Kauwgombalmisère, the wrap up

-Ten weeks of bubbling straight past the troubles-

The purpose of this essay is to look back at what happened to our source code through the weeks, and reflect on our practical progress with this course. Subsequent to that we will reflect on what we have learned from Software Engineering Methods lab, what we have learned about ourselves as a team of programmers, and how we will use this in the future to design and implement software systems.

In order to start with some context we will describe our progress during the several iterations and comment on the weekly enhancements that we made and why we think it was a proper thing to do at a specific moment. The first decision we had to make was how to realize the graphical user interface. After some consideration with the entire team we chose to use Lightweight Java Game Library (LWJGL), partially because one of our team members had positive prior experience with this library. Driven by both enthusiasm and time pressure the first playable character was set up in just a couple of days. This consisted of a map with walls and some ground in such a way that the ball would be able to bounce up and down in a designated area. As the project was still in its infancy, the playable character was nothing more than a black bar that anticipated on whether or not a collision with a ball would occur.

The next move was to improve the ability to actually play the game. The main feature that needed to be implemented was a projectile that could be used to shoot at the balls and, if successful, let them disappear. After along with other compulsory exercises this was successfully implemented we managed to create something that resembled a real playable game.

To make our creature more vivid, we considered it very workable to add some more levels to the game. It was very easy to do so because of the way we implemented the ability to read an entire level just from one text file. Due the fact that this created different in-game situations we were able to analyze how our game reacted accordingly. As our product progressed more of our time went to fixing things that were exposed by tools that we used such as Travis Continuous Integration(Travis CI) and checkstyle. In spite of how annoying this initially appeared to be, it eventually proved to be a very valuable to acknowledge the exposures because it helped the product to be more consistent and efficient.

Textures were in our humble opinion a great contribution to the game because this significantly influenced the appearance. Along with textures for the playable character and the projectile, pieces of text were added to represent functionalities within the game. This is one of the things we are the most satisfied about because at some point it seemed that we could not implement all things as we had foreseen them in the requirements phase.

To improve the game experience and extend the functionalities we decided to implement background music for terrific gameplay. On top of that an option menu was created that allows the user to adjust in-game settings. By example, the user can state whether or not he wants full screen or windowed mode and alter sound settings. This proved to be very challenging but the implementation of design patterns made this easier. This contributed to our knowledge of how important it can be to follow clear guidelines. As an extension to our texture-use we added score popups and power ups in the game. Score popups display real time the score that is achieved by bursting a ball and power ups allow the player to take benefit from certain changes in circumstances that collecting a power up triggers.

The things we have learned for the software engineering methods lab consist of a variety of things. As this course was not primarily aimed at gaining programming skills but merely at the process around it and effectively and efficiently working as a team we have learned most in that field of study. More than in any project we participated in before, the design of this project was iterative instead of linear. With weekly sprints and evaluations from our supervisor some design decisions were revisited, some even after the implementation. Following on from the theorem we were exposed to in lectures we learned the potential benefits of Responsibility-Driven-Design(RDD). Although we had some knowledge after following the Object Oriented Programming(OOP) and Software Testing and Quality Engineering(STQE) courses it was very satisfying to apply this on something we personally created. We treated our project with awareness of relationships between classes and implement functions in the class that suits the implementation the best. With emphasis on tools such as Unified Modeling Language(UML) we learned that the way classes are implemented and depending on each other can make a huge difference. Also we considered the assignment on design patterns of great educational value as it demands creativity and thorough understanding of how an application is built by a team of programmers.

We will conclude this essay with some reflection on ourselves as a team. As it is the case with all university projects, this again was very instructive. As stated before, this project differentiates itself from others by having more accent on methodological behaviorism between team members separately and as an entire team. One of the main things we took away from the entire trajectory is that it is a fundamental concern to have a start on the project that is pretty solid. At the beginning we had some doubts about the whole "have-your-prototype-ready-in-two-weeks" concept but now we can sympathize with this approach as it proved to be a decent technology. Not only for the start but for practically every software engineering method goes that you do not have to feel provoked by an approach that seems very time-consuming. As in the end you will only benefit from a deliberate construction of piece of software because it makes it far more easy to adapt it to meet certain demands. To conclude this story, this project contributed to improvement of our abilities to properly and effectively work in a team of programmers and accordingly use engineering methods that will deliver a great piece of software.

"Software development is technical activity conducted by human beings" - Niklaus Wirth

Sincerely yours,

Team Kauwgombalmisère