Project Report

BACHELOR OF SCIENCE (HONOURS)

Computer Games Development

Author: Joseph O'Keeffe

Student ID: C00258019

Date of submission: 29/04/2024

Table of Contents

Acknowledgments

I would like to thank Dr Noel O'Hara for being my project supervisor and helping me throughout the year

Project Introduction

Overview

My project is an RTS (Real-Time Strategy game) game called The Upper-Cliff Valley built in C++, SFML and Raylib. Real Time Strategy games allow players to play simultaneously, in "real time" against their opponents, where as in Turn-Based Strategy games, the player and enemies take turns to play.

SFML and Raylib

For this project, I used SFML (Simple and Fast Multimedia Library) to make the game itself and I used Raylib to make the level editor. I decided to use SFML to make the main game because I've been using it for a while and know it well. I would be working on the main game a lot more than the level editor, so SFML was the only choice for me.

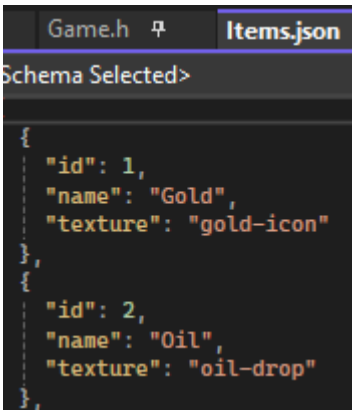
SFML lets you easily draw sprites in 2D, has an easy-to-use audio system, its fast and easy to use. SFML also has lots of documentation which came in useful when I got stuck on any problems.

Raylib was a bit harder to use, as everything seemed to be done in functions and using parameters. You create a shape, give it a texture and position and colour all in one line. After that, I couldn't seem to manipulate that ship every easily. Where as in SFML, you just create an object and then you can do whatever you would like with it whenever you want.

Key Features

Here are a few key features from my project:

- **A* Pathfinding –** I used A* pathfinding in my project for the units and enemies to navigate around the map while avoiding obstacles
- **Level Editor -** I made a level editor in Raylib that lets you create custom levels, and you can save them into a JSON file and then load them into the actual game
- **Item manager –** I have an Item manager class, that loads items from a JSON file. This is then used to check if an item is in an inventory



- **Inventory System –** I have an inventory system that lets you sort the items by quantity and ID. The inventory is made of inventory slots. Each slot is made up of an Item, a bool and a shape which displays the item. When you open your inventory, it displays the item you have with its quantity


```
Item(int id, std::string name, int quantity, sf::Texture& texture, std::string textureName)
```

- **Creating units and buildings –** You can create units and buildings. You need gold to create units. There are two types of units. Workers and Fighters. Workers gather resources and fighters attack the enemy.
  Buildings can be placed for free, but they can only be placed on a certain tile. Mine can go on gold resource tiles, Oil extractor can go on oil resource tiles and uranium extractors can go on uranium resource tiles.
- **Resource gathering –** I have a system where you need to assign a worker to a designated building to get resources. They collect the resource for a while and then deposit it back to the base. The more workers at a building increase the gather speed and how much you earn.
- **Saving/Loading with JSON – I**n the main menu, you get the choice to start a new game (one of the levels you created in the level editor) or continue game, where you can play a level that you have previously saved. Nearly everything in the game is saved to a JSON file. Unit and enemy positions, inventory, enemy bases left and more.
- **Shaders –** I have shaders on the Headquarters, so it pulsates red once it is selected. I also have them on items that are on the ground, that make them glow so its obvious to pick them up.
- **Particles –** I have made custom particles for when you select a unit, units and enemies are walking, units firing projectiles, units are working, buildings getting destroyed, pause menu, unit or enemy takes damage and probably more.
- **Parallax background –** In the main menu , the background moves in parallax, and it looks really nice.
- **Tutorial book –** I have made a tutorial book type thing that has pages. It loads from a text file and its very easy to add new pages, all you need is a title, paragraph and texture name.
- **Enemy intelligence –** Enemies have intelligent behaviour. They search for three things. Player units, player buildings and enemies that they can merge with. Once they've found one of them, they move towards it
- **Enemy merging –** When two enemies of different types are close together, they start merging together. This creates one much bigger enemy with more health and damage.
- **Texture loading –** To load textures efficiently I've made a texture loading class which uses the singleton pattern. This loads all the textures once at the start of the game. They get stored in a map of strings and textures.

```
warriorTexture.loadFromFile("./assets/images/warrior.png");
textureMap["warrior"] = warriorTexture;

archerTexture.loadFromFile("./assets/images/mage.png");
textureMap["archer"] = archerTexture;
```

So, if I want a texture, all I do is call this function with the name the texture and I can access it

```
sf::Texture& Textures::GetTexture(const std::string& textureName)
{
    return textureMap[textureName];
}
```

- **Sound loading –** I have done the exact same thing but with sounds. This way, I will have an easy way to access them whenever I want. I also have added lots of sound into the game. A lot of the sound I made myself

```
menuMusicBuffer.loadFromFile("./assets/sounds/menu.ogg");
soundBufferMap["menu"] = menuMusicBuffer;

backgroundMusicBuffer.loadFromFile("./assets/sounds/game-background.ogg")
soundBufferMap["background"] = backgroundMusicBuffer;
```

Evaluation and Discussion

I will evaluate and discuss a few things from my project. The first thing will be the use of A* pathfinding. I think it could have been better. Currently, the units and enemies move from the centre of one tile to the centre of another tile. This made the movement clunky and not as fluent as I would have liked. Ideally, what id have liked to implement would be some way to avoid all the obstacle tiles but move to the exact point the mouse is within the tile you want to move to. Not the centre of that tile.

The second thing I will discuss is the inventory system. When you pick up items or gather them from workers, it will be added to your inventory. You can sort items in your inventory by ID or Quantity. The use of this is to see what items you have gained. There is one slight issue with it, sometimes the items in your inventory can break, and the texture won't appear right, and sometimes duplicate items appear. I'm not too sure why that is but it's something to look in to.

The third and final thing I will discuss is the overall gameplay from the game. I think the gameplay is good but there are a few bad things about it. Like I said above, the A* makes the movement feel clunky. Once you select a unit and then move them, there is no stopping them. They keep on moving until they've reached their destination. Another thing is, units only start attacking enemies once they have stopped moving, whereas enemies stop moving once they detect a unit. This gives the enemies a major advantage to get some attacks in before the unit can do anything

Project Milestones

I didn't necessarily have any milestones. I just wrote a to-do list in one of my classes with comments and chipped away at those. If I was to think back and create milestones (also looked at my commits) they would be as follow:

- **Milestone 1 –** Tile map, Level Editor saving and loading from JSON
- **Milestone 2 –** Units, basic movement, animations
- **Milestone 3 -** Creating buildings and some UI

- **Milestone 4 -** Unit grouping and behaviours
- **Milestone 5 -** Worker units who gather resources
- **Milestone 6 –** Added more buildings and units
- **Milestone 7 –** A* pathfinding and enemies, particle system, pause menu
- **Milestone 8 –** Projectiles, enemy merging, units and enemies can take damage
- **Milestone 9 –** Destructible buildings, inventory, mob drops, shaders
- **Milestone 10 –** Inventory sorting, spaceships, enemy spawning
- **Milestone 11-** Enemy bases, instruction menu, win and lose screen
- **Milestone 12 –** Added sounds

Major Technical Achievements

- **Inventory system**
- **A* Pathfinding**
- **Loading/Saving from JSON**

Project Review

What went right?

I think the project as whole went well. Creating units and buildings was a lot easier to do than I expected and that all went smoothly. Loading and saving from JSON was straight forward. A* pathfinding was also good because we have done it a few times before. The inventory system went well and being able to create new items in a JSON file. The instructions menu turned out better than I expected, and I loaded them from a text file, so it was easy to add new pages in that

What went wrong?

I feel like implementing the A* pathfinding had a slightly negative impact as the movement was a bit more clunky and not fluid. I could have done it a different way so it could avoid obstacles but have fluid movement still. Formations didn't go that great as I only implemented one of them and I spent lots of time trying to get it work with A* and then scrapped that idea.

Is anything missing from my project?

I don't know if I would go as far and say anything is missing. I think I managed to make a complete game and I had enough time to do most things I wanted. There are a few things I would have liked to add into the game if I had more time

- Formations are definitely something that I didn't get around to finishing. I would have loved to make proper formations that make units stronger or have different abilities. Maybe having an indicator that tells the player who is the leader of the formation.
- I would have added more enemies. There was only one type of enemy in the game, and they only had one attack so having a variety of them that did different things would have been good.
- I would also have liked to add different attacks that do different things. Maybe an attack that heals units, an attack that stuns enemies etc.
- Maybe a level up system. When you kill enemies, they give you XP and you can level up. XP is already in the game but it doesn't do anything so that would have been nice.

- Building upgrade system that increases the amount of resources you get.
- Maybe add some sort of shop in the game, which would allow you to do more things with the items you get throughout the game.
- Maybe flesh out the level editor, so you can create enemy bases in it and basically make the whole map within that and not just the tiles.
- This is potentially a bit of a stretch, but this could have been multiplayer. So instead of the enemies its just another player that is playing against you.
- Level progression, so once you beat the first level, the second level is harder

How would I approach the project if I started again?

If I was to start this project again, I think I would scrap the Raylib level editor and just implement it in SFML. I think this would allow it to be much more flexible and I could do more things with it

Potentially use top-down sprites as it would require less animations

Know exactly what I want to do before starting it. I just came up with things as I went with this project, but I think if I knew exactly what I wanted I could have made things better

I would create better systems for creating particles and projectiles that would allow me to customise them a lot easier

I would add a wider variety of enemies and units that do different things and act differently

What advice would I give to someone starting a similar project?

My advice would be to have a set plan of what exactly you want to do. Also, plan out your ideas and milestones and cross them off as you go. Do some research and investigate other games like this and take note of the features in them that you could add to your game.

Try find the sprites you want to use early on, as finding sprites can take up a decent chunk of time

Did I choose the right technologies? If not, justify why.

I think with SFML, yes, I did pick the right technology. Raylib on the other hand I'm not too sure.

I think Raylib might have been a bads choice because I was not used to it and I didn't really like how everything was function based compared to object based

```
DrawTextureRec(spriteSheet, sourceRect, { resourcePaletteRects[i].x,  resourcePaletteRects[i].y }, WHITE);
DrawRectangleLinesEx(resourcePaletteRects[i], 2, BLACK);
```

So this draws the rectangle, but its very hard to maniplate it and customise it as its all done in one function. This as well as not being familiar with the language, I think I would have stuck with SFML if I was doing this from scratch

Doing this in a game engine such as Unity might have been much easier. Unity has lots of features that would make creating a tile based game more convinient than doing it in SFML. Unity has a tilemap which allows you to create custom tiles. Tilemaps have different layers, so you can set one layer to be fully walkable and then have an obstacle layer that nothing can pass through.

Unity also have prefabs which allow you to instantiate a custom game objects that you have made. This would make creating units, buildings and enemies so much easier.

Unity have scene management, which makes changing from one scene to another very easily, so you don't have to worry about other things going on in the background

Unity has a very easy to use UI system that places UI elements on a canvas that doesn't mess up anything else in that scene this would have made my HUD and inventory much easier to create.

Overall, making this game Unity would have been much easier but that doesn't necessarily mean better. I am happy with picking SFML as Unity does too much for you

What were the implications of your technology choices?

The implications of using Raylib would be that I felt limited to what I could do with my level editor. I got it working and I just left it at that. Ideally, I would have liked to get to know the library a lot more, but I thought the best choice would be to work on the main game. By using Raylib it limited what I could do but I feel like I learned a new library a good bit in the process, and I think it turned out good considering I've never used it before

Conclusions

I felt like I learned a lot from this project compared to other projects that I've done through my four years. In most of my other projects, I was paired up with someone and I could rely on them to do the things I wasn't so sure of. In this project, I was doing it all by myself. Because of this, I had to learn lots of new things. I finally got the chance to understand how pointers worked and how useful they can be. I never understood them before because I never found a reason to use them, but in this project, I needed to, and it was extremely useful.

Another thing I learned was using std::vector instead of arrays. This was very useful because I needed the sizes to be dynamic as there wouldn't be a set size of something such as units in the game. Also, being able to remove elements from the vector and insert elements into the vector whenever I needed.

Future Work

If I was to continue working on this, I would add more enemies which have different behaviours

I would also make the enemies more intelligent and maybe have some sort of base building mechanic where the enemies would make their own buildings similar to the player

I would make a progression system so that there are different levels, and each would get harder as you progress

Something I would really like to do would be recreate my project in unity and see how that would go. I think it would be very interesting and enjoyable

References

Dragon Ball – music and sprites

Battalion: Head 2 Head – sprites

Nlohmann JSON for C++ - https://github.com/nlohmann/json