Game Design:
Individual Reflection

Emil Erik Hansen - Birthday: 14-06-1985 - ITU-mail: emha@itu.dk

14. December, 2011

A max. 3000 word individual reflection on the design and development process. This document is evaluated following these criteria:

- *The student should identify the different parts of the game development process, as well as her/his role in each of these.*
- *The student should be able to reflect about her/his participation on each of the development stages, identifying critical situations and how those were addressed.*
- *The student should use references from the course literature to illustrate her/his arguments and reflections. This use includes critical readings of the literature as well as the use of the game produced as a case study.*

Contents

1 Introduction

Game Design is many things. The whole process of creating a game based only on an idea all the way to an actual finished product has many stages and involves a lot of different processes, all of them being important for reaching a good end-result. To achieve the product that the course required us to produce, our team of two designers and three programmers was formed, leading to the game "Plane and Simple".

Throughout this reflection, I will go into the more specific parts of the project which I was most involved in, rather than trying to cover the whole process - or all the things everyone took part in. Section 2 will elaborate more on what my role as a programmer in the project meant.

The first thing I am going to get into is the importance of having a structured back-end for the game. This was especially important because the project, as mentioned, had three programmers. As such, *Code Management* is the primary contents of section 3.

Furthermore, our large programmer-part meant that it would be a very big advantage to make the game suited to let potential expansions and extensions easier to add. The subject of expandability will be discussed in section 4.

Lastly, I will go into the explanation and reasoning behind the addition of a specific feature of the game called "The Lockdown", which I thought of and implemented. While it was easy to implement, the actual design idea behind the feature is the most vital to describe. This will be covered in section 5.

1.1 A Plain and Simple Game

Our game project is the game "Plane and Simple", which builds on the classic arcade elements of picking up coins, beating levels as fast or effective as possible, as well as having fun while doing it. It is meant to be easy to learn into, while still having lots of room for the player to improve and evolve. We wanted a *well-played* game, that worked no matter if you were playing alone or with others (DeKoven, 2002).

The uniqueness comes from the fact that the player controls two entities, rather than the standard of one. To not make it too confusing, these two entities are bound together by a rope that most importantly serves to both add a control-challenge, but also to let the player focus on one place rather than two different points of the game field. The rope also serves as a tool to pick up objects, which is normally carried out by the player-controlled entities.

To further make the experience unique, we utilized the two analogue sticks of an "*XBox 360*"-controller - one for each of the planes. While many games have used both of the analogue sticks at once, we felt that it had

never really been done in a way where each of them represented the exact same thing. Normally, one of them would be for movement, while the other would be for aiming and shooting. A good example is the game "*Geometric Wars*", from which we were inspired a great deal. The goal was to achieve a unique way of doing normal controls and bending the game mechanics in a way that is not completely expected, like "*Shadow of the Colossus*" does it with its different control-scheme (Sicart, 2008).

The game takes a very interesting turn when two players each control a plane, as the rope-mechanic ensures that a good and enforced co-operation is needed. It provided a very interesting element in keeping the players so closely tied together. The players have to actively control the set of planes around, rather than simply doing the best they can on their own to achieve the winning goal.

2 Being a Programmer

I was one of the three programmers we had in our group. We all took equal part in deciding how the various technical aspects of the game should be approached, and we often worked in pairs to settle how problems should be solved, how a certain improvement should be planned, and so forth. Thus, we all had the roles of "lead programmers". However, it makes little sense to call it "lead" when everyone is in the same position. It could easily become a problem in a larger scaled project, if everyone has an equal say, because it would be much harder to achieve agreements.

2.1 My Part in the Project

My part of the work was more focused on organizing and optimizing the code to make it more approachable, streamlined, and to remove the various duplicated functions that were often present. Duplicates were a natural result of quickly implementing new features, and abandoning them after concluding that they were working as intended.

A good example of this is that at some point there was a function for updating each of the two planes. The benefit of combining these two was to make changes easier, as you would only have to change something in one place, rather than two. It also made it much easier to find and fix bugs, as there were less places they could be present and only had to be fixed once.

As the game had been structured by the form of *Rapid Prototyping* from the very start, we had a working version of the game very early in the process. It was done to quickly get a good idea of how the unique controls felt, and it gave us much longer to fine-tune it, and to figure out how to make it work in a realistic and intuitive way (Sass, 2006). This also meant that keeping the code ordered was important from very early on, so that the multiple additions would not hinder the rest of the game development.

2.2 Multiple Roles

While it is important to have good programmers in a project, it makes little sense if communication between the programmers and designers does not work. By that, I am not referring to basic communication, but more the idea of translating a specific game-addition from sketches and the idea-phase into working code.

As our group had a majority of programmers, we could approach this in a very unique way and make the whole structure more "flat". The programmers took a great part in the overall design, as we as programmers could also quickly eliminate design-ideas which we knew would be too technically demanding for a small-scale project and focus more on realistic and effective approaches. Another advantage was that we programmers could get more into actually designing things, rather than only coding. Naturally, this was needed, as it was unrealistic to keep all of us actively programming all the time.

All in all, this gave a good - and refreshing - approach to doing group work. As a programmer, I never got the feeling that work was being *passed* to me, but rather that I took as much a part in shaping the ideas like everyone else, and followed them through. A good example of this, is the *Lockdown*-feature, which I will go in detail with in section 5.

3 Code Management

The biggest challenge and need for code management came from the fact that we had three programmers working on the project. Work had to be done very systematically if things were to work out fluently and without the need to carefully communicate each time the code was to be combined and tested when a new feature was added, changed or improved.

To achieve that level of management, more than just good communication and mutual trust was needed, while also making it easier for ourselves, so we could focus on the important thing - coding. While managing the project itself is the most important, so is the underlying sub-management of the coding and the code itself.

3.1 Tools of the Trade

To solve the potential problems, we added our project to the online repository system, *GitHub* (found at `http://github.com/`). What it does, is allowing for easy code sharing and handling the merging of the code when it has been worked on from different sources. It is always possible to see who changed what to the different files, and thus allowing for a very advanced version control.

However, while it is helpful to have a nice tool, it also has to be used properly in order to deliver the expected results. Because of that, it is necessary that different changes are committed to the system individually rather than in a big cluster of multiple changes, which makes it easier to see how exactly a specific change was made. To further help that process along, we decided to keep more detailed track of what changes were made to the game, combined with an executable after each major revision. That way, it was easy to get an idea of what changes had given which results and when it was made.

GitHub also offered some additional helpful tools to help with the management. We especially used the wiki-system to convert the more rough list of changes to a structured change log. While the abilities to revert to older versions, combined with all kinds of graphs based on use, could be useful, those features were not used to their fullest potential. That, however, is nothing negative, but merely shows that *GitHub* is also excellent for showing the numbers regarding who did what in a specific project.

A detailed look of how *GitHub* was structured, can be seen by looking at our project, which is placed at the address: `http://github.com/abcfantasy/Plane-and-Simple`.

4 Expandability of the Game

One of the decisions that was made very early, was that the game should be easy to expand upon, and be structured in a way so that we could really take advantage of having three programmers. The object-oriented style and structure of the game made this very easy and it also opened up for the opportunity of potentially adding further elements later.

This lead to the game being separated into different segments, thus making it very easy to expand in different directions at the same time. These segments included parts like menus, the player, objects and power-ups, along with an easily expandable level-format. That way, extensive changes could be made to the player and the controls, while new levels were being added without conflicting with each other in any way.

4.1 Objects and Power-ups

The object-oriented design was especially used with regards to the coins and jewels. They build upon the same object-type and share many properties and behaviors. The only actual differences were how they affected the player, the sound they made upon being picked up, and how they looked on the game field.

The real benefit about this object-type, was that it could also be used for all other objects on the playing field. Sadly, it was never fully taken advantage of, as the idea of power-ups had to be removed from the finished version

of the game, due to time limitations. The idea was that the object-type could easily produce power-ups, which would add various different (temporary or not) abilities to the player at minimal expense of the programmer.

4.2 Levels

The system behind the different levels also allows for easy expanding. Each of the levels has two files associated. The first is a list indicating which ground-tiles (which, upon touching the player kills him) are at which coordinates at the grid which every level is built upon. The other file contains lists of pixel-based coordinates of the various objects, such as coins and jewels. Anything else is taken care of automatically by the game itself.

A thing that was later added to the Level-system was the scrolling view. In the previous iterations of the game, the playing field was limited by the size of the game-window, which again was limited by the screen's resolution. By changing the view to follow the planes around, it opened up the option of making the levels much larger than the screen, which gave more options for how advanced and difficult levels could potentially be. A good example of this can be seen at the game's sixth and - at this point - last level.

The combination of the easy level-format, and not having any limits in terms of size, gave the game a tremendous bonus in terms of continuous expandability, as future levels were basically limitless.

5 The Lockdown - Adding a Unique Element

In order to best explain the *Lockdown*-feature, the idea behind the rope connecting the two planes needs to be elaborated. Each of the planes represent a physical object with a particular weight. The rope can change length until a certain distance is reached, upon which it will change to become tighter, thus pulling the planes toward each other. This makes the control of the planes difficult when the player is flying with the maximum rope-length.

What the *Lockdown* does, is that it allows for a given plane to immediately stop in its tracks. Additionally, a plane under lockdown cannot be pulled off-course by the other plane. As the game used assets from a simple physics-system, this was done by giving the plane in *lockdown* an incredibly high mass, which made it impossible for the, now much lighter, other plane to affect it in any way.

5.1 Benefits and Potential Uses

The main benefit of the *lockdown* is the ability to safely travel through a level. If the player loses control over one of the planes, he can simply perform a *lockdown*, and it will be out of harms way, unable to crash into anything dangerous. This benefit is merely helping against high speeds and dangerous

maneuvers and is not based upon the fact that there are two planes with a rope between them.

When the rope is taken into consideration, the *lockdown* suddenly gains many other useful properties. With one of the planes locked down, the rope can be thought of as a leash, keeping the other plane within a specific radius, and making sure that it does not accidentally crash into objects further away. Additionally, if a high speed is maintained, locking down one plane can make the other swing around, thus performing a daunting maneuver that would otherwise not be possible.

Thus, in its current form, it serves both as a safety-net for those that want to play it safely, and as a possible way for playing it risky to gain that little edge in speed or maneuverability. The player should be awarded for using this feature creatively to achieve improved performance, while being held down by as few rules as possible (Hughes, 1983).

6 Conclusion

All in all, the process of developing and programming "Plane and Simple" has been very interesting. The project gave great opportunities for engaging in different parts of the creative process, as well as seeing the fruition of the hard work put into making the game.

The role of being a programmer in the project showed that there is a lot more to the process than simply programming a game based on an idea. It required intensive re-thinking of how aspects should be approached, as well as a large amount of iterations before a successful result was given that felt close to the original idea.

The game itself fit well with the requirements of the project and turned out to be greatly entertaining, both in single- and multiplayer. It was very interesting to witness just how different it became in the two different environments. Furthermore, it felt great to deliver a game that had a large potential of growing, if we decide to return to the game at some point later in time.

After having worked on this project one thing in particular has come to my attention. A game is never *really* completed. There is always a little addition to make, a detail that can be polished, and something completely else that can be optimized just a little.

7 Literature List

- DeKoven, B. (2002): "The Well-Played Game: Talking About What We Are Looking For". Writers Club Press. Pp. 1-14.
- Hughes, L.A. (1983): "The World of Play: Beyond the Rules of the Game. Why Are Rooie Rules Nice?". Ed: Manning, F.E.. West Point, NY. Pp. 188-199.
- Sass, L. & Oxman, R. (2006): "Materializing Design - The Implications of Rapid Prototyping in Digital Design". Pp. 1-31.
- Sicart, M. (2008): "Defining Game Mechanics". Game Studies. Volume Eight, Issue Two. http://gamestudies.org/0802/articles/sicart