Post Mortem – Pillage the Village

Team MDJS |

Michael Jordan

2017

Contents

[Overview 1](#_Toc497318116)

[ Map design 1](#_Toc497318117)

[ Screen switching 1](#_Toc497318118)

[ Map selects 2](#_Toc497318119)

[What went right 2](#_Toc497318120)

[What went wrong 2](#_Toc497318121)

[Lessons learnt 3](#_Toc497318122)

[Conclusion 3](#_Toc497318123)

# Overview

My role on the project was to tackle some of the more mathematically demanding tasks like randomly generating a map/track inspired by Spelunky and Map select inspired by Faster Than Light. Since I was the one to suggest these additions to the project, I nominated myself to the task. All tasks could be put into one of 5 sections; Towers, Maps, Troops, Assets and UI. My areas were Map and UI.

## Map design

The design of the map was heavily inspired by Spelunky where the route/track is generated first, tiles are randomly selected to suit the track, and lastly additional flairs are added around the map to make a large permutation of potential maps. My task was mostly a solo task, with sprites created by Jack and I being required to output coordinates for troops and towers.

I’m glad that I didn’t do my original idea of copying my individual assignment and producing an underlay for each tile, on the contrary if I could to this task again, I’d think of a more stable way of sending coordinates to troops. A lot of the early development bugs was the communication between the troop class and the map class.

An alternative would be multiple move methods, where the map would call the appropriate method to move the troop along the track, having methods like moveNorth(int distance), moveSouth(int distance) and moveEast(int distance) would have limited the amount of logic required to determine where to move. This would easily solve the skipping bug with troops as they don’t need to look for a target point.

## Screen switching

My idea was to create a stack like structure of screens which is controlled in the main.cpp. Where at any point a screen could be popped off the stack and resume the previous screen. I used what I’d learned my individual assignment, and made an improved version of the stack. But I still felt that my implementation was rushed and not optimal. It seemed excessive to have multiple loops and logic statements, especially if more screens were added down the development cycle. I mostly felt this struggle attempting to fit a main menu screen into the stack where it previously been only optimised for the logo screen, map select and game.

If I could redo this task, it would be without a time crunch and additional debugging features. If I had more time, I could have optimised the code for additional screens and if I had implemented a way of skipping screens then I wouldn’t have to wait 1 minute per test. I would be wasting my time, watching the troop move to the end of the screen just to test the popping section of my code.

## Map selects

The design of the map select screen was inspired by Faster than Light’s sector map. Where it would display a graph of nodes and edges and only allow the player to select areas of the graph they can access. I was forced to rush this section, hardcoding locations just to get the task done. If I had more time I’d like add more permutation by randomising the maps available to select and expanding the graph well above the current rate.

During my free time, I theorised how I would make this section scalable and incredibly large. I had three methods in mind:

* Expanding from both ends, because the start and end must be one node, I figured that I could expand from both sides, randomising the number of nodes until they meet. Although figured this design would be difficult to balance.
* Predetermined connections where it would randomise in-between each height of graph. Although I figured this wouldn’t have expansive design.
* Randomised nodes and edges, where the number of nodes is initially randomised and the connections between them are also randomised, within reason. This would expand the design of the connections between heights a lot more than predetermined connections.

# What went right

I felt that after the alpha stage was finished, most of the code mergers was completed and anyone within the group had the freedom to impact the game without fear of clashing with anyone on intersecting tasks. Although there were still roadblocks like tasks required to be completed before others could start, everyone at any time still had something to do. I felt that the communication between the group started off quiet but we got acquainted started communicating more frequently. The group was never scared to ask for help when an issue arose and we all pitched in to help.

An example of this was when Shakeel had problems with the explosion class. He looked everywhere for the solution to the bug and offered to look at it. I point out some code design faults and got a solution back to him asap.

I also felt the scalability of my sections, excluding screens, was tremendous. Once the map was fully functional and could be drawn in the different styles it looked good. The map could intake multiple towers and generate a large permutation of designs. And also, being able to see a troop wonder along the track was rewarding to see for the first time.

# What went wrong

Most of the challenges was attempting to merge code with each of the team members. Each team member had a different coding style and ideals to how the code should be handled. Because we tended to do the project solo we couldn’t understand each other’s methods when we would merge the tasks. So, in the alpha stage of development we mainly avoided merging until the last moment which caused unexpected bugs.

I think a better way around this challenge would be to not do a gigantic merge like with the alpha milestone, but to sit down with everyone and merge over a few pc. For the planning of the Beta and gold milestone, I wanted to make it clear where the tasks would intersect.

Another issue was the timing constraint, everyone practically left it to the last few days but we did go into crunch mode. But this mainly impacted the quality of the product as features seemed missing or broken just before release. I decided to cut corner by hardcoding some of my sections rather than looping or using formulas to make a more optimal method.

We also had issues with mysterious code appearing, which after the alpha stage we all agreed to comment thoroughly and only commit working code. Although we still got bugs with committed code, mainly because of different PC builds, we fix them fast.

# Lessons learnt

I think that next time I’ll prioritise the merging of the tasks first, because it was a struggle trying to get the alpha stage together. If we had prioritised the merger we would have realised the issues within our code earlier and wouldn’t have taken as long to fix them.

Adding more debug features, to speed up the time testing/fixing bugs. Just any attempt to drastically reduce the wait in-between tests impacted how much work we completed. Earlier on in development all my files were stored on a usb drive which increased the compile time from 5 seconds to a full minute. Just every second in-between tests started to add up.

I’ve also learned from my group that it’s not entirely bad to use hard code. I’d used to waste time, pondering and implementing a loop or formula to complete tasks just to realise my code has a bug. If I initially use hard code to test out smaller feature, I could fine these bugs beforehand. I started to implement this with map select screen to save time, although I didn’t have enough time to refactor my hard code, it still proved that my code worked.

# Conclusion

In conclusion, I think that this project was a great exercise because, up until now I’ve only done solo projects. Working with this group was enjoyable and thrilling seeing everyone else’s code all work together. Just being able to focus on one area of the project removes the workload. Seeing other people methods and ideas really opened my eyes to the multitude of solution. I always challenge myself in terms of algorithm design for my tasks, but it was thrilling seeing everyone else’s code all work together.