early stages of development.

6.3 Website & Repository: 18/10/2013 – 31/10/2013

The person in charge of creating the project site will set up a website. The website must include extended description of our project, links to our documentations and proof of usage of a version control system. The person in charge of version control will set up a repository and help other team members to do the same on their computers if it is needed. Each team member will be required to commit to our repository and raise an issue, which must be closed.

6.4 Prototyping: 23/10/2013 – 13/12/2013

The people in charge of coding will create a playable prototype by the end of this term. This will be shown to a few children in the hope of gaining some feedback to help us with further development of our mod. Each team member must have an identical copy of our development environment on their computer. This copy must be linked to our version control system as well. A very basic mod will be created in order to demonstrate that the group is capable of developing mods for Minecraft. In the next stage we will have a very simple, but playable one. This could be used to show the translation of our idea to teachers to gain valuable feedback. The mod will have some basic mathematical elements to it. Our final prototype should be a complete one level implementation with some very basic mathematical problems.

6.5 Game Development: 16/12/2013 – 21/03/2014

This will consist of two major stages. During the first stage we will create a single player implementation of the game. This will be used for initial testing in a local primary school. The second part will introduce the multi player, collaborative aspect to the mod, which will also be tested with groups of children. We will have a more explicit schedule by the first week of January, 2014.

6.6 Testing: 12/16/2013 – 27/03/2014

Different types of testing will be carried out throughout the development process. Our supervisor and team members with young relatives will test the prototype during the winter holidays. The person in charge of testing will outline a detailed plan for testing operations before the beginning of spring term.

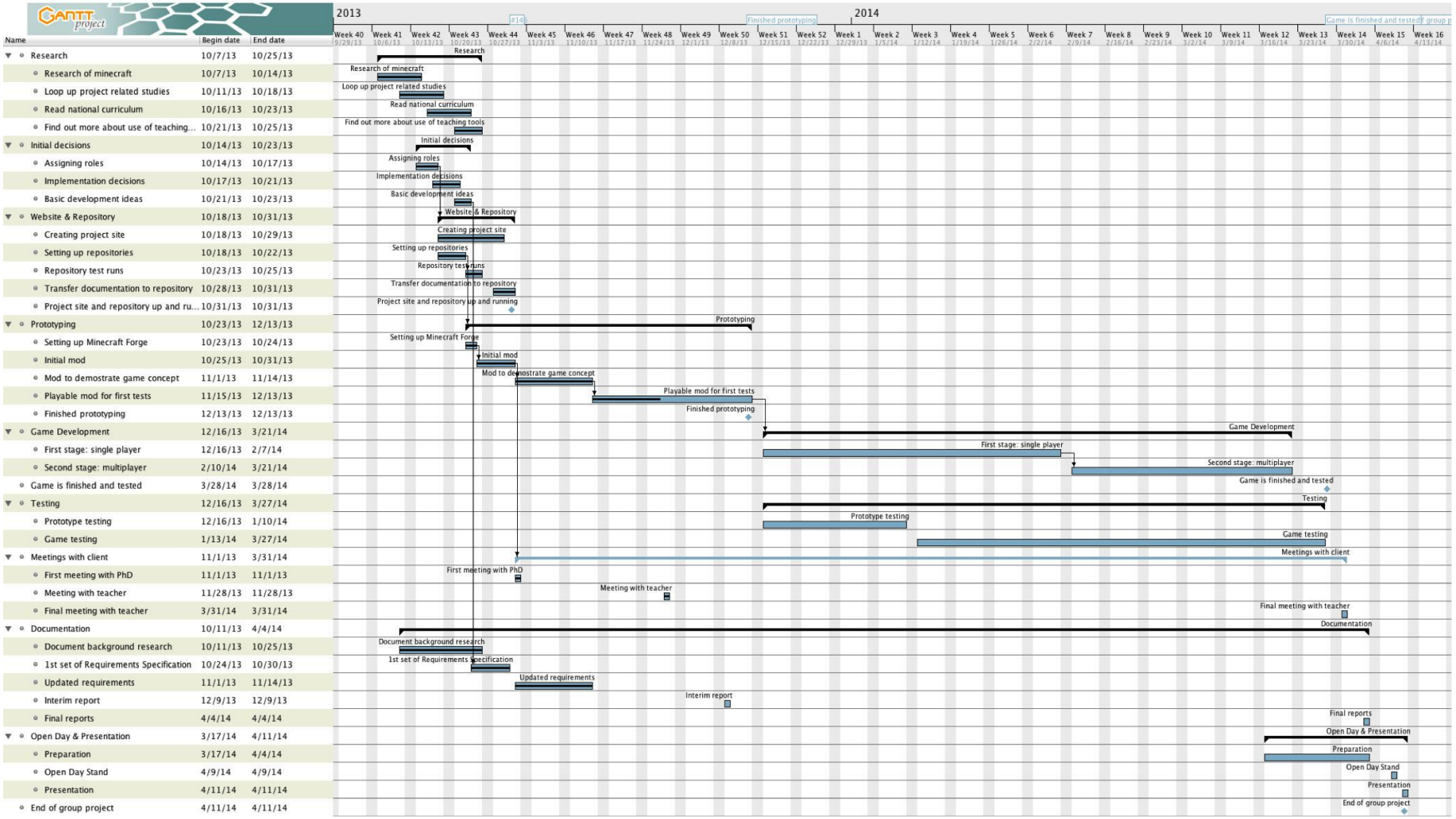
6.7 Meeting with clients: 11/01/2013 – 31/03/2013

Meetings will take place when it is suitable for both the client and the team. The team will not meet with any children until the prototypes are finished. We will show case the early implementations of our mod to teachers. Once the school approves us showing the game to children we will start testing the product with them. We will flexibly schedule our visits to the school when the need arises, which will be February, 2014 the earliest. Our final visit to the school will take place by the end of March, 2014.

6.8 Open Day & Presentation: 17/03/2014 – 04/11/2014

The team will start the preparation for the final part of the project around the time when the development of the game finishes and the final stage of testing starts. Further planning will take place by the beginning of March, 2014. The project will end on 11th April 2014.

Figure 6.1: Provisional Gantt chart describing time plan for the project



Bibliography

[1] J. Brand and S. Kinash, "Crafting minds in minecraft," *Learning and Teaching papers*, vol. 53, 2013.

[2] F. P. Brooks, "No silver bullet: Essence and accidents of software engineering," *IEEE Computer*, vol. 20, no. 4, pp. 10–19, 1987.

[3] M. P. J. Habgood, "The effective integration of digital games and learning content," Ph.D. dissertation, University of Nottingham, July 2007.

[4] D. Jacques and G. Salmon, *Learning in Groups*, 4th ed. Abingdon, England: Routledge, 2000.

[5] D. W. Johnson and R. T. Johnson, *Learning Together and Alone*, 4th ed. Needham Heights: Allyn and Bacon, 1994.

[6] Minecraft Wiki, "Minecraft in education - Minecraft Wiki," December 2013. [Online]. Available: http://minecraft.gamepedia.com/Minecraft_in_education

[7] MinecraftForge Wiki, "Tutorials - Minecraft Forge," December 2013. [Online]. Available: http://www.minecraftforge.net/wiki/Tutorials

[8] Mojang AB, "Minecraft," November 2013. [Online]. Available: http://www.minecraft.net

[9] D. Short, "Teaching scientific concepts using a virtual world - minecraft," *Teaching Science*, vol. 58, no. 3, 2012.

[10] I. Sommerville, *Software Engineering*, 9th ed. London, England: Pearson Education, 2011.

[11] E. Yackel, P. Cobb, and T. Wood, "Small-group interactions as a source of learning opportunities in second-grade mathematics," *Journal for Research in Mathematics Education*, vol. 22, pp. 390–408, 1991.