

Faculty of Arts, Humanities and Social Sciences

Cambridge School of the Creative Industries

MOD008616

Software Engineering for Games

Reflective Commentary

SID: *[2307552]*

Academic Year 2022/23

# 

Contents

[Reflecting on your Process 2](#_Toc114491476)

[Implementation 2](#_Toc114491477)

[Feature/skill/section X 2](#_Toc114491478)

[Issues faced 2](#_Toc114491479)

[Critical Evaluation 4](#_Toc114491480)

[External Assets 5](#_Toc114491481)

[References 6](#_Toc114491482)

# Reflecting on your Process

## Implementation

### Project Overview

When the details for the project were first outlined ‘This task aims to see how successfully you can recall and implement aspects of the C# programming language to build a mechanic that was common in older games - A combat mechanic with a focus around probability to hit and modifiers!’ (ARU Canvas, 2024), my initial idea was to try and create a combat system that, on some level, replicates the combat of the roguelike space game ‘FTL: Faster Than Light’. I liked the idea of making a customisable spaceship that can go into combat with a standard setup, win a fight and gain some currency to then upgrade the ship and improve the combat effectiveness. The probability to hit was initially going to include weapon accuracy, distance between shooter and target, size of target component, enemy dodge chance, and the shooters proficiency with firing the weapons. The submitted build did not include the shooters proficiency with weapons due to time constraints.

Github was used to help control of the project in the initial calculator task, and this VATS style project. I found it very useful to be able to see changes made on a given day, and for transfer of files between my desktop pc, laptop, and university machines, additionally it is useful for tracking the changes in the project to remind myself of the processes I went through in the earlier stages of the project and for writing this document.

### Timeline of implementation

For the first few weeks I was struggling to work out where to begin, I couldn’t think how to build a game with UI that is game-like in presentation, then in one of the lectures we were shown the *Console.SetCursorPosition* command, this was a huge lightbulb moment in my head, and it gave me enough context to begin building.

The first code written for this project was to draw a spaceship created with ASCII art, and to draw a border. Initially the spaceship was a single unconfigurable spaceship, and the border just had vertical and horizontal characters to draw the sides of the border, without using corner pieces it looked a bit strange, so I used an online ASCII character helper to find the character codes for the pieces to tidy it up.

Next I thought what I wanted the game screen to look like, I decided on three sections of the screen, I called these the action panel, menu panel, and stats panel. The image below shows how this looked

A screenshot of a computer

Description automatically generated

Figure First panels to be drawn

Now I have these menu elements, The first big square will be used to show the action of the game – the combat between the player and enemy ships. The stats panel on the right will be updating on the changes to the stats to the player and enemy (e.g. health etc), and the menu panel at the bottom will give the player options on how to interact with the action panel.

Once this was done, I started to think about what the flow of the application would feel like. I needed a title menu and a place to customise the ship. So, I used an online ASCII art generator to draw out a logo for STL, and created a method to draw random ‘\*’ characters to be stars in the background, and created the initial main menu.

A screen shot of a computer

Description automatically generated

Figure Main menu of STL

I also created a method to draw the enemy ship which is a drone. If I had longer in this project I would have like to implement a variety of enemy ships, that were randomly selected and got stronger with new mechanics etc as you replayed through the levels giving more meaning to the customisation and power improvements of the player ship.

Next, I worked on building the hangar – this took a fair amount of time as there was a lot of work to do in the background to allow the functionality I wanted. Initially I created the player ship as an object and then tried passing it around to the different areas of the game, but eventually defaulted to using a static class to manage the player ship. This propensity to using static methods had been a regular theme in all my coding up until this project, despite that I did start using OOP for enemy ship.

For the hangar I started with the ability to customise the components of the ship, so I drew out (and redrew a few times as I was repeatedly unhappy with how they turned out), three options for each of engines, hull, and weapons. This created a modular player spaceship that can have a different appearance based on the chosen components. After this I added in stats to the player ship, giving health, dodge chances, accuracy, armour, and component sizes. I also added in health per component, so the engine had its own health stat, and the weapons had its own and so on, this was going to be used so that if you shout out the engines of the enemy ship then it would have it’s dodge chance reduced as it cannot evade, and if you shot the weapons out the enemy wouldn’t return fire for a turn or two until it repaired them. This was cut from the submitted build due to time constraints, I did however continue keeping the total health calculation off the components, so that if you upgraded the engines, you would still see some health increase.

The last part of the project I implemented was the combat loop with targeting and chance to hit, this strategy of implementing the briefed functionality of the project is not the most effective way of building the project but it was the part I did not have rationalised in my head for a while.

I wanted the chance to hit the enemy to be fairly complex and to include all of the factors mentioned earlier. So, I modelled out some equations in excel and spent a few hours and session revisiting the equation and rebalancing some things. I think it is not perfect but in the interest of time I settled with it. The equation is below:

Di = distance between player and enemy ship

S = size of targeted component

A = Accuracy of weapon firing

Dc= dodge chance

**Chance to hit = 1-(1-1/(sqrt(Di)\*5)^(0.4\*sqrt(S)))\*A -Dc**

The sqrt on the distance and component size were to prevent a size 6 component being massively out scaled compared toa size 1 component.

Once this was done it was just a matter of adding in the sequence for a round of combat: player takes turn, check enemy is alive, update stats, enemy takes turn, check player is alive, update stats. The sequence ends when either player or enemy is dead.

### Feature/skill/section X

### Overall design

I try to keep my main method as simple as possible, only showing the general flow of what subsections of the application to navigate to. When going into a new menu, it is also fairly intuitive and easy to read with all the legwork being done in other classes and methods referenced. When entering a menu, once complete you should come back out of it before moving to a different part of the app to minimise the call stack.

## Issues faced

### My psychological approach

One of the big issues I face when completing project work is managing my efforts and spreading the workload out over time. I have ADHD and when I get assigned a project with a deadline, I would scope the project and determine how much time it would take to complete the project, then not do work on it until I start to feel the stress of the timeline. I also put off the parts of the project that seem less fun to me. This applies to a project level, and a course level. I enjoyed this modules coding project more than the unreal project from Development for Game Engines, so at this point I have only done around 15 hours work in my unreal project and many more in this VATS project.

### Structure

When building projects, I usually start building everything in the main method, then filter it out into its own class once I have enough context for the build. This started out well and nicely structured but nearer the end of the project I feel like it has gotten more disorganised. I would benefit from having a more rigid plan of the overall structure before building so that I can keep things organised.

# External Assets

Factory pattern structure: This was adapted from the following source for use in building the SpaceshipFactory [*https://www.youtube.com/watch?v=SLEu6rNdJj0&ab\_channel=campbelltech*](https://www.youtube.com/watch?v=SLEu6rNdJj0&ab_channel=campbelltech)

ASCII art: Logo, Game Over, and Victory screens all were generated using *https://patorjk.com/software/taag/#p=display&f=Graffiti&t=STL*

# References

ARU Canvas, 2024. *010 Element 100hr project (2024 MOD008616 TRI1-2 F01CAM).* [Online]   
Available at: https://canvas.anglia.ac.uk/courses/41871/assignments/402213  
[Accessed 4 42 2024].