**Reflection - Project Blizzard**

**Introduction**

In this document I will be reflecting on how the production of my game went. This will be encompassing how I followed my initial plan, what refactoring was done, what testing I used, an evaluation of the final game, what I would change about the game with future work, what I have learnt from the project and how this project will aid in future employment.

**Initial Plan**

For this project I ended up straying from my original plan. I started off originally by following each set out task but as I get further into development I realised that some things were going to take longer than I had originally anticipated and also that some of the more important pieces of work that needed to be done were left for a later time than they should've been. This caused me to end up having to change the order in which tasks were completed so I could have a working game by the deadline.

As well as deviating from my original plan I also deviated from the design of my game, although the fundamentals of what I had in mind for the classes is there, a lot of the classes were meant to work and connect differently. The reason I made these changes was because I realised some things weren't going to be necessary because of how much time I had and also some of the classes seemed redundant in the end, as some of the classes could be replaced by a function.

Also due to the time constraint I ended up having to decide which features were necessary and which weren't for the gameplay. In the end I have a working game but it is missing a lot of features that would've made it a more refined game.

**Refactoring**

Refactoring my code ended up being a large part of my game. I would create quick classes to get the mechanics down and make sure that what I planned for them to do would work then I would refactor the code to be more organised and usable. I ended up refactoring how my projectile took the model it would use so that instead of creating a copy of a model it directly takes a model and doesn't delete the memory. This bit of refactoring reduced my ram usage and also fixed a problem that was occurring within the projectile class with memory allocation.

The main refactoring that was carried out was on how the terrain was generated and drawn. I spent two weeks looking into ways that I could optimise the rendering of the cubes (Number of cubes was over 20,000) for the land and also refactored the generation of the cubes and how they were stored every other week. I started off with a vector of cubes but after a while I realised this was taking up a lot of memory and changed it so that it was a vector of vectors and only has one cube which I drew the land from.

**Design Principles and patterns**

In my project I used an assortment of designed principles and patterns to help make my code more reusable and expandable. I managed to incorporate the factory design pattern within my scene class to help make handling of scenes easier, each scene inherits from the scene class which is an abstract class and need to cover the six functions within the class to work. As well as the scene class my model class also uses this pattern as all objects inherit from model, model isn't abstract though and can be created.

On top of the factory pattern I also used a singleton for my time control class, as it calculates the delta time so it didn't need to be remade every time a class wanted access to the delta time. The scene select class also makes use of the singleton pattern as i needed to make sure only one level was loaded at a time to avoid using too much memory and my scene selection class does this by being a singleton and deleting a scene before it can create a new one.

As well as design patterns I also used some of the design principles in my code. The main principle that i kept too was the Liskov substitution principle and I did this by creating my classes in a way that down casting wasn't necessary in any places throughout my code. I also used the open and close principle with my scene class as it is easy to add in new scenes and new content to them without having to change any of the other classes.

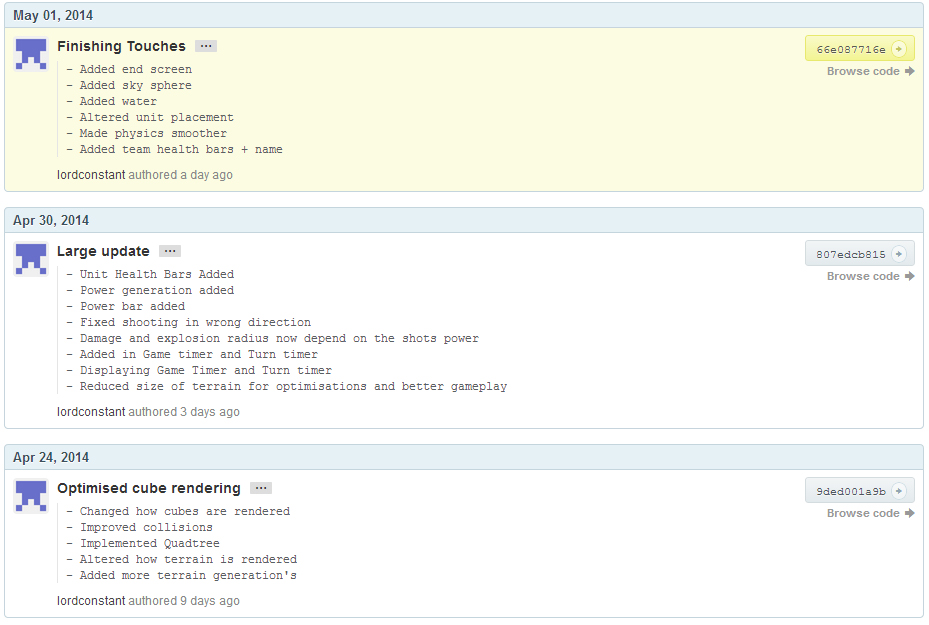
**Testing**

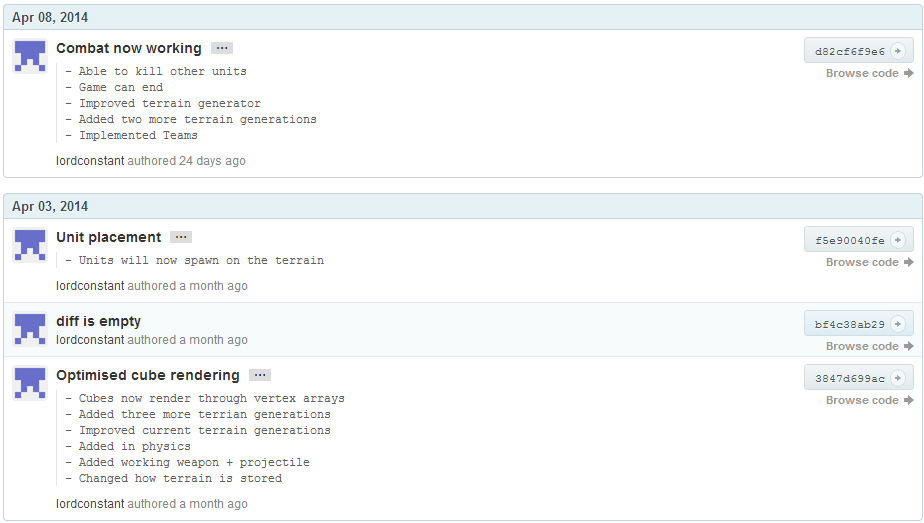
While creating my game I mainly used manual testing to test features in my game as a lot of what I did to start with was setting up my terrain which I created a bmp from the perlin noise to know if what I was doing was working correctly or not. As the project progressed I added in automated testing to some of the classes to check them as they were didn't have as much going on in terms of visual effects and were more number related than anything. I had to use automated testing to start with for my collider as I had needed to make sure it was being set to the right size.

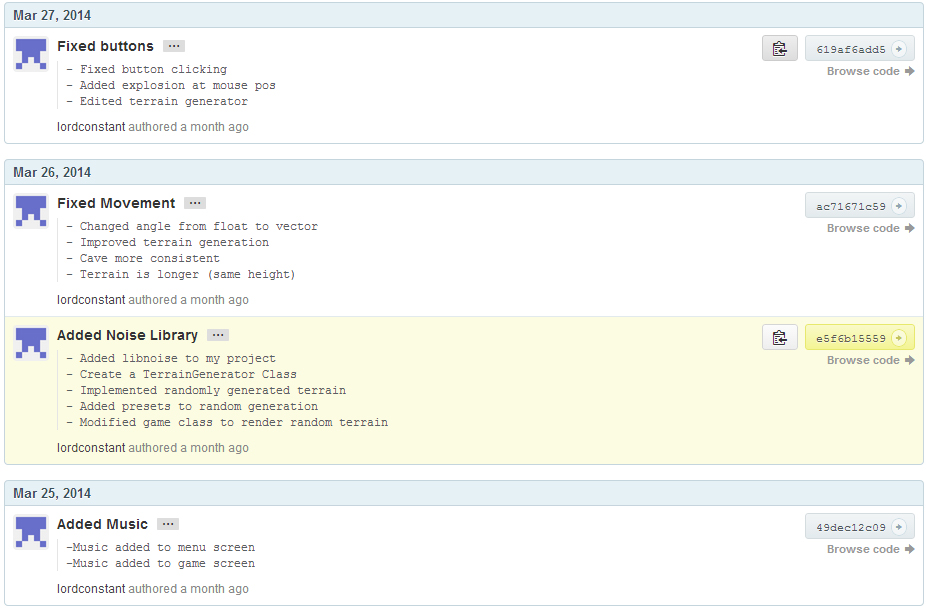
I did a lot of manual testing, continuously running my game to see if bugs were still persisting such as falling through the ground and projectiles firing in the wrong direction, a lot of problems required something to happen multiple times for them to occur where automated testing may have missed this, my manual testing allowed brought these problems to me and I managed to fix them.

**Project Monitoring Meetings**

Throughout the project I had two meetings with my tutor to discuss my progress on the project and what I should make sure I do. In the first meeting we agreed that I had underestimated the workload that I had taken on and should cut out some of the features I had planned, we also agreed I should continue to use github for source control for my game and keep the github uploads relevant.

In the second meeting I was told to stop adding features to the game and make sure the that all the memory was being cleaned up properly and that the game didn't have any game destroying bugs in it.

****

**Final Game Evaluation**

overall I am extremely proud of what I managed to produce for this project. Although it is a bit buggy in places the game runs well, with high frames and low ram usage(less than 100mb). There are a few features that needed refining such as the collision with terrain as you can travel through the side of walls and also the placement of units at the start of a game. I had to leave these features as they were, so I could focus on other parts of the game and get them working so that it would be a game.

In the end I managed to get quite a few impressive features into my game which are as follows:

* Quad tree - I created this for optimising the collision in my game for the explosions and the units colliding with the terrain.
* Vertex Buffer Objects (VBO) - These are used for rendering out the cube class, this was for optimising the cubes rendering, I was going to use glDrawInstancedElements alongside my VBO's but I realised I need to create shaders for this and the project was too for along to refactor something so fundamental.
* Randomised Terrain - I use perlin noise with a randomised seed for the terrain generation. I manipulated the noise to create different types of land, then stored it within a vector of vectors.
* Explosions - I created an explosion class which causes damage based on the units distance from the centre of the explosion, it also knocks the unit backwards depending on the distance as well as destroying the terrain in the area.
* Physics - I created a physics class which is used to move the units around and the projectile. The class applies gravity to objects when they are in the air and you can also set its velocity and acceleration.
* Scene Manager - I designed and implemented my own scene system which allowed me easily and quickly switch between each of the different parts of my game, such as the menu screen, game screen and end screen.

In the final version of the game I made sure that all pieces of work that were outsourced were royalty free this includes the music, images and any library's that I used in the game.

**Future work for the game**

If I were to carry on working on the game I would go back and refactor the code making use of shaders for rendering so that I could implement instanced rendering. I would also expand on the physics class to add in applying forces to an object, as well as a better way to stop physics in a direction. In terms of features I would actually add to the game I would implement the lobby system so that the game could be more personalised and add more weapons as the only one in my game is a rocket launcher. There are a few issues deeply embedded in my game that require me to completely break everything apart to fix them so I would also spend the time and going through to sort these issues as they caused a lot of problems near the end.

**What I've Learnt**

From this project I have learnt how to implement vertex buffer objects and how to efficiently render out large portions of objects through batch rendering as well as instancing. I've also gained a better knowledge of how to use a quad tree as my previous knowledge was based on a binary tree. As well as the technical side I've realised the importance of spending more time on planning out the project to decide on a good way of going through the project so that the main core of the game gets built up. In the future I will make sure to set aside more time for the planning stages of my game and research features more in depth to help decide on accurate time frames for the work that will be carried out in.

**Future employment**

I will be refining the game over the summer to make it more suitable for my portfolio. I will create a video of the gameplay and features, outlining some of the more complex code in the game to show off the work that has been carried out. The reason I wont use a demo is because the game is multiplayer and not single player so it will work better in video form.

For the remainder of my course I plan on refining the skills I have already acquired and expand my knowledge in different areas of programming to help improve the quality of my work and also my employability. I will also take the time to create independent pieces of work that can be used within for my portfolio.

**Conclusion**

In conclusion although I ended up having to deviate from my original plan I did manage to produce a game that was along the same idea as I started with and learnt a lot about optimisations for rendering out large groups of objects as well as the importance of having a detailed and in-depth plan of the project ahead of time so that when something goes wrong you don't need to complete change the plan just alter a couple of things.

I am also very proud of what I have produced at the end of this project and since when I started I had never created a game like this, it was a new experience and really increased my knowledge in a lot of areas of programming.