Post Mortem

Overview

An assignment to create a game of choice using a C++ skeleton and shared middleware. This game is called GemShooter and is more or less a copy of bubble shooter. Which was meant to be a little more complicated with different "abilities" and theme design. Jumping into this assignment using a new language and framework proved difficult, and there was much to grasp, understand and learn.

What Went Right

After a few weeks of confusion some of the framework and exercises started to make sense, clicking and connecting. From this the coding side of the project became possible and somewhat enjoyable to work on. The Planning of the GDD and TDD proved useful to look back on and plan the code based on these documents.

Working on SVN was helpful as was the committing function which allowed note taking, about each part of work was done and note which area's still needed work, as well as bugs to fix. Sitting down for a programming session, there was the SVN logs holding notes on what I can work on next or check over for issues.

The project started off slow due to confusion, but soon became enjoyable and work really began adding up, bit by bit the project parts started to form, some had to be redesigned and redeveloped to work. Each mistake was more confusion, but each fix the language became more clear learning became faster and the progression of the project moved faster.

There is still much to learn, and many parts of the code are written poorly but now, if this project was to be done again the confidence and skills developed would provide a much better product overall.

Whenever I had issues that I couldn't solve in a reasonable amount of time I would seek help. This help came from the discussions board in which I asked two questions. Steffan the lecture helped point me in the right direction so I could solve my issue or error. I considered this something that went very right as the answer wasn't given to me, Steffan helped point the right way, or said things in a way that would help me find the answer.

This meant that I learnt the way to fix the issue from identifying it to delving into the code, illuminating parts to find the true cause and fixing it leading to a sense of achievement and better learning. This was done throughout the assignment by myself or with the help of the discussions board and Steffan.

What Went Wrong

There is a long list of things that went wrong, or could have gone better, to begin planning wasn't what it should have been. The game concept wasn't thought out correctly and was to much for the current skills. This was during the time C++/ framework and the exercises was still confusing. With limited experience and skills, the game was planned without knowledge of how the language worked and without how data structures, variables and objects interact with each other.

In the original planning the class diagram included the class “HexGrid” which was to hold all active gems in the game and where any new ones could be placed. This was not researched how to implement in C++. Turns out it could not be implemented the same way as C# or Java which resulted in the removal of the class, and multiple for loops and if statements to copy this. Which moved into vectors without ever being able to implement 2D vectors etc. This issue in planning resulted in many rewrites to get the game to work and much time wasted which could have been used to create a more refined product.

Lack of exercises completed was another issue that cause problems, such as trying to implement the use of text in the game, which created errors, which proved to time consuming in the final days of development that backtracking and removing those implementations had to occur.

The same goes for implementing music and sound, which was left to late in the development cycle and proved to troublesome to implement, which would’ve been different if the exercises was completed in advance.

Lessons Learnt

- Planning is useful, but without any knowledge or skill in the language it is limiting and can cause confusion and going off script later in the cycle.
- Researching how data structures, and objects should / can interact helps when planning to limit going off script.
- Exercises are there to help, completing each of them during the week given would have provided a huge upper hand.
- C++ language is complicated and despite the bad code written, the language is enjoyable and improved my overall knowledge and skill.
- Basic version of what happens under game engines.
- Respect for how complex and how much code is required to create a simple game, including using all the middleware and .libs created by vast amounts of other people.
- How game engines simplify so much and are so powerful allowing much quicker game creation.
- Techniques such as sprites, rotation, game controls, and layering.
- C++ header / source file design.
- Commenting and code design requires a lot of work.
- The backend process of game creation and runtime

Conclusion

At the start this paper, this project was worrying and seemed to difficult I felt there was no chance I would pass the paper and thought this assignment would fail. A few exercises later and working hard I started to understand and learn. This learning was some if not the most enjoyable assignment / paper I have done despite how bad it has turned out.