Player-aided development of
*Gods and Mortals*, the world’s first mind-controlled multiplayer Facebook game

2011
Aðalsteinn Sævarsson
Bjarki Einarsson
Vífill Valdimarsson
Þórður Roth
BSc in Computer Science
Table of Contents

1. Introduction ......................................................................................................................... 3
2. Artifacts ................................................................................................................................. 4
3. Project description ............................................................................................................... 5
4. Planning ................................................................................................................................ 7
5. Design ................................................................................................................................... 9
6. Progression ............................................................................................................................. 10
7. Conclusion ............................................................................................................................. 12
1. Introduction

1.1 The purpose of the project

MindGames develops mind-controlled games which teach players how to relax and concentrate when they want to. The purpose of this project was to apply a well-structured process to develop the world’s first mind-controlled multiplayer Facebook game, Gods and Mortals, into a compelling and popular experience.

1.2 Previous work on Gods and Mortals

In spring 2010, three RU students developed a prototype of a multiplayer mind-controlled Facebook game as their final project. MindGames then expanded on their work to create the first version of Gods and Mortals. The game enables owners of brainwave (EEG) headsets to play “Gods” with awesome mental powers, against “mortals”, who are trying to reach heaven in order to overthrow the gods with no benefit of brainpower. If there is no god present then the game turns into a free-for-all battle between the mortals. The first mortal to reach heaven wins.
2. Artifacts

The CD attached to this report contains all artifacts produced related to the project. The documents on the CD provide much more details than this report.

2.1 CD contents

- Final Report
  - This document in digital form
- Project Planning
  - Contains all information concerning early analysis and planning regarding the project, such as risk analysis, tools used and the work schedule.
- Design Document
  - Explains the design of the game.
- Progress Document
  - The progress document provides a detailed description of every sprint, including stories, tasks, a burn-down chart and more. The progress document is broken up into two separate documents, the first half of the project and the second half, to reflect the change in organization.
- User Manual
  - Explains the basic of the game and helps out new users.
- Administration Manual
  - Explains how to set up and update the game.
- Developer's Manual
  - Contains class diagrams and information about the programming languages and tools used.
- Test Report
  - Contains details concerning the user tests.
3. Project description

3.1 Performance

There were many aspects of the game that needed to be remedied in order to make the game enjoyable for the users. Listed below are the major issues.

3.1.1 Complex graphics

The graphics weren't simple bitmaps and involved complex vector calculations for some of the most simple animations, such as the “idle” animation for mortals. This led to an FPS (frames per second) drop and made the game very taxing on the user's hardware.

3.1.2 Response time

As this is a multiplayer game a server is needed to keep track of everything and synchronize the game so that every client is having the same experience. This leads to delays when executing certain tasks. The solution to this was to execute some actions locally, if the client permits it, and then ask the server for permission. For more details see chapter 5 - Design.

3.1.3 Map awareness

The game contained no mini-map, so players had no idea what was happening elsewhere in the game. As the player's screen doesn't capture the entire game at once, one could simply move to the side and attempt to reach the top while outside everyone else's point of view. This was also confusing as the game would sometimes end abruptly as someone reached the top quickly. Other players were maybe not even aware of the fact that anyone else was above them.
3.2 Gameplay

The main goal of the project was to improve the gameplay, firstly to improve the mortals-only gameplay and secondly to improve the god-mortals gameplay.

3.2.1 Mortals-only gameplay

The previous version of Gods and Mortals featured a free-for-all battle between mortals. Having gone through testing it was clear that the gameplay lacked both depth and variety. It was basically just a race to the top, where you paid little or no attention to other players.

In order to improve the mortals-only gameplay it was decided to change it from free-for-all to a team battle, where two teams fight to reach the top. This means that mortals will not only think about themselves, but also their teammates. In order to win a team must reach the top with two more mortals than the other team or reach the top with all of its players. This means that you must assist your own team and stop the enemy team if you want to win.

3.2.2 God-mortals gameplay

The gods in the earlier version of the game were too powerful and could simply kill any mortal who was advancing too fast. It was decided to remove the god’s destructive powers and instead grant them less lethal powers that slow the mortals down, without killing them. Killing a mortal, in an earlier version of the game didn't remove him from the game. The mortal would respawn at the bottom. Even though the mortal was “still alive” this move was too powerful. As hitting the mortals with the destructive spells was exceedingly easy, a god could simply wait at the top and then kill any mortal who travels so high. By only giving the god spells that confuse and disrupt player movement the god has to be more tactical. As the winning conditions state that in order for a team to win it must have 2 more mortals at that top than the other team, a god might even decide to assist the losing team in order to prolong the game as the longer the game goes on, the more odds are that the god will win. The reason for this is that it takes the god a few minutes to charge his ultimate spell. Once fully charged the spell kills all mortal players and the god wins the game.

With an invisible god that casts global spells nobody can ever anticipate what will happen next. As players have no idea where the god is located his striking is more unexpected.
4. Planning

4.1 Methodology

It was decided to use Scrum. Scrum is an iterative, incremental framework for project management often seen in agile software development, a type of software engineering. Although the Scrum approach was originally suggested for managing product development projects, its use has focused on the management of software development projects, and it can be used to run software maintenance teams or as a general project/program management approach.

4.1.1 Roles

Product Owner: Deepa Iyengar
Scrum Master: Þórður Roth
Team: Aðalsteinn Sævarsson, Bjarki Einarsson, Vífill Valdimarsson, Þórður Roth

4.1.2 Sprints

At first each sprint was two weeks, starting on a Friday at 16:00 with a sprint planning meeting at MindGames. There was a mid-sprint meeting one week into each sprint and on the last day of the sprint a sprint review and a sprint retrospective was held.

Several weeks into the project it was decided to shorten the sprints so that each sprint would take no more than a week. This also meant that there would be no mid-sprint meeting, as it was obviously not needed. It was also decided to have the sprints start on a Monday.

4.1.3 Tools

As MindGames had experience with Scrum we were advised on how to keep track of our backlogs and hours worked. For time tracking purposes we used Tick and as a backlog/task-board we used Pivotal Tracker, which is designed for Scrum. However, this was difficult to manage, as all tasks had to be recreated in Tick in order to add hours to them and Tick wasn’t made for Scrum to begin with. Also, Pivotal Tracker was quite complicated while providing very little information about tasks, stories and progress. Following our second status meeting we stopped using Pivotal Tracker and Tick and started using Excel instead, as advised by our moderator and our instructor. This made keeping track of sprints and hours spent much easier and provided a much better overview of our progress.
4.3 Testing – Unit tests and user tests

At the end of each sprint we would test the new features added to the game. In the final stages of the project, when we had more time, we also added mid-sprint tests. These tests were then used to write a list of bugs found, which would be addressed in the upcoming sprint. On the 5th of May we held our first alpha test, inviting 11 players to play the game for an hour and subsequently fill out a questionnaire. On the 13th of the May we held our second test. The attendance was not as great as in the first test.
5. Design

The most important thing in designing this game, we believe, was to make it so that another group could easily take over and expand upon, or even alter the game if need be. When documenting and commenting we thought “what would we have liked to have gotten at the start of the project”. The final product can be broken up into two separate pieces, the client and the server.

The client can only operate if it can connect to a server. As we did not want someone to be able to modify the client, or the messages that it sends, in order to cheat – the server needs to check every move and every action taken and compare it to its own game state. The client creates a simulation of the game running on the server, in order to reduce lag and give the client an acceptable response time when executing any given action. While a person may be able to alter the client to fool itself, for example by relocating the player’s avatar or turning off the gravity, the server will not acknowledge such actions and will not move the player accordingly.

The reason behind having the client create a “copy” of the server game-state is that the client must execute some commands locally before asking the server for permission. In today’s games it is unacceptable to, for example, press space bar and then waiting half a second for your avatar to jump. What the client does is that he executes the command locally and then asks the server for permission. As the server is constantly sending information to the client the difference between the game states shouldn’t be too great. So if the client can execute a command locally odds are that the server will allow it. However, in the rare cases where the client’s state doesn’t correspond to the server’s state the server will deny the client when asking for permission. This leads to the client rolling back the action. This could happen if the client experiences a severe amount of lag over all of a sudden or his internet connection goes down for a few seconds. This is mostly in place to prevent cheating though.

There are some known issues concerning the amount of players in a room at any given time. As the server sends out an individual message for each action taken the amount of messages can be too great if every player is spamming actions at an extreme rate. A possible solution to this would be to group up multiple messages into a single message when dealing with such an extreme amount.
6. Progression

6.1 The start

During sprint zero the team spent its time organizing the project resources and project plan. The team familiarized itself with Scrum, Pivotal Tracker, Tick, Adobe CS5, PlayerIO, the Facebook API, Assembla, Actionscript 3 and the existing code. A lot of time also went into brainstorming in order to find ways to improve and expand upon the existing gameplay. These ideas were then introduced to the Product Owner and once she was satisfied a release backlog was created. Little or no coding took place during these first weeks.

6.2 First status meeting

After the first status meeting it was obvious that many stories and tasks were too large in scope. It was also apparent that not having any specific days nailed down for working on the project caused other classes and projects to take priority. This was pointed out by both the instructor and the moderator. It was then decided to devote Saturdays, Sundays, Mondays and Thursdays to the project. Also, some of the stories/tasks were redefined and sent to the instructor for approval.

During this time it became apparent that the existing code was not so easily expanded upon. It proved to be way more complicated than what was initially thought. Our main contact at MindGames, who best knew the code, also quit during this time. As there existed no documentation for the existing codebase the complexity of all code related tasks grew exponentially.

The team discussed rewriting the game from scratch, which was not in the scope of the original project. It was obvious that the first game hadn’t been designed with such drastic changes in mind, but whether rewriting everything would solve anything wasn’t clear. The team did not manage to complete the first sprint and it had to be “redone”, thereby delaying all other sprints and pushing an entire sprint off of the release plan. While the second sprint went better there were still tasks remaining, which had to be moved to sprint three. In the middle of sprint three we had our second status meeting.
6.3 Second status meeting

Nobody was happy with the progression of the project and it was obvious that something drastic had to be done. Well over 500 hours had been spent on the project and we had very little to show for it. Our instructor advised us to create a small demo in order to show him, ourselves and MindGames how simple it would be to rewrite the game, with all the new features in mind. It was also pointed out how our current documenting system had too much overhead and was too complex, while not really providing any useful information. It was decided to take one week to completely re-plan and “restart” the project.

Over the course of a few days a small demo client/server was created and shown to the instructor and Product Owner. A meeting was held to discuss whether or not it would be better to rewrite the game from scratch. Even the original author of the server agreed that the existing code contained fundamental flaws and had to be rewritten if it was to be expanded upon. Everyone agreed that both the client and the server should be rewritten.

To address the documenting issue it was decided to switch to Excel (google docs) as was advised by our moderator. It didn’t take long for the entire team to realize that this system was much simpler, while providing an abundant of information and, for example, the ability to automatically create burndown charts.

It was also decided to shorten the length of sprints, from two weeks to one week. This would allow our instructor to better keep track of our progress as a product increment had to be released each week.

We also got the help of a couple of MindGames employees that helped with player synchronization and set up the Facebook client and Neural Headset communication.

6.4 The “restart”

It was immediately decided that, even though we were rewriting the game from scratch, we still had to complete all MUSTs that had been defined at the start of the project. There were a total of seven sprints. We managed to finish every single sprint on time and even hold a couple of user tests, while rewriting the game from scratch and implementing all the MUSTs. The team, as a whole, spent over 600 hours during this time, totaling around 1200 hours over the entire course of the project.

6.5 Summary

In retrospective it is quite obvious that the team grossly underestimated the complexity of the code, even though close to a hundred hours were spent on getting to know the existing code. The team also failed to acknowledge the risk that the main contact at MindGames would quit and had no plan in place to deal with that. The author of the original server did later replace the previous contact and proved to be invaluable regarding his knowledge of the old code. Having him state the fundamental flaws in the existing code was key in making the decision to rewrite the code. The demo created also helped to confirm the decision taken to rewrite the code. It might have been better to create this small demo earlier, as it only took a few days.

A new version of the game, with certain features added, had to be in place before the 26th of April for Tækníþróunarsjóður. This served as a disincentive to rewrite the game from scratch and was one of the reasons for not making the decision to rewrite the code earlier. In the end, even though the code was rewritten from scratch, the deadline for Tækníþróunarsjóður was met.

As the team lacked experience in managing a big project like this it was not apparent how flawed Pivotal Tracker and Tick were, until the decision was made to switch to Excel (google docs) instead, as advised by the moderator and instructor.
7. Conclusion

One might assume that a lot of the time spent on the project was simply wasted having spent so much time on the existing code and then taken the decision to rewrite the code. However, this is not the case. By examining the old code in such detail and attempting to expand upon it the team learned a lot. As we grew more familiar with the old code we knew what pitfalls to avoid when rewriting, we knew what worked and what didn’t work. This led to the second half of the project being extremely smooth and productive. The lessons learnt during this time have already proven to be invaluable.

Everyone learnt a great deal over the course of the project, especially things that concern project planning and management. If we were to take on such a project again, we would use Scrum and do some things quite different from last time, such as:

- Make Sprint 0 at least two times longer.
  *When the project involves using code written by someone else one should take as long a time as is needed in order to get to know it. While adding a tiny element to the code would not require much knowledge of the existing code, making fundamental changes would.*

- Use a file-keeping /documenting system that doesn’t get in the way or take too much time to manage.
  *Very much time went into using and managing Pivotal Tracker and Tick. Switching to Excel was an excellent decision and we doubt a simpler program could be found to keep track of everything.*

- Have short sprints, short stories and short tasks.
  *The smaller everything is the easier it is to both define it and manage it. By having short sprints you should see a product increment more often, thereby spotting problems early on, while also encouraging the team. We found seeing a product increment every single week to be most invigorating.*
  *By breaking stories down as much as possible it is much easier to spot potential problems and to see if no progress is being made.*

- Begin working on as many stories as possible as early as possible.
  *The sooner work starts on a specific story, the sooner problems may be identified and dealt with.*

- Ask for help and acknowledge mistakes.
  *Obviously we should have reacted sooner to the situation that came up in the first half of the project and asked for help. This is mostly due to the fact that the team was inexperienced in managing a project of such caliber and thought that things would simply “work out”, which was not the case.*

All in all, everyone on the team is extremely proud of their work and the final product. The road may have been bumpy, but nobody gave up and in the end a fantastic product was delivered.