

Industrial Technology: Multimedia

Student Candidate 3223 7878

For my Year 12 Multimedia major project, I plan on developing a complete video game through Unity. The video game will include fun gameplay, intuitive controls, and complex kinds of AI or procedural generation.

MAJOR PROJECT
PORTFOLIO

CONTENTS

Statement of Intent	4
Research	5
Processes	5
Planning:	5
Prototyping:.....	5
Production:	5
Evaluation:	5
Technologies	6
A Game Engine	6
Unity.....	6
An Integrated Development Environment	7
Choice:	7
Screen Recording Software.....	7
Materials.....	8
Resources	8
Models and Textures	9
Planning	10
Ideas of potential games.....	10
Modified Chess game.....	10
Virtual Reality Spaceship.....	10
Fantasy Role Playing Game (RPG)	11
Idea justification.....	11
Comparison of Existing Products	12
Beat Saber	12
Guitar Hero	12
Faster Than Light	13
Use of information gathered from Existing Products	13
Playing Styles.....	16
Project Proposal	17
Challenges I'm anticipating are:	17
Revised idea	17
Schedule for Development.....	18

Sequencing	18
Project Management	20
Issues Encountered.....	20
Gantt Chart.....	23
Risk Assessment	24
Finance Management	26
Prototyping, Modelling and testing.....	27
Game Functions	27
Test Computers	28
Test Builds	29
V1 – 13/02/2020.....	29
V2 – 16/02/2020.....	29
V3 – 20/02/2020.....	29
V4 – 01/03/2020.....	30
V5 – 14/05/2020.....	30
V6 – 28/06/2020.....	30
V7 – 12/07/2020.....	31
V8 – 20/07/2020.....	32
V9 – 11/08/2020.....	32
V10 – 22/08/20.....	33
V11 – 25/08/2020.....	34
Artificial Intelligence	35
Possible Types of AI.....	35
Considerations.....	37
Final Choice	37
AI Navigation.....	38
Visual Design.....	39
Visuals to source	39
Spaceship Models	39
Space Backgrounds	41
Explosion VFX	42
Visuals to develop.....	43
UI Design	43
HUD Design.....	43

IT Multimedia – Spaceships!

Evaluation	44
Skills Acquired.....	44
Progressive Diary	46
Before Term 4	46
Term 4	47
2019-2020 Holiday	51
Term 1	52
Term 2	56
Term 3	67
References	73
Final Evaluation	76

STATEMENT OF INTENT

For my Year 12 Multimedia major project, I plan on developing a complete video game through Unity.

The video game will include fun gameplay, intuitive controls, and complex kinds of AI or procedural generation. My previous experience programming for the Robotics team will assist me in programming scripts in Unity, and my interest in video games will give me the knowledge necessary to create an engaging and entertaining game.

My interest in video games is why I want to make one, as I now feel I have the skills to contribute to the video game community. Several years ago, in primary school I was introduced to online flash games, I enjoyed discovering new games and competing with my class mates. Ever since then I have been discovering new games and learning about them.

My experiences with video games will allow me to take the parts I like out of my favourite games and put them into my own game, of course with my own creative twists and ideas.



Image Source: <https://nikithalifestyle.wordpress.com/2016/11/24/wow-my-childhood-favourite-computer-game/>

To make a video game, requires a lot of time and resources. I have spare time, and I have many of the resources necessary, such as a computer and programming software, but I don't have the same budget as a video game development business, and I won't be able to work on this game full-time.

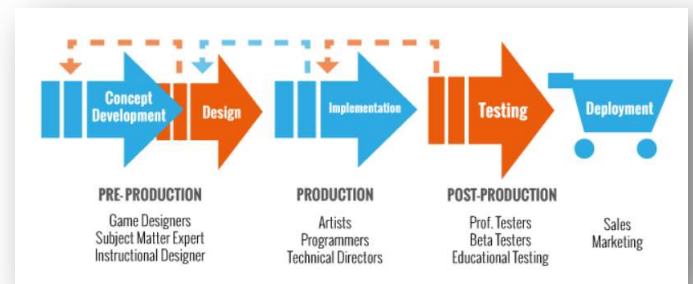
Because of this, I intend to design the game in a way that minimises the need for modelling, graphic design, and sound effects, which are time consuming things I'm unfamiliar with. I instead intend to focus on the game's mechanics, such as including AI, and procedural generation. Creating a game with these mechanics will create variety in gameplay and give the game depth. By designing the game to create maximum playtime with minimum design time, I can create a full-scale game with the limited time and resources I have.

RESEARCH

Before I can plan or develop my game, I need to know about what is required to make video games, and how it can be done. This will include researching other video games, researching game development engines, and learning the programming languages they use.

Image Source: [https://r-](https://r-stylelab.com/company/blog/mobile-technologies/how-much-does-mobile-game-development-cost)

[stylelab.com/company/blog/mobile-technologies/how-much-does-mobile-game-development-cost](https://r-stylelab.com/company/blog/mobile-technologies/how-much-does-mobile-game-development-cost)



PROCESSES

These are the different stages of game development and what they involve:

PLANNING:

- Planning – The planning phase is the first and most important stage, it's where you develop ideas and decide what you want your game to be. This stage requires research of other games, research of game development, and the development of ideas.

PROTOTYPING:

- Prototyping – The prototyping stage is where you make a basic version of the game to see what you can and can't do, and find issues in the game plan. The prototype is the framework of the game, and everything after the prototype is completed is built off the prototype. This stage mainly requires programming, there's no need to give the game art and design just yet.

PRODUCTION:

- Production – The production phase is the largest, it's where you work towards a final product. It has multiple tasks that can be completed in any order. The completion of the framework allows for the rest of development to begin.
 - Design – Design is the first part of production, now you can begin giving your game a look. This involves making textures, models, designing the UI, and creating/finding sounds and music for the game.
 - Level Creation – This is where you piece together all the different aspects of the game. You have to think about the player's experience, and create challenges and ways for them to be overcome. Level creation involves more planning, then more programming.
 - Programming – up until this point a lot of programming has been done, this stage is where you clean the code so it runs smoothly, work out bugs and errors in the code, and implement new features as necessary. Programming involves more programming.

EVALUATION:

- Evaluation – This stage is where you decide if your game is ready to call finished or not. You have to test every feature, and judge the games style and feel. If the game isn't ready, you have to return to the planning stage to figure out what must be done, and continue through to evaluation again. Evaluating involves playtesting and reviewing.

TECHNOLOGIES

There are three main technologies I need to develop a game; a game engine, an IDE, and screen recording software

Alternatives available

- A game engine that will run the game's timekeeping, physics, collisions...
 - Unreal Engine
 - Unity
 - GameMaker Studio
 - A homemade engine
 - Cry Engine
- An Integrated Development Environment, where I can code scripts for the game
 - Visual Studio
 - Eclipse
 - Notepad++
- A screen recording program so I can record my gameplay for demonstration purposes
 - NVIDIA Shadowplay
 - Xbox Game Bar
 - iSpring Suite

A GAME ENGINE

For a game engine, my main requirement is that it's user friendly. I don't have a lot of experience with developing in game engines and I'm looking for a game engine with a good support system, and sufficient documentation so that I can learn the engine.

The engine that best fulfils these requirements is Unity.

UNITY

Unity is a powerful free engine (they only take a % of your revenue if you sell your game, which I don't plan on doing) that has an extensive support system to help new and experienced users. Unity Learn has a library of official and user made tutorials on many different topics. Unity also has a great deal of documentation: they have an API for their scripting functions (there's a LOT of functions), and a user manual for engine functions. Unity also has a very large community of users who share questions and answers on Unity forums, and post tutorials on YouTube.

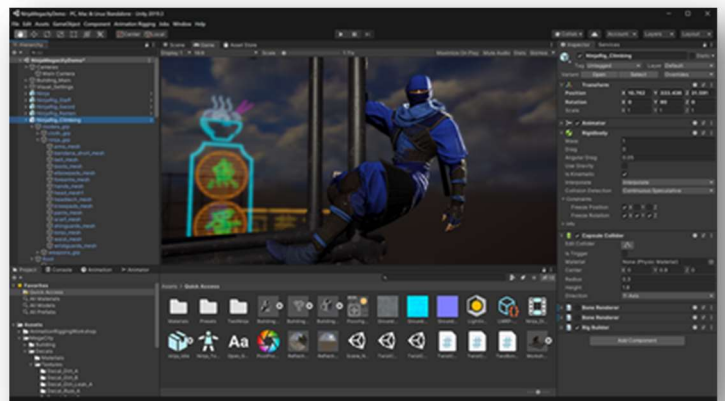


Image Source: [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine))

I also need to choose a version of Unity to use, Unity offers new versions multiple times a year, but only the most reliable and commonly used versions are put on Long-Term-Support (LTS). At the time I downloaded Unity (15th Dec 2019) the most recent version put on LTS was release 2018.4.14f1.

AN INTEGRATED DEVELOPMENT ENVIRONMENT

The IDE I use mainly depends on the game engine, some game engines don't even require an IDE, while some provide their own. Because I've chosen to use Unity, I need an IDE that supports C#.

CHOICE: Unity recommends **Visual Studio 2017**, this has been made the default option for the version of Unity I have chosen to use.

Visual studio is a good option as it's compatible with any programming language (if you download the right extension), it has a highly customisable layout, and it features a code-completion aid called Intellisense.

Intellisense will read the line of code you are writing and offer a list of all the potential variables or functions you could write next, with a brief description and list of parameters. For these reasons I've chosen to use Visual Studio 2017.



Image Source: <https://www.kunal-chowdhury.com/2018/08/download-visual-studio-2017.html>

SCREEN RECORDING SOFTWARE

The screen recording software needs to record in a high quality and high framerate, be compatible with games (some recorders have issues with fullscreen games), and would ideally be free.

Choice and justification: NVIDIA

Shadowplay best fits these requirements, and additionally is capable of saving in many different video formats. Xbox Game Bar records in a lower quality, and can only export as a WMV, and iSpring Suite records well but is more complicated to use and costs money. Bandicam is free, but records at a low quality and leaves a watermark on the recording.

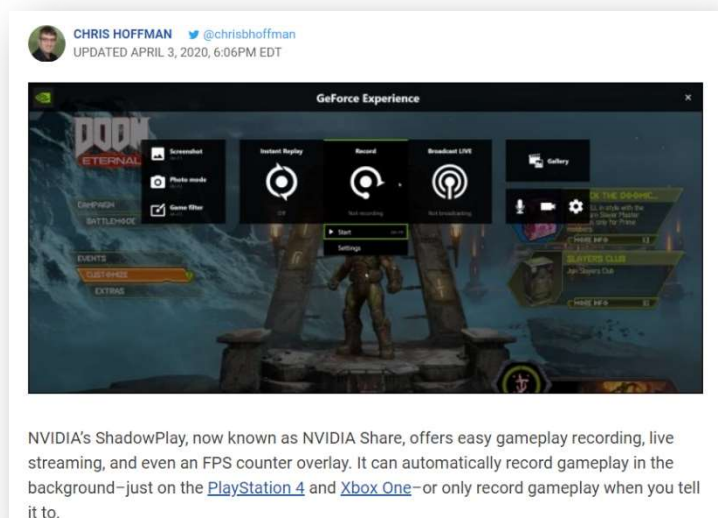


Image Source: <https://www.howtogeek.com/259573/how-to-record-your-pc-gameplay-with-nvidia-shadowplay/>

MATERIALS

To develop or run a video game, you need a computer of a certain standard.

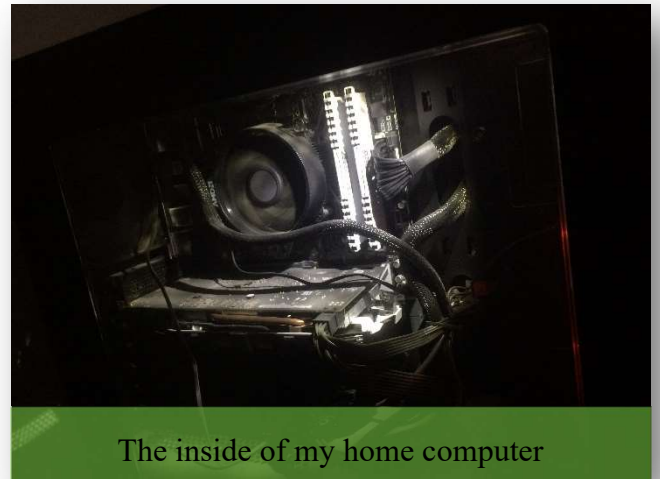
The minimum hardware requirements for the version of Unity I intend to use (2018.4.14f1) are:

- OS - Windows 7 (SP1+) and Windows 10, 64-bit versions only
- CPU - X64 architecture with SSE2 instruction set support
- GPU - Graphics API DX10, DX11, and DX12-capable GPUs
- Additional requirements - Hardware vendor officially supported drivers

My home computer has the following specifications:

- OS – Windows 10 Home, 64-bit
- CPU – AMD64 architecture with support for SSE2 (Ryzen 5 2500)
- GPU – DirectX version 12.0 (GeForce GTX 1060 6GB)
- Additional requirements – Running officially supported drivers

Because I already own the necessary hardware for game development, I will be able to install the necessary software and use this computer to develop my game.



The inside of my home computer

RESOURCES

Video game development requires resources like any other kind of development, but mostly in the form of information. I'll need places where I can get certain information.

Because I've chosen a highly popular and long-standing game engine, I can use Unity's extensive support system to find out anything I need. YouTube videos are another excellent source of information, there are many different tutorial series I can follow to learn about programming languages, Unity, game development, etc.



Unity tutorials on YouTube

Image Source:

https://www.youtube.com/results?search_query=unity+tutorials

MODELS AND TEXTURES

Another important resource is the models and textures I will use for my game. While it is possible for me to design them myself, this can be extremely time consuming and difficult, and I likely won't have time.

There are so many free models and textures available on the internet that I can use, and so many more that I can purchase for small sums that I will be able to find what I need with relative ease. These are some of the websites I can browse to find the models and textures I'm looking for:

- <https://free3d.com/3d-models/>
- <https://www.turbosquid.com/Search/3D-Models/free>
- <https://www.cgtrader.com/free-3d-models>
- <https://sketchfab.com/features/download>
- <https://3dexport.com/free-3d-models>
- <https://clara.io/library>

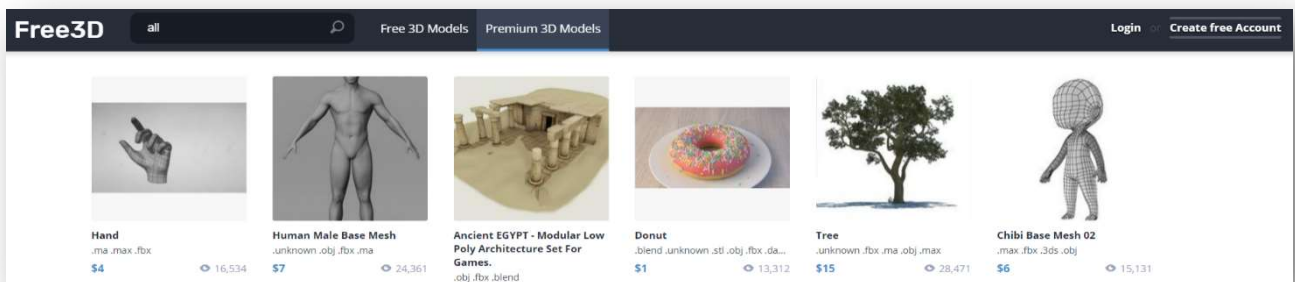
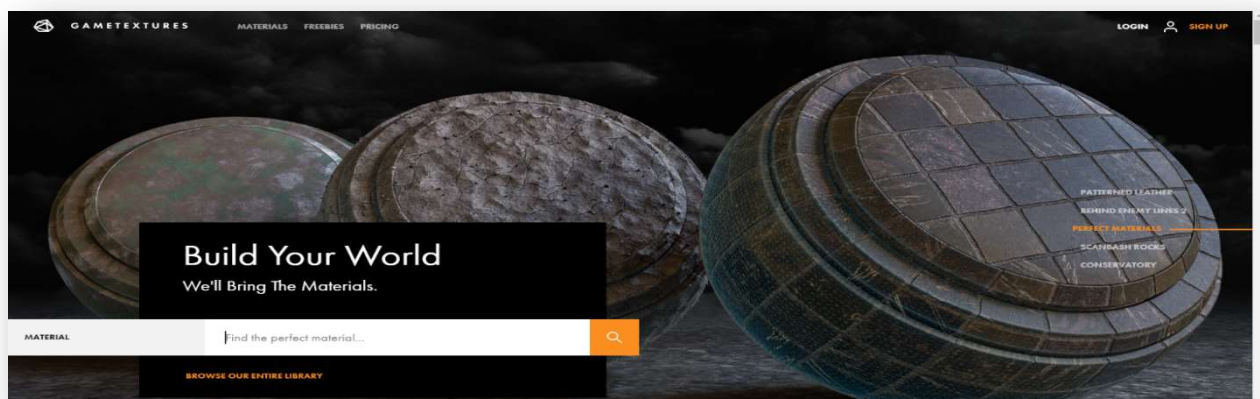


Image Source: <https://free3d.com/premium-3d-models/>

Their contents are all produced by their communities, so the website I get the models and textures I use from isn't relevant. The only thing I should look out for, is that the models and textures I choose match each other's style. I can't have space age laser guns mixed in with steam punk air balloons,



so it would be ideal if I could get most of my resources from one group of creators, or at least labelled under the same theme.

Image Source: <https://gametextures.com/>

PLANNING

This section will include the use of research I have completed in generating game ideas into a game I can develop. Now that I know how I can make the game, I need to decide what I want the final product to be, and figure out I can get there.

IDEAS OF POTENTIAL GAMES

MODIFIED CHESS GAME

Modified chess game, where pieces gain abilities upon taking another piece. Game would begin as a normal chess game, and when one piece takes another, that piece would be given a new ability. A pawn taking another pawn, for example, would allow that pawn to attack pieces directly in front of it as well as diagonally. A pawn that took a castle could move back and forth any number of squares, but still not move side to side. Creating new combinations of pieces gives the game variety that a regular chess game doesn't have.

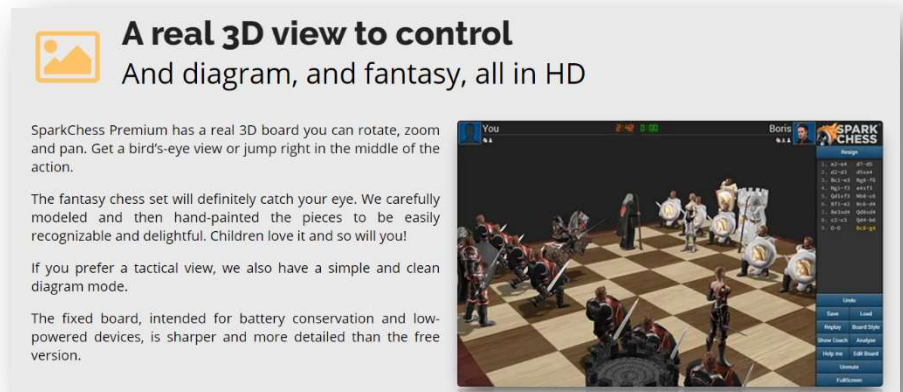


Image Source: <https://www.sparkchess.com/download.html>

VIRTUAL REALITY SPACESHIP

VR game where player pilots a spaceship. The player sits at a console inside the ship and manoeuvres with a series of buttons and a joystick (using VR controllers, not mouse and keyboard). The game would be focused on the complex controls, this will be the main challenge for the player. The controls would be highly interactive, each component requiring different actions to make full use of VR. The player would sit in an "open" cockpit, where they would be able to see all around the ship through glass. Gameplay would include fighting other AI driven spaceships in waves, the player would have to survive as long as possible.



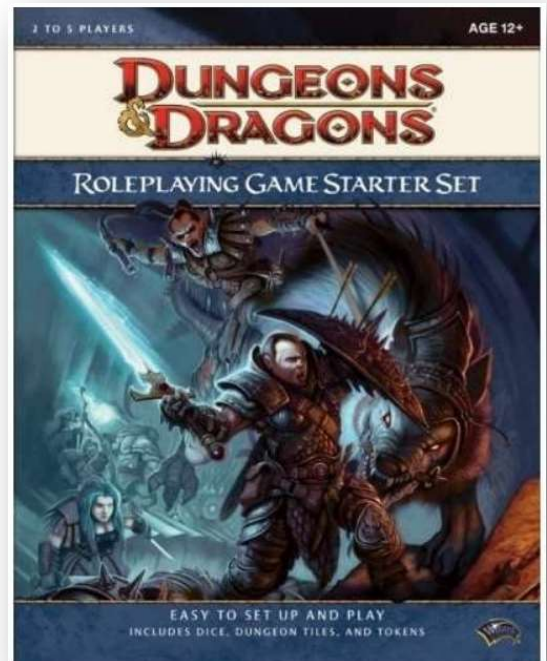
Image Source: https://store.steampowered.com/app/982700/Space_Battle_VR/

FANTASY ROLE PLAYING GAME (RPG)

A fantasy RPG type game where the player leads a team of AI controlled characters. It would be a lot like playing as dungeon master in the RP game Dungeons and Dragons, you control certain events around them while following certain rules and limitations, and lead them towards certain goals, but can't directly affect their decisions.

Image Source:

https://www.dmsguild.com/product/110213/DD-RPG-Starter-Set-Quickstart-4e?filters=0_0_44703



IDEA JUSTIFICATION

The VR spaceship game is the best option for my major project because it's an achievable idea that will be fun and interesting to play.

Because the game will be played in space, I will be able to avoid the modelling and designing of the ground and scenery. Modelling and design are not my strong suits so this will allow me to focus on the gameplay. The VR aspect provides a level of interactivity most games don't offer, this will allow me to create an entertaining game in a shorter amount of time.

COMPARISON OF EXISTING PRODUCTS

There are a lot of VR games already in existence that are able to take advantage of the possibilities offered by VR. A good example of this is the incredibly popular VR rhythm game, “Beat Saber”.

BEAT SABER

Beat Saber is a brilliant game that has the player physically swinging light sabers at floating blocks to the beat of the music, whilst dodging walls and other obstacles flying at them. Most rhythm games aren’t VR based and the player simply has to press buttons or click on certain areas to the beat of the music.

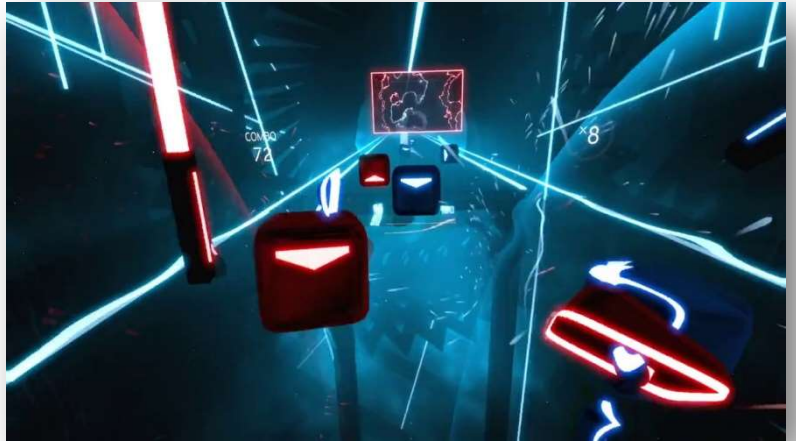


Image Source: <https://i.ytimg.com/vi/vL39Sg2AqWg/maxresdefault.jpg>

GUITAR HERO

Before VR was commercially available, “Guitar Hero” was had immense success largely due to the fact that it could be played with an actual guitar. The game itself involved holding the right colour buttons on the guitar and strumming at the right times.

Both of these games were able to fully utilise the technology they had available to them. By not fully taking advantage of the VR technology, the VR would become an annoying gimmick that doesn’t contribute to the player’s experience.

The spaceship is a perfect environment to use VR, the player can interact with everything inside the spaceship, without have to of their real-life chair. All the while their character can fly around in-game in their ship, so they won’t feel limited at all.



Image Source: https://s3.amazonaws.com/production-media.gameinformer.com/styles/thumbnail/s3/legacy-images/Reunion%20Tour%3A%20The%20Best%20And%20Worst%20Of%20Guitar%20Hero/Gh3_2D00_337_2D00_610.jpg

FASTER THAN LIGHT

“FTL: Faster Than Light” is a popular game released in 2012 that was also set in space, it took the world by storm because of its innovative gameplay mechanics. FTL was a roguelike game, meaning the game’s levels and challenges were procedurally generated, and it used this technique to keep the game fresh.

Once a player died, they would have to begin the game again on a new, randomly generated set of levels, but this time they would have unlocked certain upgrades that would allow them to go further in their next run. The actual gameplay was a blend of strategic spaceship combat, and text-adventure style decision making.



Image Source: https://steamcdn-a.akamaihd.net/steam/apps/212680/ss_052d698926073e8d407a864f0e63a486af24ec0d.1920x1080.jpg?t=1589331228

Player's would navigate the universe in their ship and make decisions on random encounters and events, such as passing by merchants or being attacked by pirates. If certain decisions were made, they would be thrown into a combat where they would order crew members to man guns, repair machinery, or fight aliens. The ingenious game mechanics in FTL made the game a hit, despite the somewhat simple graphics and art style.

USE OF INFORMATION GATHERED FROM EXISTING PRODUCTS

My game will be extremely low budget and will be made with limited resources, which is why I want to take a similar approach to my development. The core of the game will be in its mechanics and gameplay, not in flashy visuals or fancy models.

After planning how the game works, and all its features and functions, I need to decide how it will look. The visuals are an important part of video games, and although they aren't the focus of mine, they're still necessary so the player can see what's happening.

There are two general themes I think will suit the game, realistic or arcade style. I could either give the ships a semi-realistic look, or create an immersive and suspenseful atmosphere, or I could make the ships colourful and fun. The two examples I'll use for comparison are Elite Dangerous, and No

IT Multimedia – Spaceships!

Man's Sky. Both are space simulation games, but Elite Dangerous has a darker, more realistic art style, and No Man's Sky is almost cartoonish with bright colours and imaginative landscapes.

From the title alone you know **Elite Dangerous** is a serious game. You play as the captain of a spaceship, and are given the ability to travel the universe and do as you please. The only downside is that everyone else can do the same.

As you explore different star systems, you might try to mine asteroids for precious metals, or carry cargo between planets for money, but around every corner there are enemies. Sometimes you come across pirates who want to destroy and loot your ship, sometimes you attract the attention of the authorities and are forced to flee or die, you might even stray too far from home and encounter the alien species known as the Thargoids.

There's always danger in Elite Dangerous, and this is reflected in their art style. The lighting is dark and moody, the atmosphere is tense, and the brightest colours are missiles, lasers, and explosions.



Image Source: <https://store-images.s-microsoft.com/image/apps.10600.71402406373421469.b1c2a6be-9ffb-4b2b-a6f5-d74bc63cea2b.f13510ae-a5cf-497b-a78b-5096354d5c73?mode=scale&q=90&h=1080&w=1920&format=jpg>

The realistic setting makes it that much more impactful when you see a star collapse, or you bring down an enemy ship. The link to reality helps immerse you in the game, allowing you to forget that there aren't really any space pirates hunting you down.

IT Multimedia – Spaceships!

No Man's Sky is similar in that it's about space exploration but is focused less on fighting for your life and more on exploring life in space.

Much like *Elite Dangerous*, *No Man's Sky* features an entire universe of randomly generated planets, but each is far more unique. On each planet there's a completely different eco system, with completely different flora and fauna, and you're free to travel between all

of them, although there's 18,446,744,073,709,551,616 planets, so it might take a while.



Image Source: <https://pxlbbq.com/fifa-19-devoile-tous-les-titres-de-sa-bande-son/>

The game is designed to give the player a sense of wonder, and their art style is a huge part of this. The wild colour scheme highlights how alien everything is, and it makes discovering a new planet for more interesting.

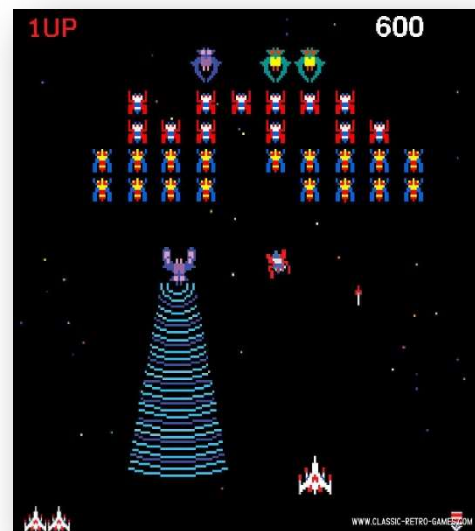
PLAYING STYLES

The two games have different playstyles, and their art styles highlight those differences. The art style of my game should utilise its playstyle, and contribute to the player's experience.

While my game will not include any exploration, and is combat-focused, it also doesn't feature the open-world or realism features that Elite Dangerous has. My game lacks certain features that are necessary for the game to be a serious space simulator, such as high-quality graphics and audio, and realistic physics. These features require a larger development team or amount of time to be completed, and would be more demanding on the computers that ran them.

My game instead will have arcade-style features such as a point system and waves of enemies. An art style reminiscent of old school arcade games such as Galaxian, but with a modern remastering, would make the gameplay clear to the players, and attract fans of old school arcade games.

Galaxian is a classic arcade game that involved piloting a spaceship and fighting aliens to achieve a high score. The different aliens had different shapes and abilities, but were most easily distinguished by their colour. Anyone who played for a wave or two quickly learned that the blue and yellow ones were the cannon fodder, the purple ones could abduct your ship with a beam, and the red ones escorted the green ones. This kind of colour coding could be used in my game as well, to help the player differentiate the different enemies.



PROJECT PROPOSAL

FOR MY MULTIMEDIA MAJOR PROJECT, I PLAN ON CREATING A VR BASED VIDEO GAME WHERE THE PLAYER ACTS AS A SPACE PIRATE PILOTING A SPACESHIP.

The game will have players sitting in the cockpit guiding the ship with a variety of complicated controls as an extra layer of interactivity and challenge. Players will complete missions to buy upgrades for their ship, allowing them to take on more challenging and rewarding missions. Missions will consist of tasks such as navigating through asteroid fields and battling other spaceships.

CHALLENGES I'M ANTICIPATING ARE:

- Modelling the spaceships and asteroids (I may have to buy/download the ship model/s)
- Modelling the ship's console
- Making the controls work in VR
- Designing a GUI
- Making/finding game soundtrack
- Scripting the enemy AI
- Scripting asteroid fields
- Testing the game with a VR headset and controllers

REVISED IDEA

AFTER IT WAS MADE CLEAR THAT VR WAS NOT A VIABLE OPTION FOR THIS PROJECT, THE PLAN HAD TO BE REVISED TO EXCLUDE VR

(more info at Issues Encountered, Page 20) Because VR is still relatively new to game development, and has not been officially supported by Unity yet, there is a lack of documentation that made it difficult to learn to develop a VR game. This difficulty was exacerbated by the knowledge that existing sources of information assumed. The Virtual Reality Tool Kit, the plugin used to develop VR in Unity, requires users to have a strong understanding of Unity and previous experience in game development. To gain this knowledge would consume a lot of my time which could otherwise be spent progressing in the development of my game, so by simply cutting out the VR section of my project I was able to save a lot of time making the realisation of my project a reality.

Removing VR from my plans would mean I had to rework the controls system, and modify the game so it would still be entertaining. The main purpose of the VR was to make the game as entertaining with the most efficiency possible. Because of my previous plans to include complex AI and randomly generated levels, I can still develop an entertaining and complete game within the time frame.

SCHEDULE FOR DEVELOPMENT

SEQUENCING

As mentioned on numerous occasions already throughout this Portfolio I am extremely conscious of the need to manage my project carefully in order to meet the deadline required.

To not only meet this deadline but to create a project of quality within that time will require careful planning and management.

1. Process Diary/Portfolio – The first thing that needs to be done is begin my portfolio, or more specifically my process diary. This way I can document everything that happens as it happens. I can also document the decisions I make such as choosing a game engine.
2. Learn to develop in Unity – After deciding to develop in Unity, I need to research Unity and how to develop in it.
3. Research C# - Unity runs in C# so I need to learn it for scripting.
4. Research HSC guidelines and marking – I need to know what can and can't be included in my major project, and what it will need to end up like to receive the best possible marks.
5. Generate game ideas – After I know what can and can't be done, I can begin to come up with ideas for my game.
6. Create game proposal – Once I have an idea of what I want my game to be, I need to make a proposal so I can get a teacher's permission.
7. Scheduling/Gantt chart – Now I have to decide what has to be done, when it has to be done, and how long each task will take.
8. VR related – There were several VR related aspects that I originally scheduled for that will no longer be needed.
 - a. Setting up VR at desktop – Before I can begin developing or learning to develop in VR, I need to set up my VR headset and controllers
 - b. Setting up VRTK – I also need to download and install VR Tool Kit, the extension I've chosen to use that will assist me in VR development
 - c. Learn to develop for VR – I need to learn how to develop for VR
9. Ship Movement scripting – I need to establish a basic prototype for the ship's movement before anything else
10. Camera Movement scripting – I need to make the camera follow and look at the ship without shaking
11. Creating HUD – I need to create a simple version of the Heads-Up-Display, which is a screen that displays certain information for the player, while I'm still developing the game, then I'll make the final product once the game is closer to completion and things are less likely to be drastically changed.
12. Weapons and projectile scripting – I need to script weapons for the ship and the projectiles it fires
13. Enemy AI Scripting – I need to create Ai that are capable of fighting the player
14. Creating Skyboxes (planets, stars, sun) – I need to create skyboxes that will contain the game's scenery.

15. Modelling ships – I'd like to create the models for the ships myself, but if I don't have time I may have to buy or source them.
16. Creating Textures – I need to create textures for the models, HUD, and the GUI
17. Develop game GUI – I need to create a GUI that will allow the user to change settings and to navigate game scenes
18. Set up endless mode (waves) – Endless will be the main game mode, and once I have all the assets, I can set up this game mode
19. Finding/buying SFX and music – I'll need to find/buy SFX and music
20. Scripting SFX and Music – I'll need to script it to play at specified times
21. Set up scoring/progression system (stretch goal) – This is something I'd like to do if I have extra time
22. Port to Mobile (stretch goal) - This is something I'd like to do if I have extra time
23. Project Evaluation – I'll need to evaluate the game, and see what went well and what didn't.

PROJECT MANAGEMENT

ISSUES ENCOUNTERED

The first issue encountered was actually setting up a VR headset and controllers, so that I could begin development.

My family already owned a VR set, so I began to set it up at my desktop computer. I discovered that the controllers required Bluetooth, which my desktop didn't have. I borrowed a Bluetooth 2.0 adapter from my dad, but it needed Bluetooth 4.0. I went to the local Jaycar and JB-Hifi, and found they didn't have a Bluetooth 4.0 adapter either. I ordered a Bluetooth 4.0 adapter online, from <https://www.amazon.com/Plugable-Bluetooth-Adapter-Compatible-Raspberry/dp/B009ZHLLI>

I expected the adapter to arrive in about two weeks, but due to the bush fires blocking off roads it arrived after nearly four weeks.

Once it did arrive, I was able to set it up and begin working in VR immediately. This brought on my next issue; because VR is a relatively new technology, it hasn't been officially integrated into Unity yet.

Originally, I planned on using a Unity extension called VR Tool Kit, which contained a collection of assets that would allow me to develop in VR. The toolkit worked fine but it assumed a lot of knowledge that I didn't have. I had already completed their beginner's guide to VRTK, so I had to go elsewhere to learn about VRTK.

They had a website which documented all their functions and assets, however I was unable to understand most of this information as well. I searched the internet for tutorials and information, but because VRTK is still in beta and not widely used, there wasn't much information out there yet.

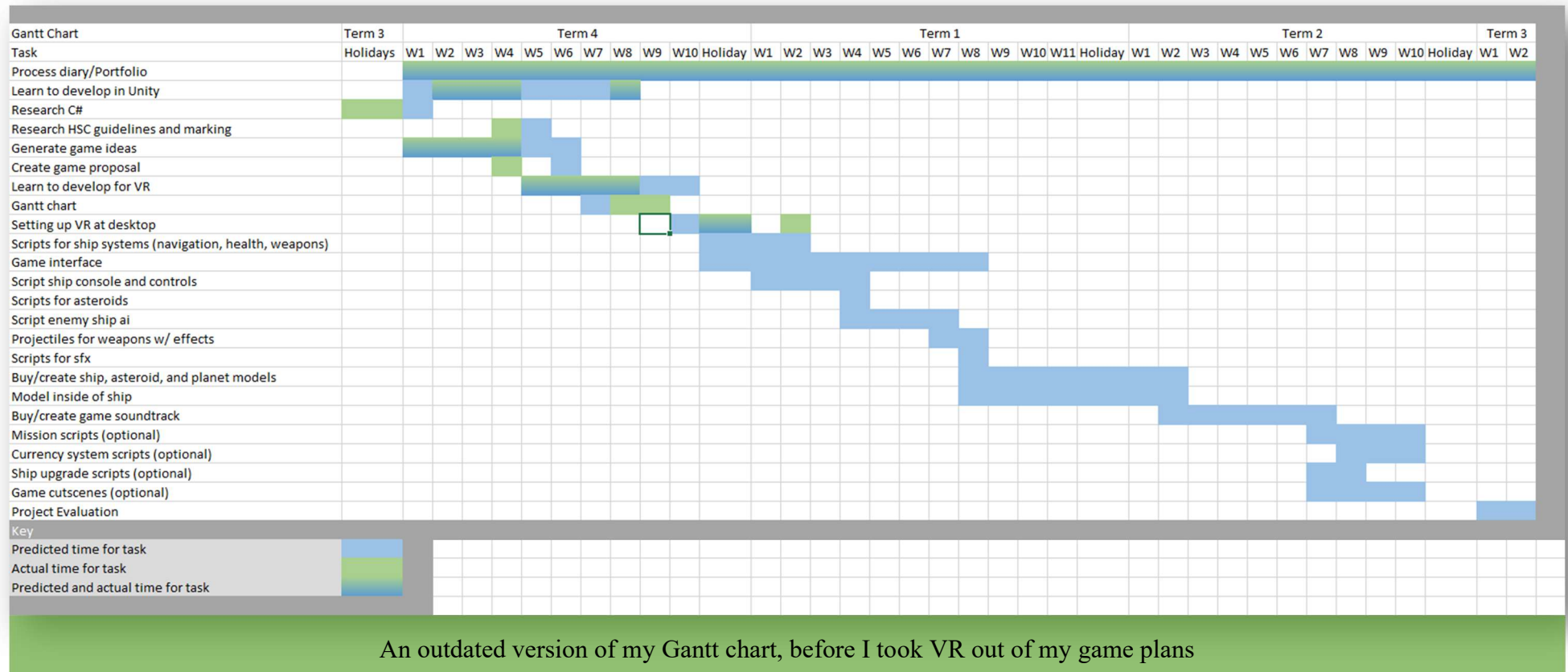


IT Multimedia – Spaceships!

It became clear that if I were to continue to attempt to develop the game in VR it would take far longer than the given time frame, as I'd have to learn about VR and Unity for months before I could begin.

I decided that VR was not a viable option, so I replanned my project so that it would be the same game, without VR.

My Gantt chart before I removed VR:



Another issue I encountered was scheduling.

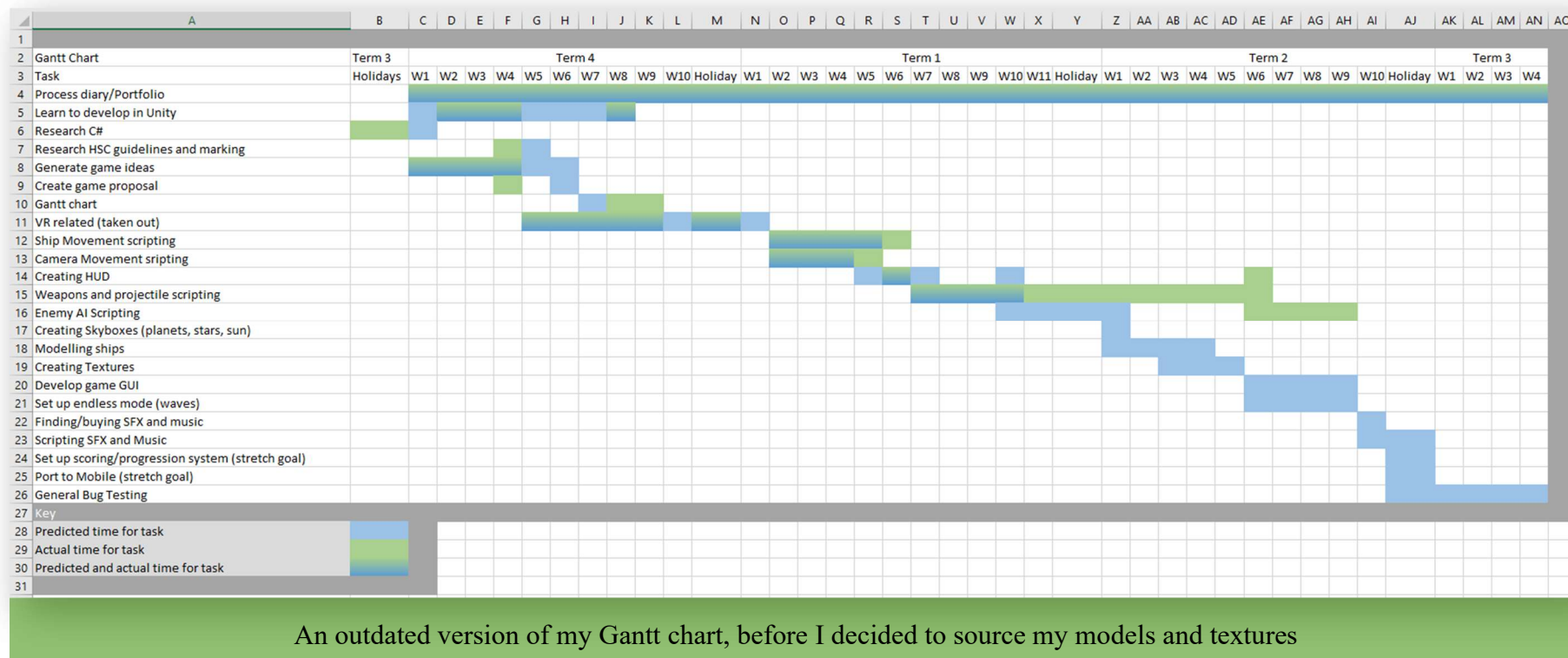
I planned to have the weapons system complete in Week 10 of term 1, but it took an additional 9 weeks, due to COVID-19 isolation.

IT Multimedia – Spaceships!

Once our school closed down shortly before the end of term 1, I began attending my classes online. There were three main reasons my progress was slowed so severely during isolation, poor time management, lack of motivation, and limited access to my computer.

Before isolation I was accustomed to the structure that school provided, each class was separated by a bell, and I had no choice but to work on each subject equally. During isolation I found myself working more on some subjects than others, and I put all my multimedia class time into the in-class industry work, rather than the project which usually required multiple uninterrupted hours to make any progress. I also had to assist my family with various issues, and share my computer with my siblings, which greatly limited the amount of time I could spend working on my project. As a result, I was put 9 weeks behind what I originally intended.

My Gantt Chart before COVID-19 lockdown was released:



To make up for this, I decided to rearrange my Gantt chart.

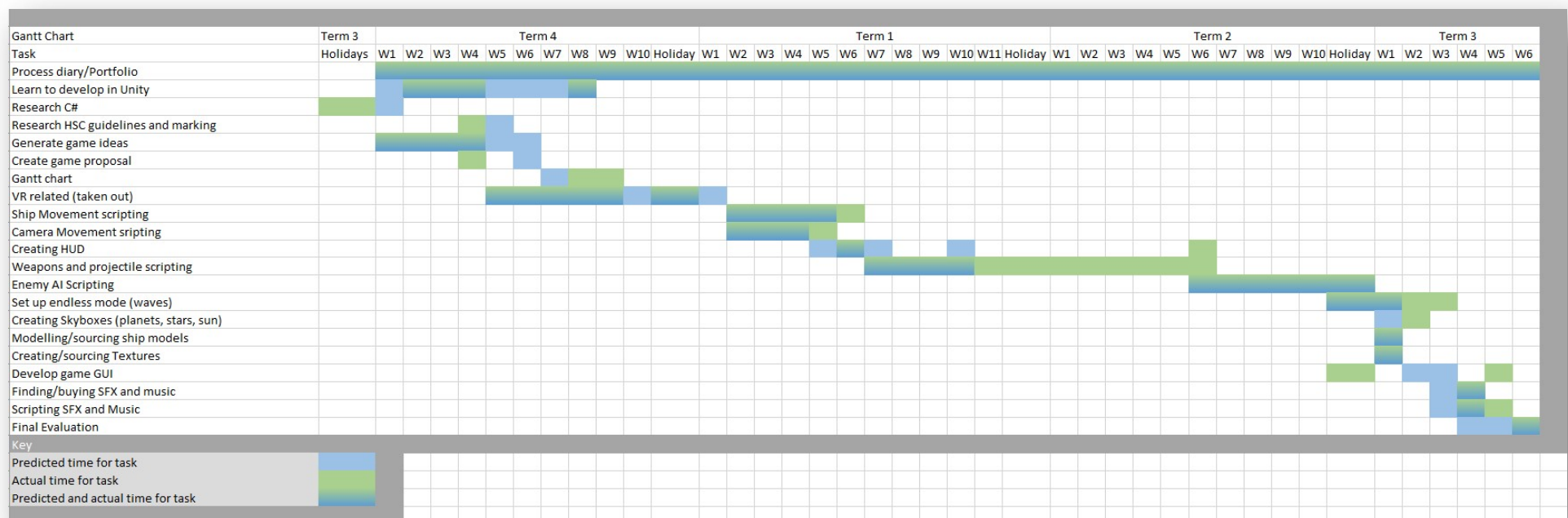
Originally, I wanted to model my ships and design my own textures, but due to the time limitations I decided to source them instead, which would take far less time to complete.

This made up for lost time due to Covid-19 and allowed time for the more complex areas in the game that I couldn't buy or download, such as the AI.

GANTT CHART

I originally planned on creating a VR game, however during Term 1 Week 2 the game was redesigned to not include VR. The first Gantt chart includes the VR development and research components, and the second includes the newer game plan that doesn't include VR. This is the latest version of my Gantt chart, after I rearranged it to make up for lost time during COVID-19 lockdown.

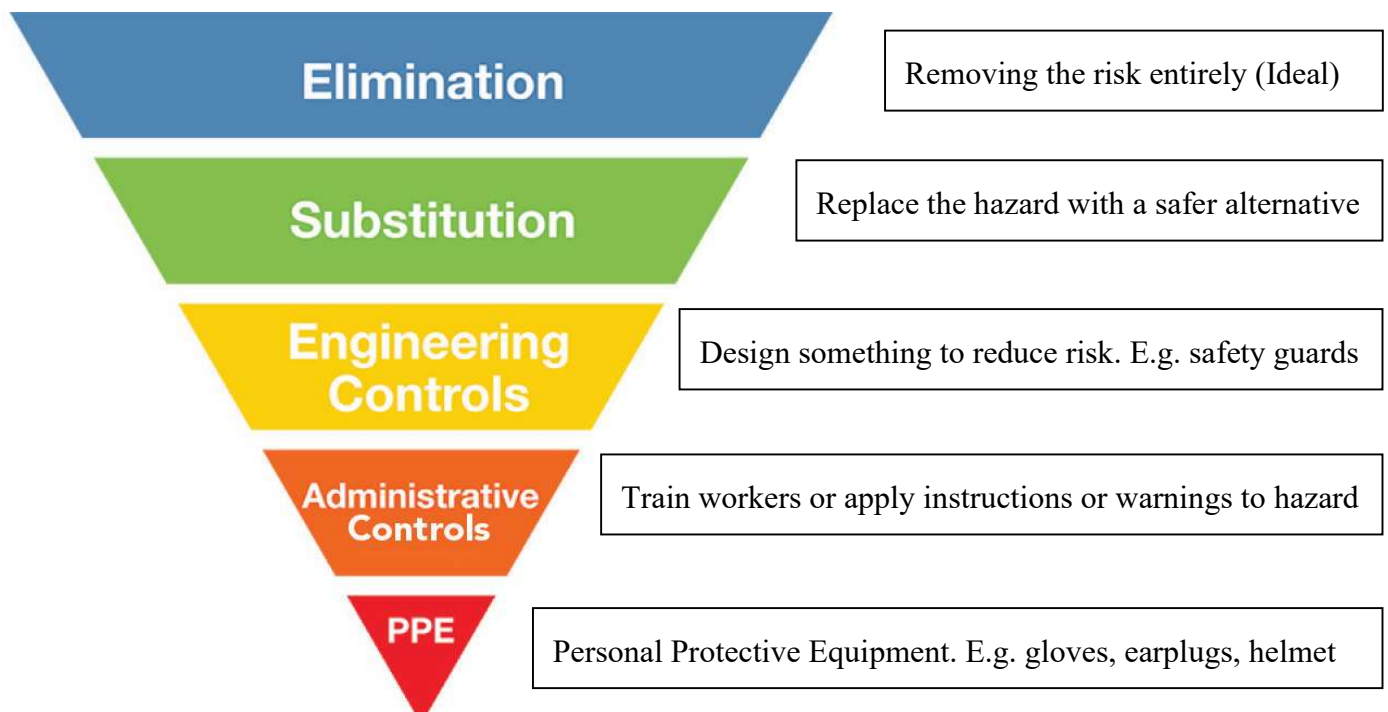
After COVID-19 rearranging:



RISK ASSESSMENT

There are hazards in any workplace, both large and small. I need to identify the hazard in my workspace so that I can develop my game safely. Then I need to assess the risk by using a likelihood vs consequence matrix. I need to take appropriate actions to minimise or eliminate the hazard using the hierarchy of controls, based on the level of risk that I assess.

Likelihood	Consequences			
	Insignificant injury or damage	Minor injury or damage	moderate injury or damage	major injury or damage
Very Unlikely	Low Risk	Low risk	Moderate Risk	High Risk
Unlikely	Low Risk	Moderate Risk	High Risk	Extreme Risk
Likely	Moderate Risk	Moderate Risk	High Risk	Extreme Risk
Very Likely	High Risk	High Risk	High Risk	Extreme Risk



Source: https://www.safetyandhealthmagazine.com/ext/resources/images/2018/04-apr/HierarchyOfControls_Alt.png?1522181445

Hazard Identification	Risk Assessment	Risk Reduction
Muscle Strain Prolonged use of a computer can cause muscle strain if proper ergonomics are not employed. An arched back or neck can put cause excessive stress on your muscles due to, and cause back or neck pains.	Moderate Risk	To prevent muscle strain, I need to use administrative control, and sit upright in my chair with my back against the back support, sit with my thighs parallel to the ground and knees at a 90° angle, with feet flat on the floor, and type with elbows at a 90° angle. This will prevent excessive weight being placed on my back, legs, and arms. I'll also adjust my monitor so the top is at eye level, to

		prevent me from arching my neck (and to minimise eye strain)
Repetitive Strain Injury RSI can occur in muscles if you move them quickly for extended periods of time. This is most common in the hands and fingers, as keyboards require you to tap keys quickly and lightly. Symptoms include cramping, pulsing pain, aches, tingling, and extremity weakness	High Risk	RSI is caused by repetitive use for extended periods of time, so I can eliminate the risk by taking frequent breaks and reducing the amount of time I spend typing consecutively. I can also use engineering control to reorganise my workspace, for proper typing technique. I'll move my keyboard to be directly over my thighs, with the mouse to one side, and the monitor directly in front of me.
Eye Strain Eye strain is a condition that's caused by extended visual tasks with poor lighting, such as glares, dim lighting, and contrasting shades. This forces the eye to strain to see properly, and create headaches, blurred vision, fatigue, and pain in and around the eyes.	Moderate	To prevent eye strain, I need to eliminate the hazard by ensuring consistent lighting on my monitor. I will close the window blinds to prevent glare on my monitor, and move my desk so overhead lighting doesn't cast shadows on my monitor from my head.
Slips and Trips Trips occur when a person unexpectedly catches their foot on an object or surface. Slipping occurs when a person's foot loses traction. Although they usually result in minor scrapes and bruises, they can cause fractures and dislocations.	Low Risk	Keeping a clutter free workplace is helpful for not only maximising productivity, but preventing trip hazards. This means using engineering control to zip-tie loose cables out of the way, and eliminate hazards by clearing the floor of trip or slip hazards.
Electrocution Computers require many cables and wires, and with improper treatment, this can cause a shock hazard. Exposed wire and faulty live equipment can send surges of electricity through your body and stop your heart.	High Risk	Electrocution is extremely uncommon due to the amount of risk reduction already in place in Australia. Power outlets have several safety mechanisms that will minimise harm, such as the grounding wire that offers a safe path for the electricity to be discharged through, instead of the person's body, and the circuit breakers that activate upon detection of stray voltages. In addition to the safety measures already in place, I'll use administrative control to look over the cables and wires to check for exposed metal once in a while.

FINANCE MANAGEMENT

Because of my access to free software and equipment I already own, I don't plan on spending any money on my project. Because of this I've listed all the free software and equipment I've used in a spreadsheet.

Item	Cost	Notes
Home Computer	\$ 1,500.00	Already Owned
Unity	Free	Game development software
Microsoft Office	\$166.99	Already Owned - various publishing software
Adobe Creative Suite	Free	Free for one year as NSW student - various editing and design software
Visual Studio	Free	Integrated development environment
NVIDIA Shadowplay	Free	Screen recording software

Estimated costs

As I've developed the project, I've come across a few unexpected costs, which I've tracked in a self-calculating spreadsheet.

Item	Date	Cost	Notes	Total
Bluetooth 4.0 USB Adapter	27/11/2019	\$ 19.51	To connect controllers to computer, as computer lacked BlueTooth 4.0	\$ 159.50
Multimedia Fees	3/02/2020	\$ 50.00	Industrial Technology - Multimedia has a class fee of \$50	
Seagate 2TB Expansion Hard Drive	10/04/2020	\$89.99	Bought a portable hard drive to bring major project to school	

Actual costs

PROTOTYPING, MODELLING AND TESTING

GAME FUNCTIONS

Development requires prototyping and testing to work out how the final product should work, and to find issues with the current design before it's too late to change easily. In game design specifically, this means a shell of the game must be made before it's given textures.

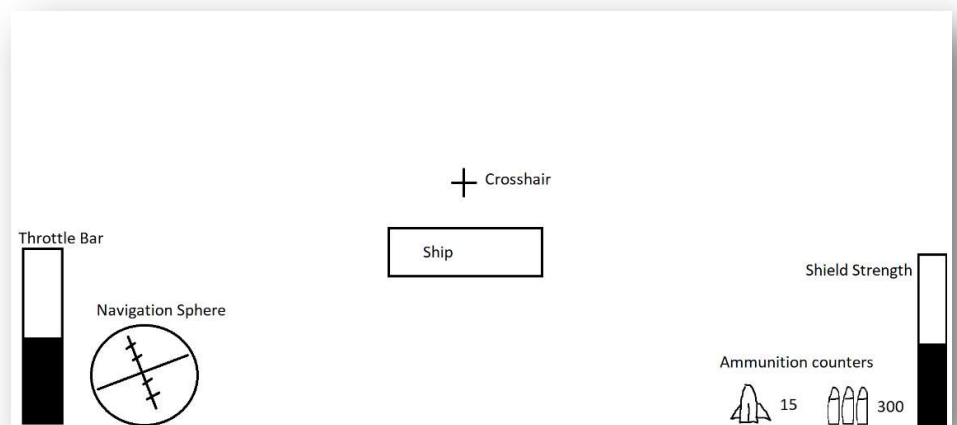
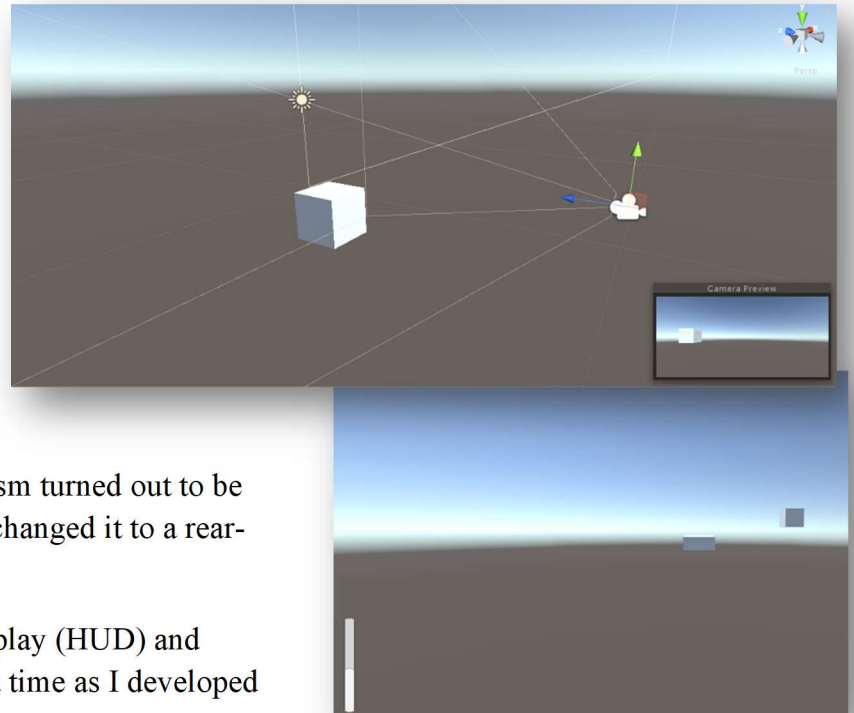
To begin my game, I made a simple version of the ship that could rotate and move forward, with a camera that followed it.

I prototyped a camera system that would allow the camera to orbit the ship so the player could see behind the ship. This camera mechanism turned out to be difficult and irritating to control, so I changed it to a rear-view camera while holding a button.

Then I planned out the Heads-Up-Display (HUD) and began implementing elements one at a time as I developed other game systems.

First, I implemented throttle control and added a throttle bar to the HUD, then I added the weapons. Initially I planned to have an ammunition counter, but I realised limiting the player's ammunition just slowed down gameplay, so I took them out. I also added a health bar that displayed your ship's shield strength.

I intended to add a navigation sphere that would allow the player to easily find surrounding objectives and points of interest but as the game was being developed, the missions and objectives I had originally planned were taken out of the game and the navigation sphere became redundant. Most games have a mini-map that allows the player to see their environment to get to different locations, but I found that as there is no environment in space, and there were no longer any objectives to travel to, there was no need for a mini-map.



TEST COMPUTERS

My home computer which I use to develop my game is above average in terms of hardware. I want to ensure that my game will run on the majority of modern computers. Not all computers are as powerful as mine, so I need to be frequently testing it on cheaper, and more common computers. I tested my game on 3 different computers throughout the development process:

SASHA

This is my home computer that I used to develop the game, which I named Sasha. I designed and built it in the holidays between 2018 and 2019, and it's the most powerful computer I tested my game with.

Hardware Specifications:

- CPU – AMD Ryzen 5 2600
- GPU – GeForce GTX 1060
- Memory – 16GB
- Monitor – 1920 x 1080 pixel, 60Hz



The inside of my desktop computer

HP ELITEDESK 800 G2

This is the school computer that will be used to display the game to the major project markers, so it's essential the game works effectively on this device. This is an average computer that could be found in offices and homes.

Computer Specifications:

- CPU – Intel i5-6500
- GPU – GeForce GT 730
- Memory – 8GB
- Monitor – 1920 x 1080 pixel. 60Hz



HP EliteDesk 800 G2

TOSHIBA SATELLITE S50T-B

This is a laptop owned by my family. It's around 6 years old and it's been heavily used. This is an example of a low-end computer.

Computer Specifications:

- CPU – Intel Core i7-4500U
- GPU – AMD Radeon R7 M260
- Memory – 8GB
- Monitor – 1366 x 768 pixel, 60Hz



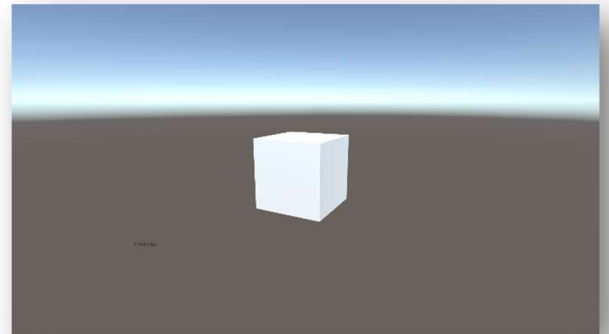
Toshiba Satellite S50-B

TEST BUILDS

I've been saving different “builds” of my game as I developed it which can be ran on different computers. Builds are compiled versions of programs that allow you to run the game without the use of any external software, such as Unity. I saved a new build every time I reached a significant milestone in my development, to see if that change prevented other computers from running the game. These are the different builds:

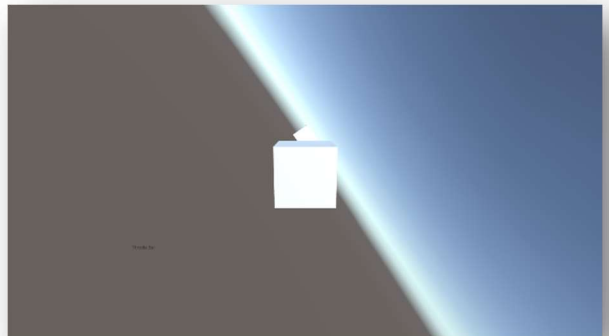
V1 – 13/02/2020

This is the first build I made; it contains a very barebones movement system with a box as a temporary model, and an indicator of where I wanted the throttle to be on the HUD. The player was able to move forwards and backwards, and rotate around only the y-axis (turn left and right). The camera was fixed to the back of the ship, so any rotations appeared very twitchy. **V1 ran without any performance issues on each of the three computers.**



V2 – 16/02/2020

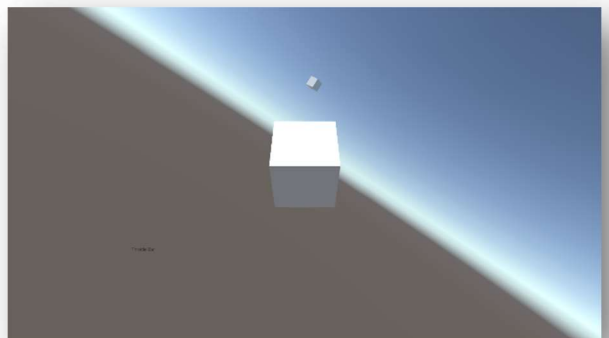
This version had a greatly improved movement system, and allowed the player to rotate on all the axes. The camera motion was also smoothed, and allowed for more fluid movements (although still a little stiff). This fluid motion would not work with rotations around the x-axis however (roll). I also added another box as a point of reference for movement while testing. **V2 ran without any performance issues on each of the three computers.**



V3 – 20/02/2020

This version fixed all the issues with the camera, the camera will smoothly rotate in any direction, and will “linearly interpolate” to the fixed position behind the ship. This means that when the ship moves, there’s a target location behind it that moves with it, and the camera will move towards it at a percentage of the total distance (so it might close half the gap every time). This allows for smooth movement and actually shows the speed of the ship through the distance from the camera.

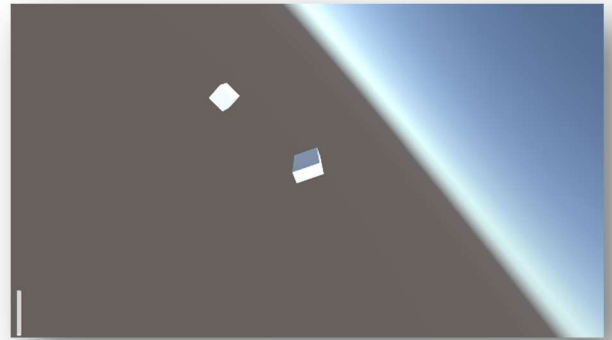
V3 ran without any performance issues on each of the three computers.



V4 – 01/03/2020

Version 4 modified the shape of the ship to be a little more suitable, and introduced movement control with the mouse and a throttle bar to the HUD. Mouse movement allowed users to more precisely move the ship by moving the mouse, and the throttle bar displayed how fast the player was going.

V4 ran without any performance issues on each of the three computers.



V5 – 14/05/2020

Version 5 implemented a prototype weapons system. It allowed the player to fire lasers (which didn't do anything upon collision yet) and missiles (which exploded upon contact and created a temporary sphere to demonstrate an explosion). It also included a bar to the HUD which showed the player's shield strength. Although there weren't any enemy weapons, the shield could receive damage and regenerate over time. This version would also include a reticle which would guide the player in their weapon firing.

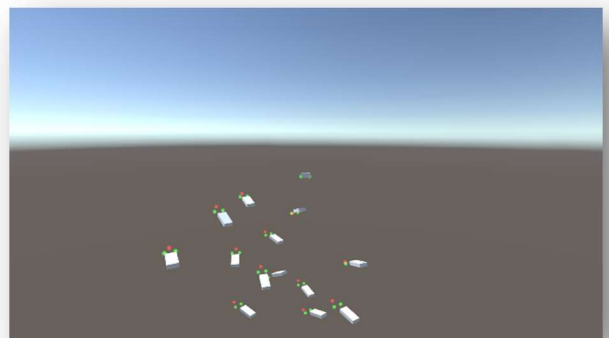
V5 ran without any performance issues on each of the three computers.



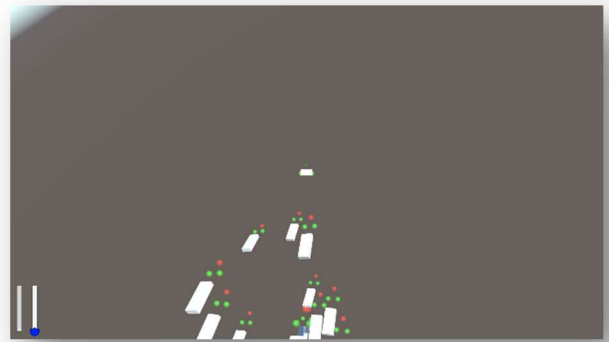
V6 – 28/06/2020

This version included a large amount of changes, as it took an extremely large amount of time and effort to add the AI in a functioning state. The complexity of designing and implementing the AI prevented me from creating a functioning build until the AI was closer to completion, so this version also includes several improvements to the weapons system. I saved the build with an extra two different scenes to demonstrate the different features, one scene that displayed the AI, and one that displayed the combat. There was the old scene from the previous builds which I used for testing, but it didn't display any new features

The AI scene contained a swarm of enemy ships that flew around in unison, demonstrating their ability to avoid each other and travel in a group without straying. I explain how I did this in the AI section on page 35.



The combat scene contained a similar swarm of ships chasing down the player, this demonstrated their ability to pursue the player, and fire upon them. The player was also able to test their improved weapons, the missiles will now home in on enemy ships if you point your launcher towards them for long enough to establish a targeting lock. Lasers and missiles also deal damage now, and will destroy enemy ships (or the player ship) with sufficient damage.

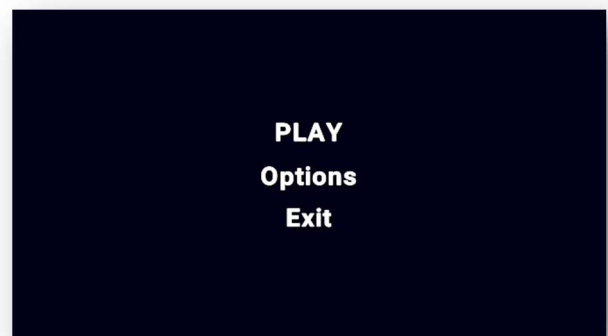


V6 ran without any performance issues on each of the three computers.

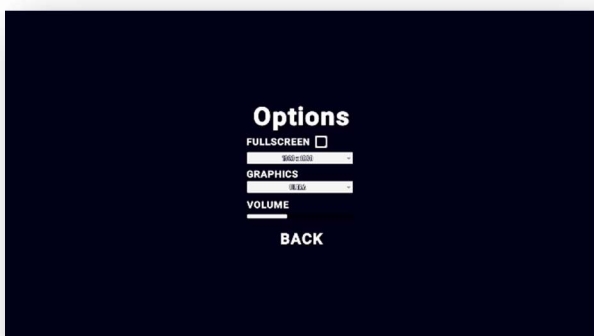
V7 – 12/07/2020

In version 7 I developed the first version of my User Interface (UI). It included a functional (although visually basic) main menu, options menu, and pause menu. It was difficult to script the menu functions, as I was completely unfamiliar with Unity's UI system prior to this, so some buttons don't work properly.

The back button in the options menu does nothing, and if the user changes the game's settings outside of my game, the options menu doesn't reflect those changes. There was also issues with the pause menu, as enemies continued to move around while the player was frozen, as a result of my lack of understanding of Unity's built-in time management system when I designed the AI.

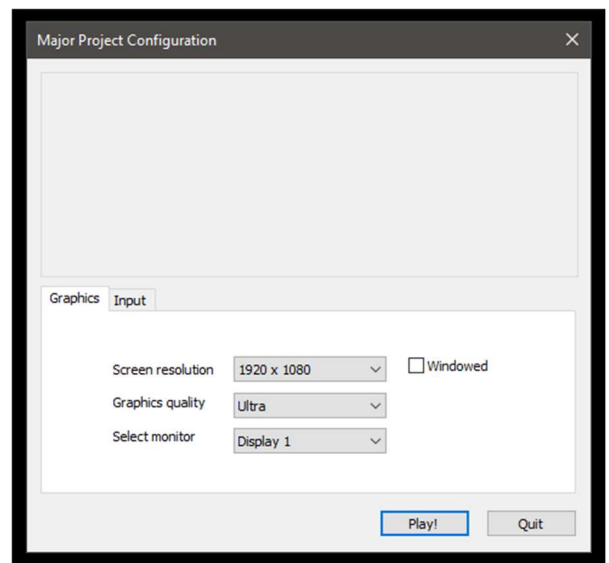


V7 ran without any performance issues on each of the three computers.



V8 – 20/07/2020

In this version I fixed several issues with the UI as well as adding another menu that appears when the player dies, and prompts them to play again or open the main menu. I fixed the pause menu by modifying the enemies code so they'd freeze in place while the game was paused. I also fixed the back button in the options menu, and the settings issues. Unity has a built-in settings menu that opens when you first boot up the game, I created a script that plays when the game starts that matches the game's settings to what's on that menu. I also added saving functionality, so the player's settings and high score can be saved. Now on boot-up the default settings will be whatever was used last time, and the player's high score is displayed on the death screen.

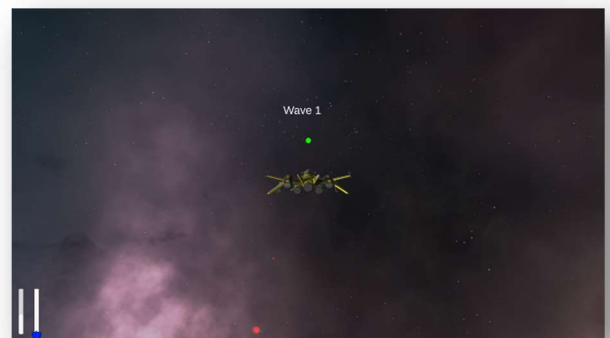


V8 ran without any performance issues on each of the three computers.

V9 – 11/08/2020

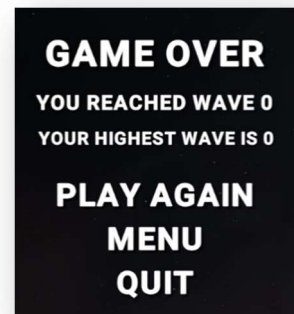


This was another big update, I added models and textures to most of the assets in the game. I created four separate types of enemy ships, and applied a unique model and texture to each of them, plus a model and texture for the player ship, and a space skybox. Upon adding the models I noticed they seemed to move very slowly compared to their size, so I gave them all a much higher movement speed. This worked great for the player's ship but giving the enemies a higher move speed drastically changed the AI, and I would need to rebalance all of the movement statistics to suite the speed. Instead of doing this, I reverted the speed and scaled down the size of all the ships. This way they appear to be moving much faster, and the AI functions normally.



I also finished the wave system, the first 5 waves will spawn a pre-set squad of ships in a random location around the player, and every wave after that has randomly generated squads of increasing difficulty. I also added a wave counter to the HUD

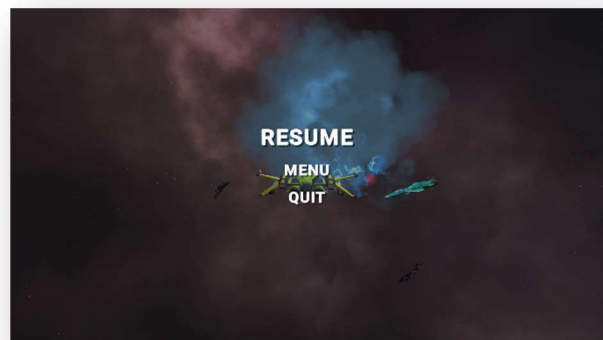
V9 ran without any performance issues on each of the three computers.



V10 – 22/08/20

In version 10 I added all the remaining models, alongside a pause menu and some game balancing changes. First, I searched through the unity asset store and found some explosion prefabs which I for the missile, lasers, and ships. Then I added created a pause menu that allowed you to resume, got to menu, or quit. This also included a pausing script and a series of changes throughout various other scripts that would pause each different game system while the pause menu is open. Finally, I

changed the enemy statistics to suit their type, so the light ship has less health and was faster, the heavy was slower and had more health, and the boss was made far stronger than the others.



V10 ran without any performance issues on each of the three computers.

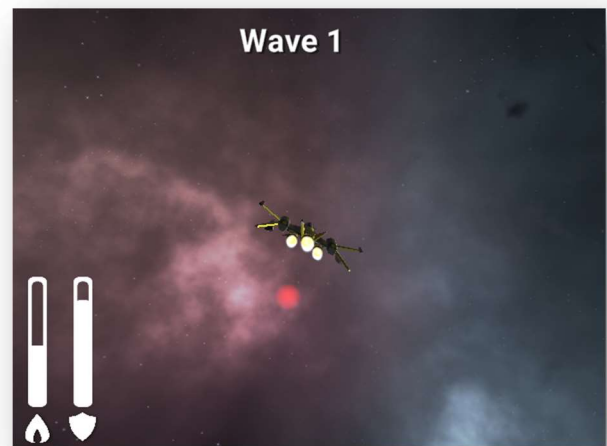


V11 – 25/08/2020

This is the final version of the game; in it I add sound effects, redesign the HUD, and the title screen. First, I browsed freesound.org and found a collection of suitable sounds (this was difficult as anything high quality costs money) and scripted them to play at suitable times in the game. This included things such as explosions, firing of weapons, and engine sounds. The engine specifically is scripted to increase in volume as the throttle is increased.

I also used Adobe illustrator to design some new icons and for the HUD. I created a shield for the shield slider, a flame for the throttle slider, and a hollow bar and a filler for the sliders.

Previously the main menu and options screen had a navy-blue background, which I thought looked very bland. It also didn't have mention the name of the game, so I added a text box that read "Spaceships!" I removed the blue background and added the space skybox, so the text was floating over the space background. Then I wrote a script that rotated the camera in a random direction so the space background moved around the screen.



V11 ran without any performance issues on each of the three computers.

ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is a series of algorithms designed to complete a task or tasks; most games, including mine, have one in some form. The AI that I need will be used to pilot the enemy ships, this will be more complex and difficult than most other tasks in the project. There are two main methods of developing AI, state machines and neural networks. I have also considered a type of algorithm known as a boid, which is a unique type of artificial intelligence that is only useful in scenarios such as this.

POSSIBLE TYPES OF AI

A neural network is where a series of inputs, outputs, and a goal is given to an algorithm. The algorithm connects the inputs to a randomly generated layer of modifiers, and connects those modifiers to the outputs. This process is repeated over and over to create a large “generation” of new algorithms, these algorithms are then simulated and measured by how effectively they achieve the given goal. The best algorithms are copied and modified, to be put into a new generation. The process repeats until the algorithms are capable of completing the goal efficiently enough.

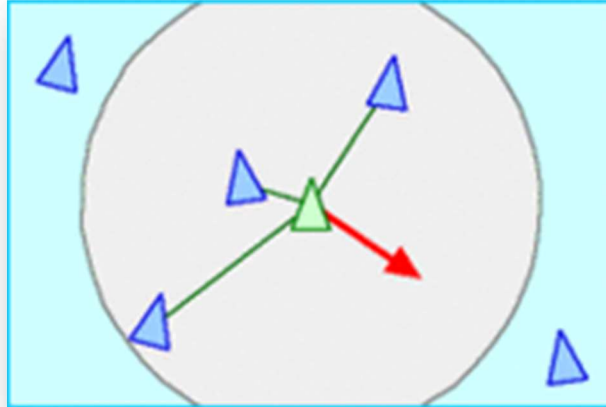
Neural networks are extremely effective at simple tasks, where they receive one stream of data, and produce one stream of data, but become less effective and more complicated to establish the more complicated their task is. A network could easily be designed to create sheet music out of a song, as it could distinguish notes and tones and write them down, but it couldn't easily be designed to create a song, as it would have to understand melodies, rhythms, song structure, etc.

A state machine is an algorithm that uses sensory inputs to decide which one state the AI will be in. An AI will have a list of states which each read the inputs and calculate different outputs. There're different purposes for each state, an AI in Mario Kart might have a state for driving forward and avoiding the edges of the track, a state for driving towards an item on the track, a state for using that item, and a state for returning to the track after being knocked off.

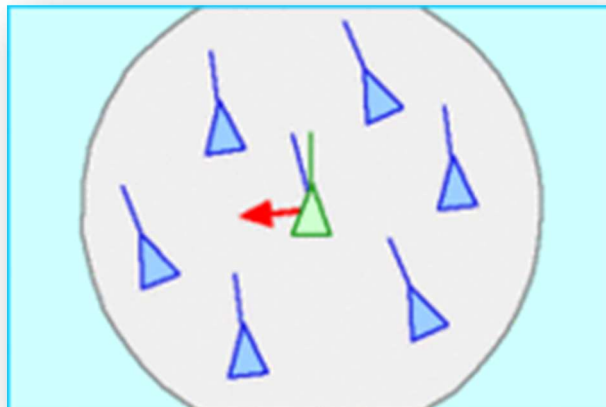
State machines are more versatile than neural networks, but they require more work to set up, and only work effectively in predictable situations. The more situations the state machine needs to work in, the more states that have to be programmed, and the more complex each state becomes. A state machine can be designed for almost any situation, but it can be vastly complicated, and requires lots of time, knowledge, and skill to develop.

A boid is a “bird-like-object”, boids can be used to simulate birds flocking, or in this case, a swarm of alien spaceships. Boids have three simple rules that allow them to form a collective intelligence when nearby other boids:

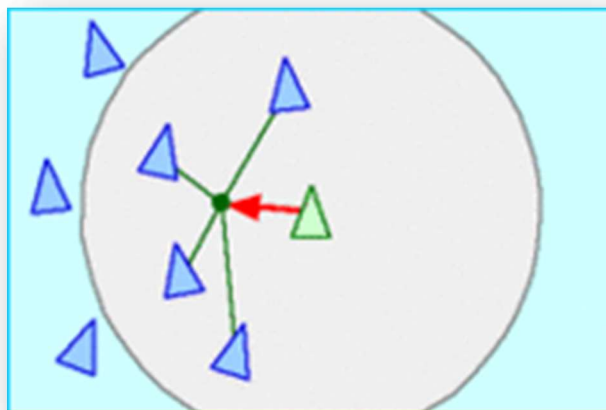
- Separation, Boids must move away from nearby boids



- Alignment, boids must orient themselves with nearby boids



- Cohesion, boids must move towards the centre of the flock

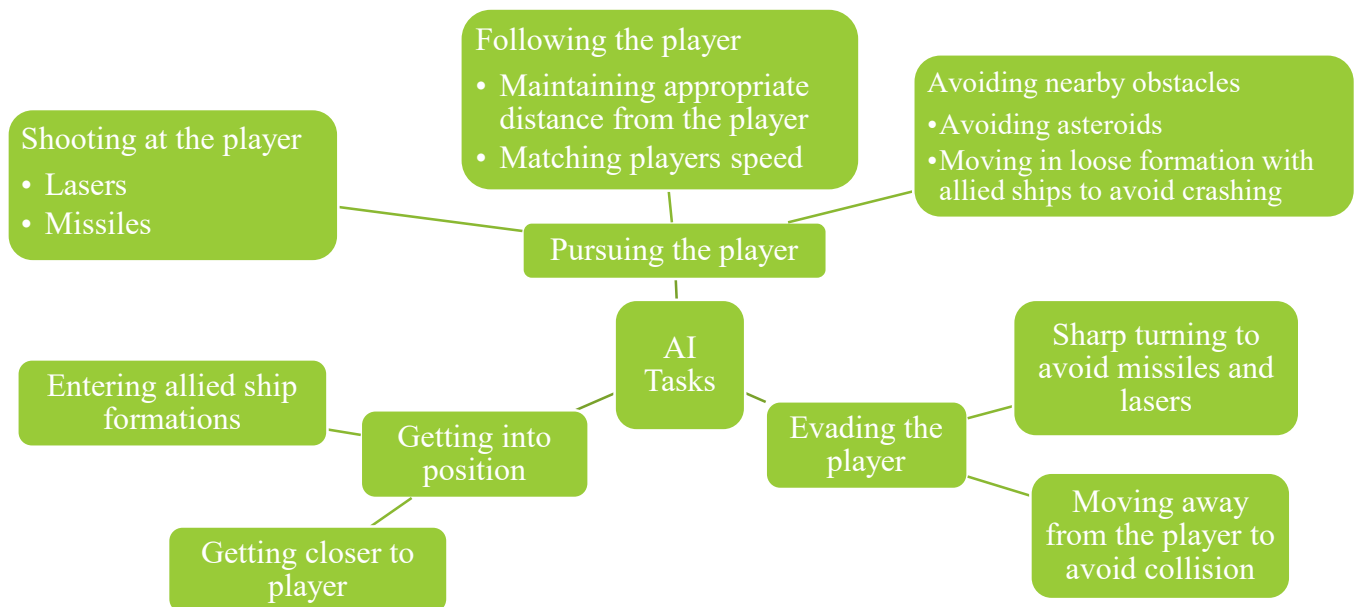


These three rules each produce a vector, the importance of the vector is weighted (allowing you to decide how important each rule is), and then a final vector is calculated that incorporates each of the three rules. Boid programs are mainly used for coordinated movements between multiple objects, to simulate a group intelligence, which could potentially fit nicely for the enemy ships in my game.

Boids are efficient and effective algorithms, they do the job well and they're easy to establish, but they are very situational. You can only use boids for providing vectors based on context, and aren't suitable for situations that don't require vectors or are unable to provide the necessary contextual information (what's surrounding the boid, how far is it, what is it...)

CONSIDERATIONS

To decide on an artificial intelligence type, and begin planning and developing my AI, I'll need to figure out what my AI will need to accomplish. This is a brainstorm of all the tasks the AI needs to accomplish

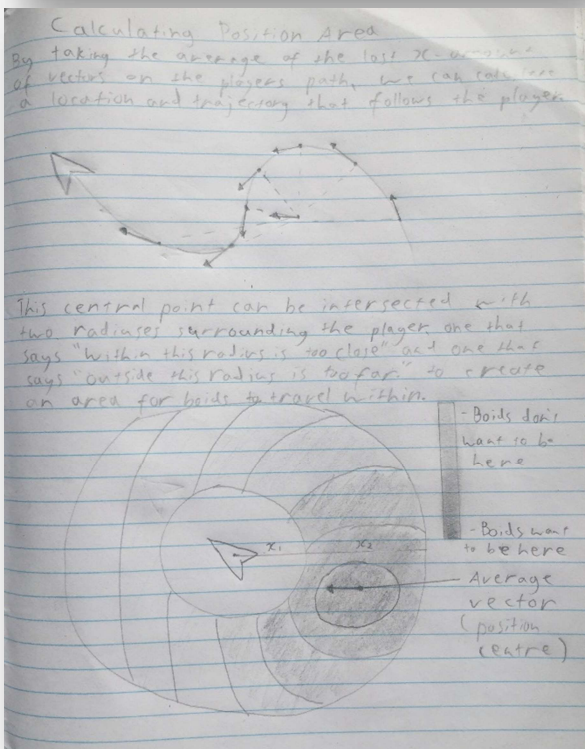
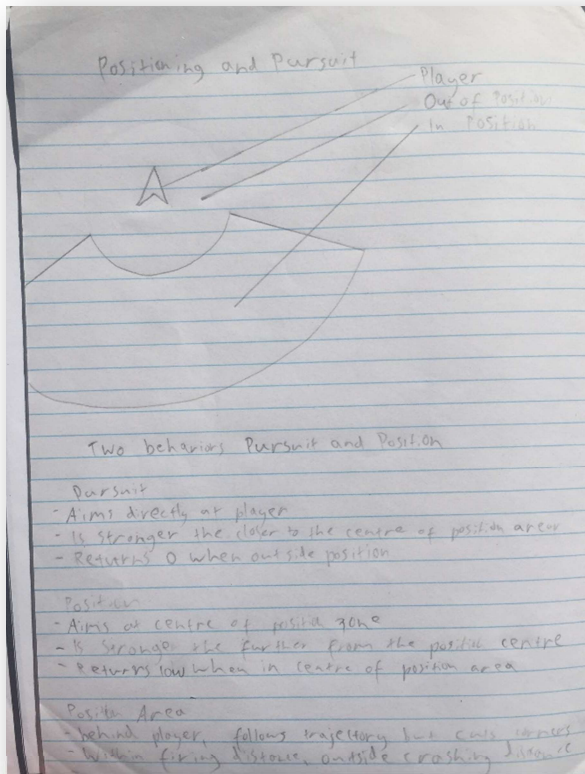


FINAL CHOICE

The AI will be required to decide where the ship should go based on all of these factors. All the necessary information about other enemies and the player is available as it's all occurring within a simulation, and the AI needs to use that information to complete these tasks. Boids are the best choice for this scenario, as they will provide the best AI with the least time and resources. The situation has too many variables and is unsuitable for neural networks, and a state machine would take too long to develop due to the magnitude of the tasks. A boid can be quickly implemented and fine-tuned to work properly due to its weighting of boid behaviours.

AI NAVIGATION

The AI needs to be able to pursue the player without getting too close, or allowing the player to get too far away. I needed to design a new system that would feed the boid system I already have in place a new vector; this would point the boid towards the player and get the ship into position. To do this I made some sketches in my workbook.



Positioning and Pursuit

I drew the player and the position I wanted the AI to be in, the area was a combination of three things:

1. Within a maximum radius
2. Outside a minimum radius
3. Within a certain range of degrees behind the player

Anything within that area was considered to be in position, anything outside that area had to move to get into position. This allowed the enemy ships to pursue the player while in position. I decided the pursuit needed to be handled separately as there's no need for the ai to be aiming and firing at the player if they aren't in position.

I also added an ideal position for the ships to be in, based on the average location of the player. I took sample vectors from the players path every x number of units travelled. Originally, I took a sample every x number of seconds, but if the player was stationary the average would get closer and closer to the player until they crashed. Then I averaged the samples and created a vector that led each ship towards it. The further away from the ideal position, the stronger the vector became.

The combination of these allows the enemies to follow the player and get into position to fire at them without crashing into the player or getting too far away.

VISUAL DESIGN

After looking at other games' visual designs, I have decided to model my art style after Galaxian, but with 3D graphics. I have also decided to incorporate Galaxian's colour coding of different enemy types. Since I decided to buy or download the models and textures, I'll need to find models for at least two different ship types, and fitting textures for each (which will likely come packaged with the models). I'll also need to develop certain visuals based on the art style of the models and textures I find, such as the Graphical User Interface (GUI) or the Heads-Up Display (HUD) as these assets are specific to my game.

VISUALS TO SOURCE

I am going to need to find at least one model and two textures for my game. Ideally, I would have one model for the player's space ship, and more models for the different types of enemy space ships, but as I need to find models of the same style, I may be limited to one model for both the player and the enemies, with different textures to differentiate them. Different types of enemies could have different models and textures, but this may not be possible if I can't find suitable models and textures, and I may have to simplify my game to just one type of enemy.

SPACESHIP MODELS

There were many spaceship models and textures I found and considered while browsing the various websites mentioned previously. Many I passed on either because they didn't include textures, didn't look professional, or didn't fit the game, but these were the ones I seriously considered using:

- Kyan0s' "Spaceship" model was free, looked fantastic, and fit the game, but was the only spaceship of its kind. If I were to use it, I would be limited to having one spaceship model (as I couldn't find other models that would fit this one) and would have to create a texture myself that I could use to make an enemy ship.



IT Multimedia – Spaceships!

- msgamedevelopment's "Pirate Clan Yellowjackets Spaceship Collection" contained a large variety of spaceships and textures that looked exceptional, but the pack costs \$178, and the ships didn't come with a colour variety that fit my arcade-style theme.
- ebalstudios's "Star Sparrow Modular Spaceship" asset pack included a group of spaceship models that could be used to design your own spaceships (wings, tails, thrusters...), 10 different coloured textures that fit the modules, and 13 examples of spaceships with different textures. This pack's art style was brightly coloured and imaginative, and fit my theme exceptionally well, also the example models alone are enough for my game, as I could use one for the player, and three for each type of enemy.

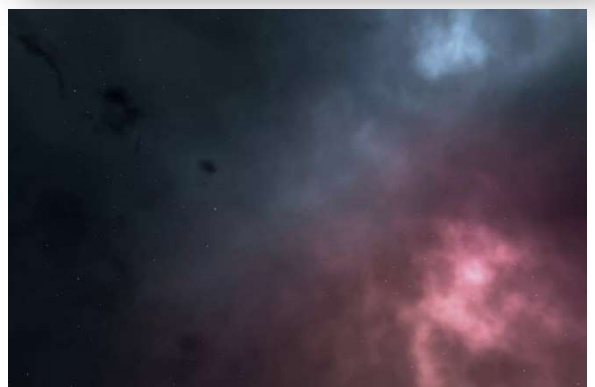
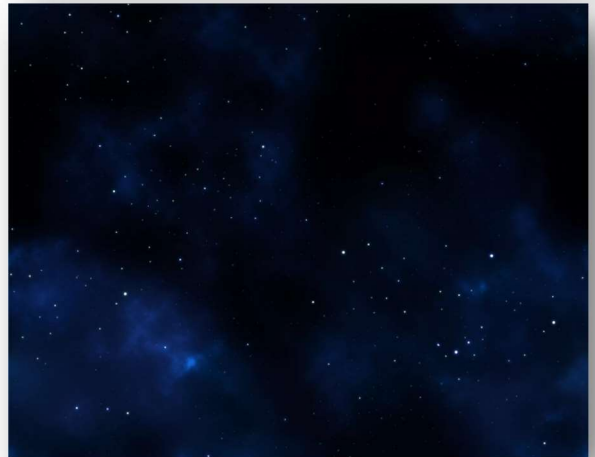
This is the pack I decided to use for my game



SPACE BACKGROUNDS

I also had to find a texture to use for the space background. Unity has a built-in skybox system that allows you to insert an image of a certain type and it will convert it into a background for the game which will be visible but won't interact with the game. I browsed the Unity asset store for compatible images and found these:

- Adam Bielecki's Milky Way Skybox looked realistic and suitable, and its format fits my game, however as you looked around every area looked like every other area, which would make it hard to be aware of what direction you're facing. The background I'm looking for needs to have points of reference such as a planet, or different colours in different areas.
- DinV Studio's Dynamic Space Background looked much better, it offered differently coloured areas which would help the player navigate, and it even offered different layers to provide parallax effect. The problem was that it was designed for a 2D game, so it wouldn't work in my game
- Pulsar Byte's Starfield Skybox was the best choice. It had coloured areas, was in the correct format, and it looked fantastic.
This is the background I decided to use for my game



EXPLOSION VFX

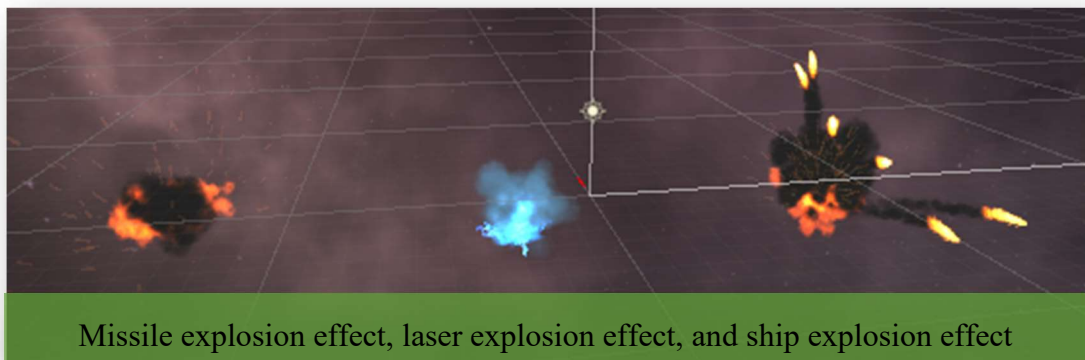
Finally, I had to source visual effects for the explosions. Different parts of the game required different explosive effects, such as a small explosion for missiles, and a large explosion for ships. I browsed the Unity Asset store for appropriate, free assets and found these:

- GAPH's Fx Explosion Pack contains 10 different sprite animations of explosions. What sprite means is that the explosions are 2D, and will always face the player, so they might appear 3D if you don't move around too much. While sprite animations are extremely CPU efficient, which is why they were so commonly used in the early days of computer games, they look extremely out of place in a 3D game when you move around. Modern computers can easily handle 3D computer generated graphics that modern games have, which is why I decided to find some explosions that were more visually impressive.



- Unity Technologies' Unity Particle Pack 5.x. This is an official assets pack made by Unity that contains a variety of particle effects, such as fire, blood, water, and explosions. The explosions in the pack fit perfectly. They came in pre-scripted prefabs that were ready to use, that would procedurally generate a slightly different, 3D explosion every time they were used. The three varieties that I decided to use were the small explosion, for the missiles, plasma explosion, for the lasers, and the large explosion, for the ships. The actual explosions were very colourful, but didn't appear clearly animated, or cartoon-styled, so they fit very well with my game.

These are the visual effects I decided to use for my game



VISUALS TO DEVELOP

Due to the limited amount of time I was left with, after COVID-19, I've decided to source most of my game's visual assets, but there are several elements that I will have to develop myself as they are unique to my game. The main two are the User Interface (UI, the menus the player interacts with to do things like change settings or start the game) and the Heads-Up Display (HUD, an overlay that displays gameplay information to the player such as health or speed)

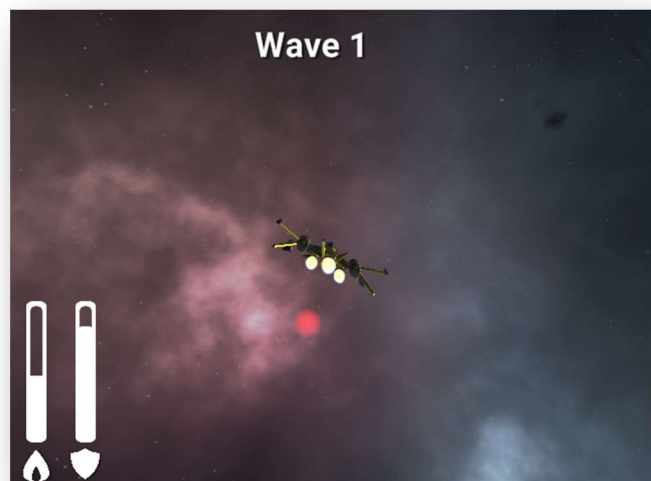
UI DESIGN

For the UI I'm going to use the space skybox as a background, this way I won't have to develop any images or artwork. I also wrote a script for the main menu that slowly rotates the camera in a random direction. The Text needs to stand out on the background, so I've decided to use a bold white font with a shadow to highlight it. I modified a default Unity font called Roboto-Bold SDF and added a drop-shadow. These two design elements can be applied universally to the whole game, bold white text on a space background.



HUD DESIGN

I've decided to keep my HUD as minimalistic as possible, so only a small amount of the screen is covered. The three elements the HUD needs to include are the thrust bar, the shield bar, and the wave counter. The wave counter was simple, I just used the same font as I used in the UI and created a script that would update it to match the current frame. For the thrust and shield bar I needed to design a slider that would move between min and max values. I used Adobe Illustrator to create two icons for the thrust and shield, and to create a background and fill for the sliders.



EVALUATION

The development of any large-scale project such as this requires ongoing evaluation of development and progress. The project will also have a final review at the end, to see what was effective, how it compared to what was intended in the statement of intent, and what could have been improved.

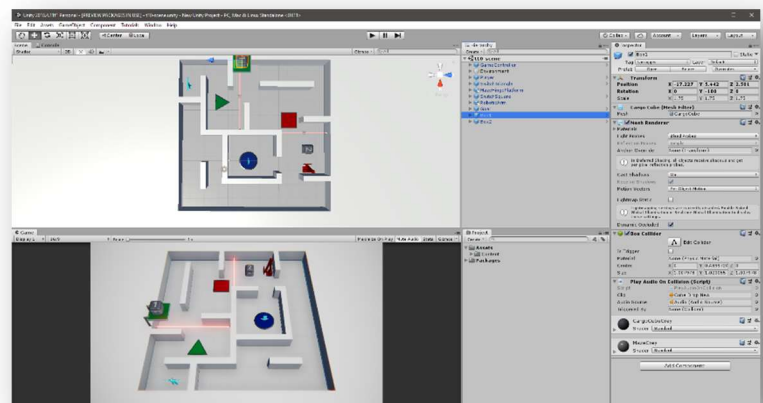
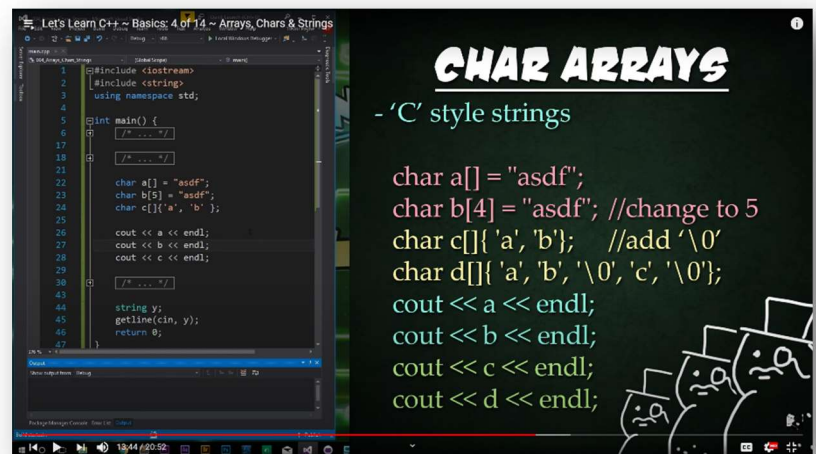
SKILLS ACQUIRED

Over the course of the production of this project, I have acquired and improved a variety of skills. Game development is very new to me, so I had to learn a lot of new programs and features.

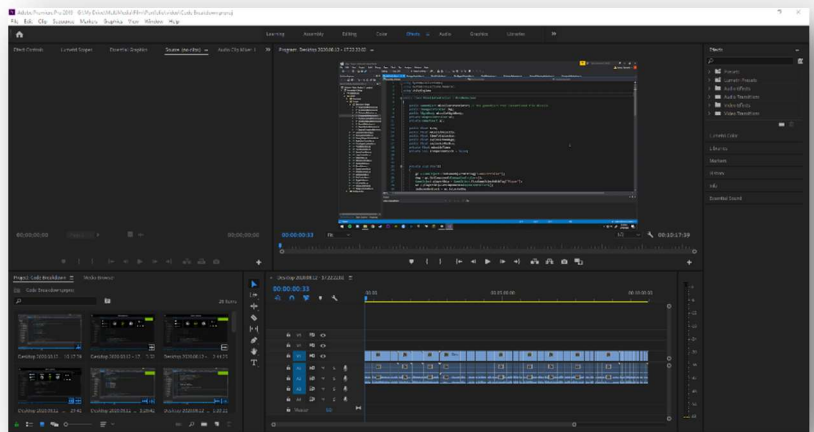
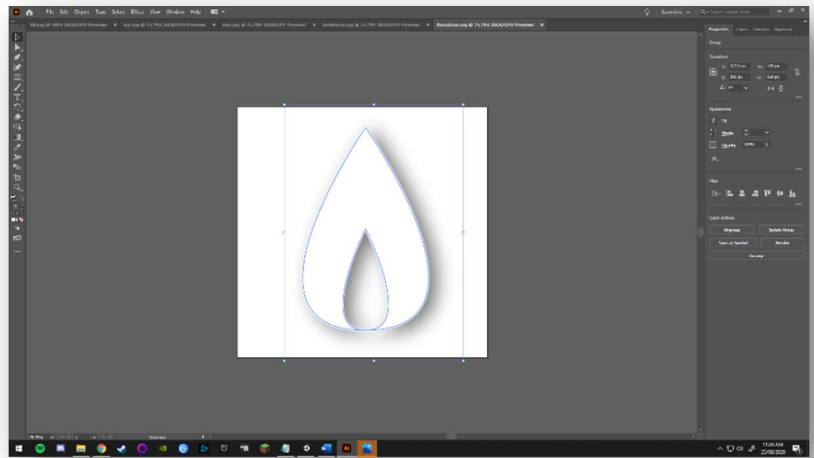
Before I could properly use Unity, I had to learn its programming language, C#. I was recommended to learn C++ first as C# is based on C++, the main difference being that C++ will compile directly to machine code, and C# runs in a virtual machine, so it has some more user-friendly features such as automatic memory management, but is less efficient.

If I were to learn C# first it would be difficult to learn C++ as I would depend on many user-friendly features that didn't exist in C++, and since I plan on having I future in programming, I decided to learn C++ first. I did this by following along a YouTube tutorial series to learn C++, and then a Unity Learn scripting series to see how it applies to Unity, and what differences exist between the languages. As a result, I'm now familiar with C++ and fluent with C#.

I also had to learn to use Unity itself, which meant becoming familiar with its interface, features, and scripting library. Unity's documentation and community were a massive help for this task, Unity has a library of tutorials made by Unity and the community called Unity Learn, and Unity's scripting reference was also extremely helpful, as it gave detailed explanations and examples of unity's built-in methods and classes. Through using Unity to develop the game I have also figured out a lot of features as I gained experience.



I also had the opportunity to use certain adobe applications. I used Adobe Premiere Pro to assemble and edit my Code breakdown video, and I used Adobe Illustrator to design certain UI elements. Although brief, I did gain a familiarity with video editing and vector graphic design. I learned to organise and cut clips using Premiere's tools and I learned to use illustrators pen tools as well as effects such as drop shadows and rounded edges.



PROGRESSIVE DIARY

The ongoing evaluation of my project will be tracked in this Progressive Diary along with problems and solutions, notes on my progression, screenshots and images, and my short-term goals.

- ♦ **Problems and challenges will be marked like this**
- ♦ **Successful and attempted solutions will be marked like this**

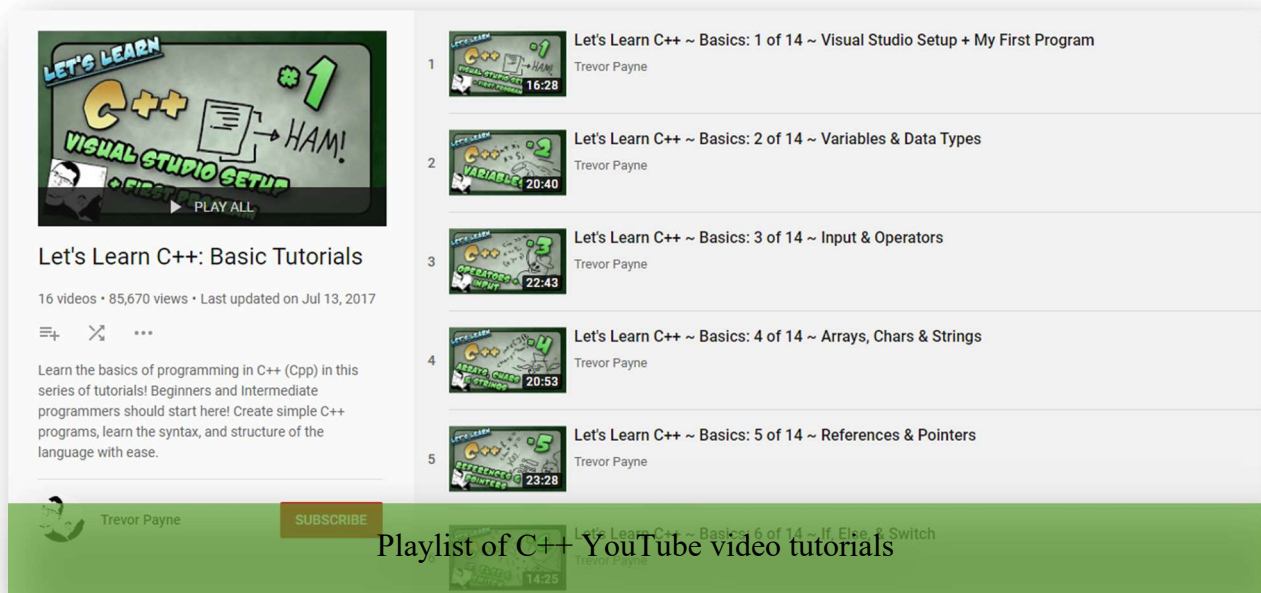
Images will be annotated by these boxes

To see the references for a particular topic, cross reference the date from the process diary with the dates in the references list in the next section to see what sources I used that day.

BEFORE TERM 4

8/10/19 – 14/10/19

- Learned basics of C++ over two weeks (already knew Java from robotics so it wasn't as hard)
- Used Trevor Payne's series on C++ Basics
- Began working on ideas for major project

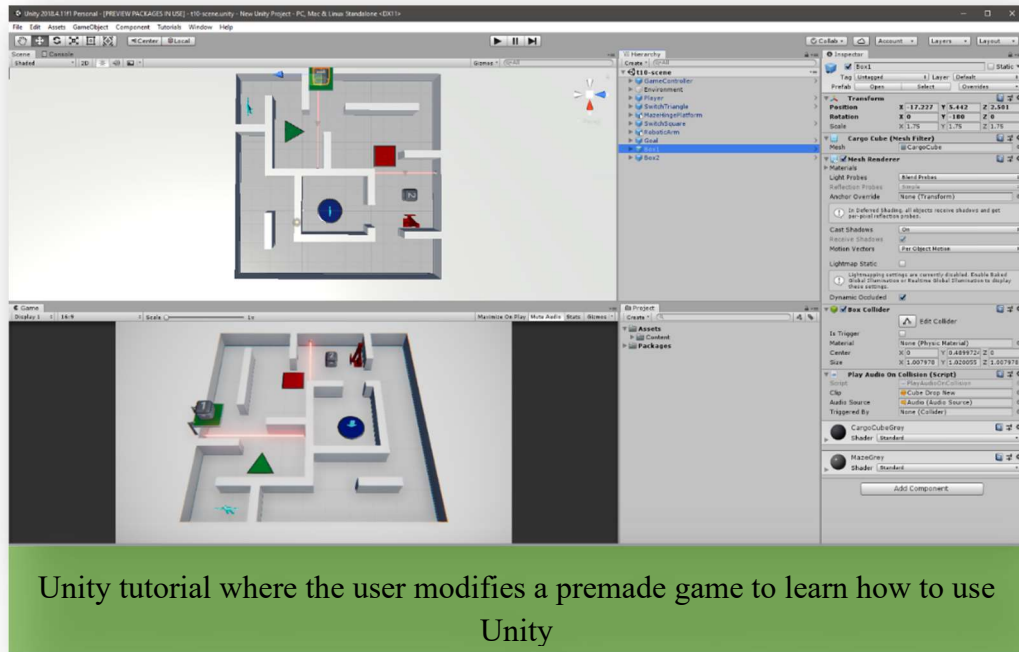


Playlist of C++ YouTube video tutorials

TERM 4

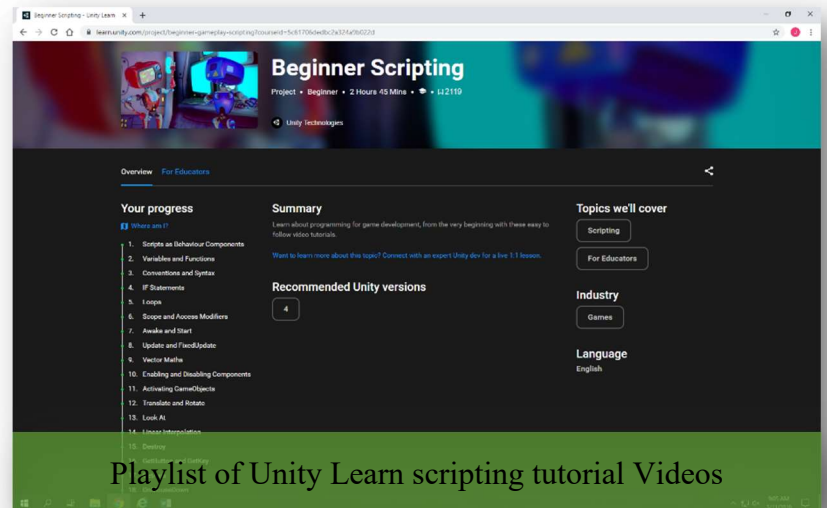
15/10/19

- Downloaded Unity 2018.4.14f1 at home
- Completed Unity Basics Tutorial



17/10/19

- Completed Unity Scripting Tutorial
- ♦ **Discovered Unity actually uses C# not C++**
- ♦ **Researched differences between C++ and C#, found C# is less efficient than C++, but has some additional user-friendly features, and minor syntax changes**



21/10/19

- ♦ **Found out about NESAs restrictions on games (violence, nudity, anything that might be seen as offensive by the adult community)**
- Did research into specifics and found out that most of my ideas weren't suitable.
- ♦ **Modified certain ideas to better suit restrictions by replacing certain enemies with robots**
- Looked into creating engine sounds that match the physics of the game (like making the engine sound get louder and faster when the car speeds up)

IT Multimedia – Spaceships!

- Found RevHeads, an app that simulates engine sounds made by an Australian programmer and sound engineer who has experience creating engine sounds for games like Project CARS, The Crew, Colin McRae: DiRT, Need For Speed: Shift, and others.

22/10/19

- Wrote summary of project to see if it was suitable
- Learned about structs, classes, instances, arrays, and invoking in Unity Scripting Tutorial
- Finished Unity scripting tutorial

28/10/19

- Went to Unity Master Workshop at Camden Haven, where I learned how to make 3D terrains, mass place objects, create cars, and revised scripting.

4/11/19

- Looked into methods of sharing work between school and home computers
- Using GitHub Desktop, I can create a repository I can download easily from either location
- Emailed IT support asking to download GitHub desktop for me as I don't have permission to download anything on a school computer

5/11/19

- Organised and planned portfolio

6/11/19

- Looked into VR in unity
- Decided on VR game where player must guide spaceship through asteroid fields and fight other space ships.
- Looked into Board of Studies rules on what's appropriate for a major project

8/11/19

- Researched VR development plugins for Unity
- Downloaded VRTK and extracted into VR Project Assets folder
- ♦ **After extraction, project always crashes unity**

11/11/19

- Looked into copyright
- Began organising proposal listing all programs I'll need to install.

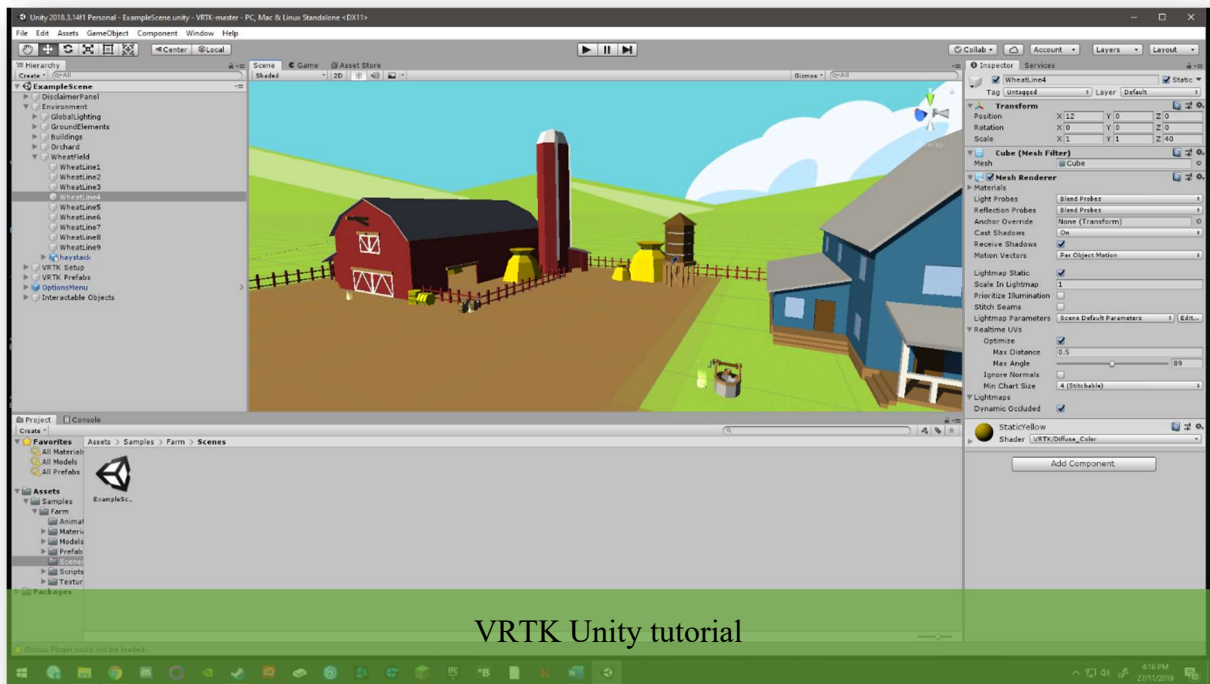
14/11/19

- ♦ **Installed Git for windows to retrieve VRTK, meant to prevent crashing**
- ♦ **Unity continues to crash after adding VRTK**
- ♦ **Found I also need a more recent unity version, 2018.3.10f1 or above**

19/11/19

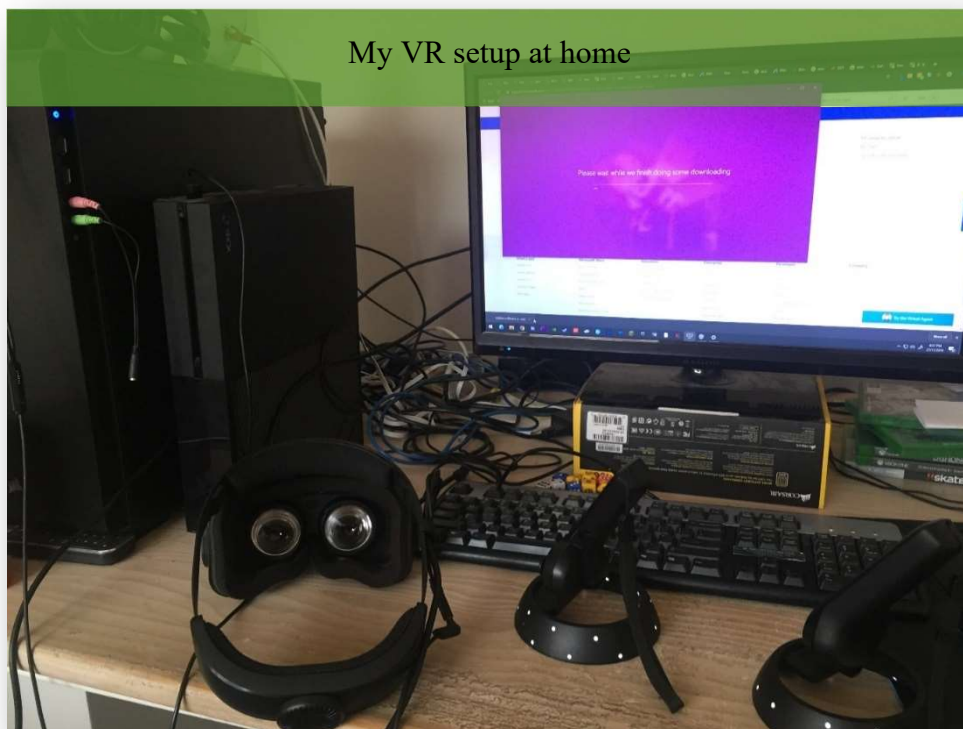
- Downloaded VRTK successfully
- ♦ **Unable to progress through tutorial without VR setup**

IT Multimedia – Spaceships!



25/11/19

- Learned about audio
 - bit/sample rate: no. of times per second a sample is taken
 - bit depth: number of amplitude levels the samples are taken at (8-bit is 256 levels)
 - harmonics: fluctuations in audio waves



IT Multimedia – Spaceships!

26/11/19

- ◆ Began setting up Lenovo Explorer VR headset at desktop
- ◆ Downloaded Windows Mixed Reality and Steam VR
- ◆ Desktop PC's second HDMI port not working, VR headset won't work without HDMI

27/11/19

- ◆ Connected monitor to computer with VGA cable, so VR could use HDMI port
- Plugged Bluetooth 3.0 adapter into computer to connect VR controllers
- ◆ Controllers won't connect, Steam VR doesn't recognise the Bluetooth adapter

29/11/19

- Handed in new project proposal for VR project

12/12/19

- Finished and handed in Gantt chart
- Handed in portfolio

2019-2020 HOLIDAY

22/12/19

- **Went on holiday to Melbourne with family, unable to work on project**

6/1/20

- **Returned from holiday**

7/1/20

- ♦ **Continued trying to figure out why controllers aren't working**
- ♦ **Discovered VR controllers require Bluetooth 4.0**

8/1/20

- ♦ **Unsuccessfully went to local Jaycar and JB Hifi to find Bluetooth 4.0 adapter**
- ♦ **Ordered Bluetooth 4.0 adapter (\$13.95 + \$5.56 shipping)**

TERM 1

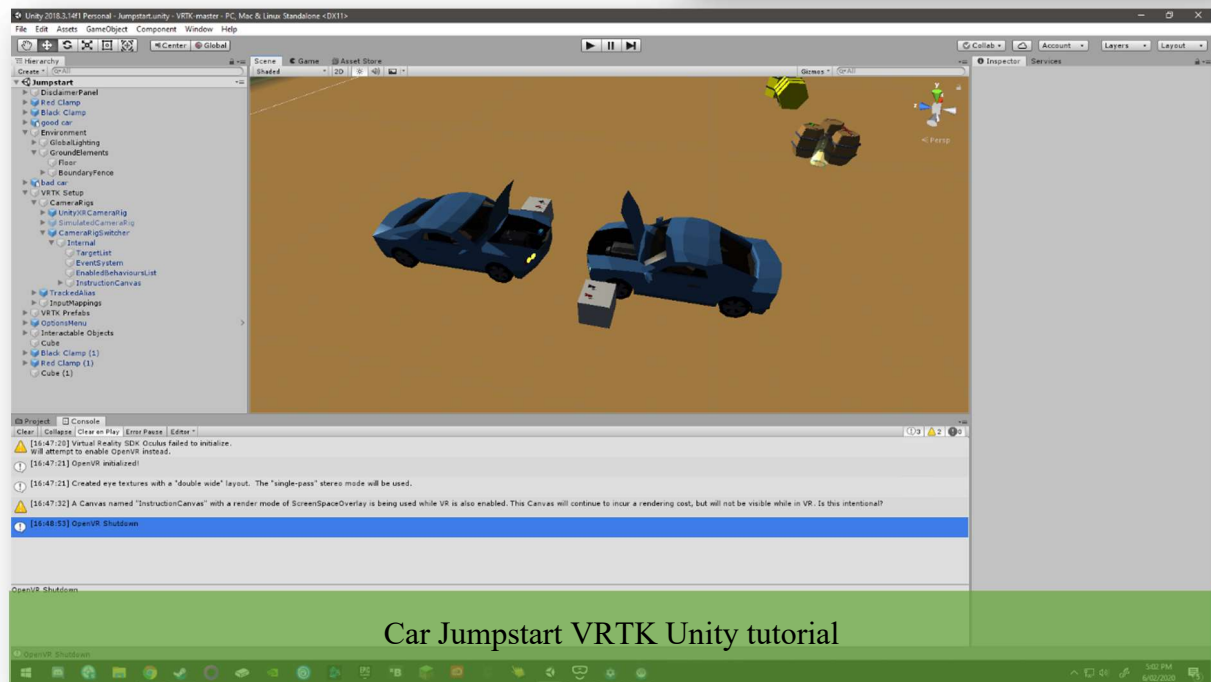
4/02/20

- Bluetooth 4.0 adapter arrived
- Nearly two weeks later than estimated, apparently the delivery was delayed due to the roads being closed off because of the bush fires
- Set up Bluetooth controllers and began VRTK course



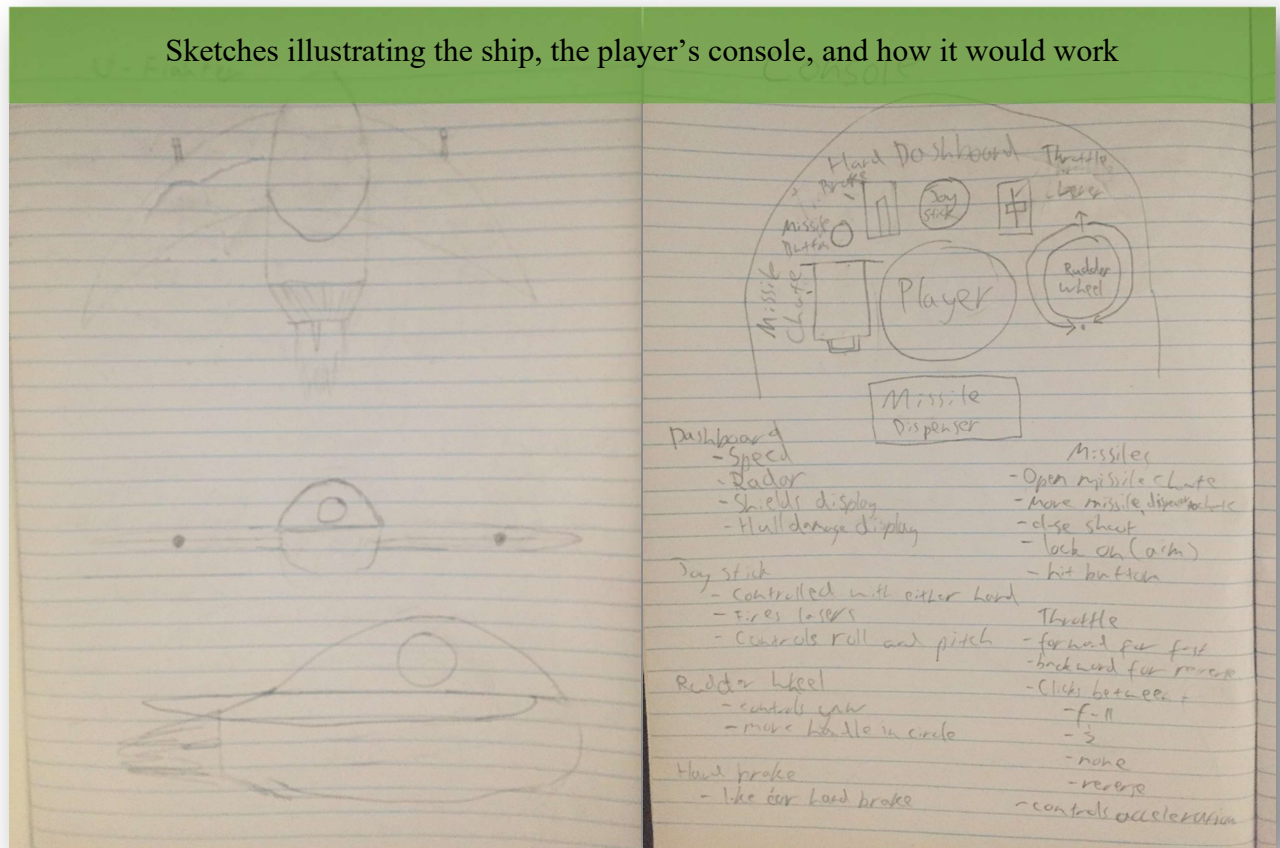
5/02/20

- Followed tutorial to make basic VR car jumpstart game



8/02/20

- Began working on space ship game
- Drafted the ship and what the console would look like



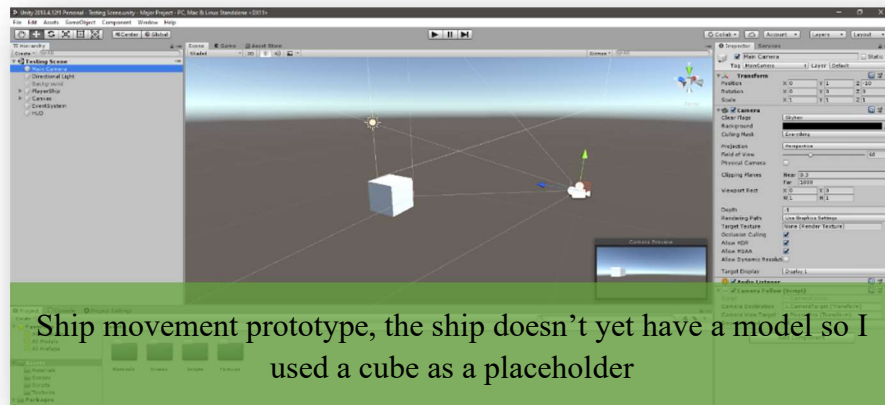
12/02/20

- Attempted to create the console, with a moving joystick and lever
- Stuck on creating a joystick that VR controllers can interact with, that will pivot on a joint, instead of floating in hand
- Began reading through official VRTK documentation to find features that would suit purpose
- ♦ **Documentation assumes an amount of knowledge of unity and VR development that I don't yet have**
- ♦ **Decided that to learn how to program a VR game would take far longer than I initially thought, and would exceed my given timeframe. Downscaled project so that it would not include VR, but would still fulfil original intent, and have same basic game structure.**

13/02/20

IT Multimedia – Spaceships!

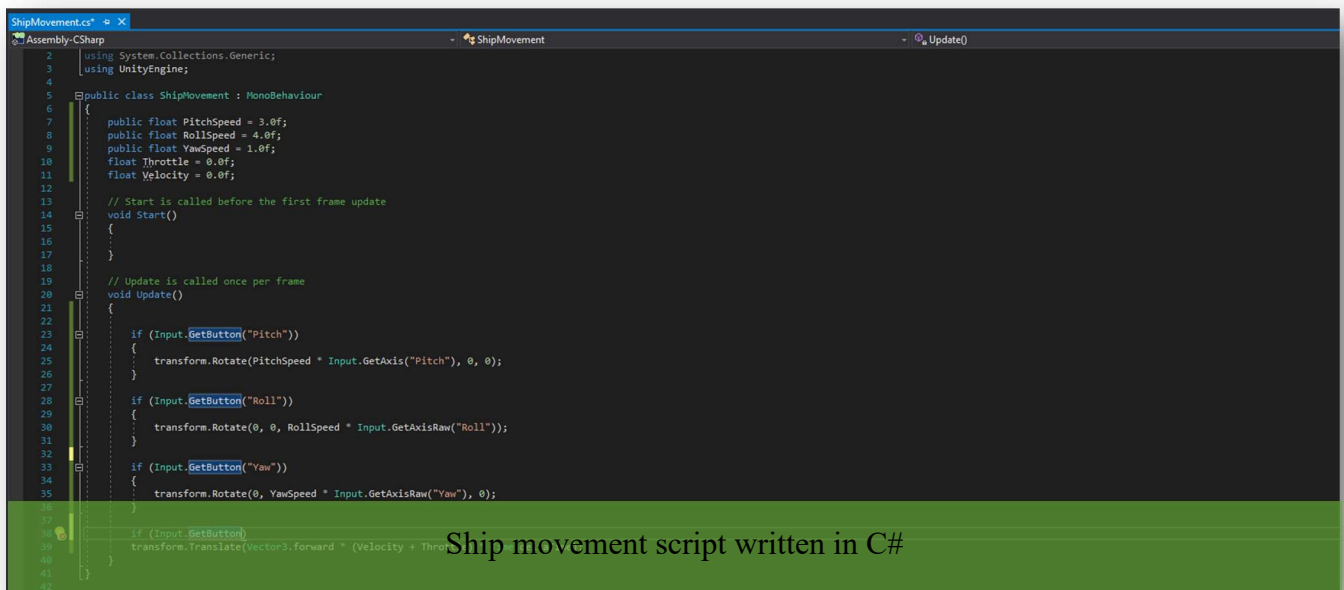
- Made prototype ship, used cube as a model temporarily



- Scripted basic ship movement, can move left and right, forwards and backwards
- Wrote camera script that made it follow the ship while looking at it
- Used Unity Scripting API extensively to learn the different methods and functions in Unity

14/02/20

- Improved ship's movement, so player can control roll, pitch, yaw, and throttle
- Smoothed camera so that it doesn't jolt around with ship, but will glide into position, also shows speed of ship by distance from camera



20/02/20

- Attempted to have camera orbit ship with mouse when holding right click
- ♦ Camera orbit clashed with camera follow and made controlling ship too confusing
- Made camera roll smoothly

- ◆ Made new camera script that allowed player to swap between free orbital and following camera
- <https://www.youtube.com/watch?v=urNrY7FgMao>
- https://www.youtube.com/playlist?list=PLiyfvmjtjWC_V_H-VMGGAZi7n5E0gyhc37

1/03/20

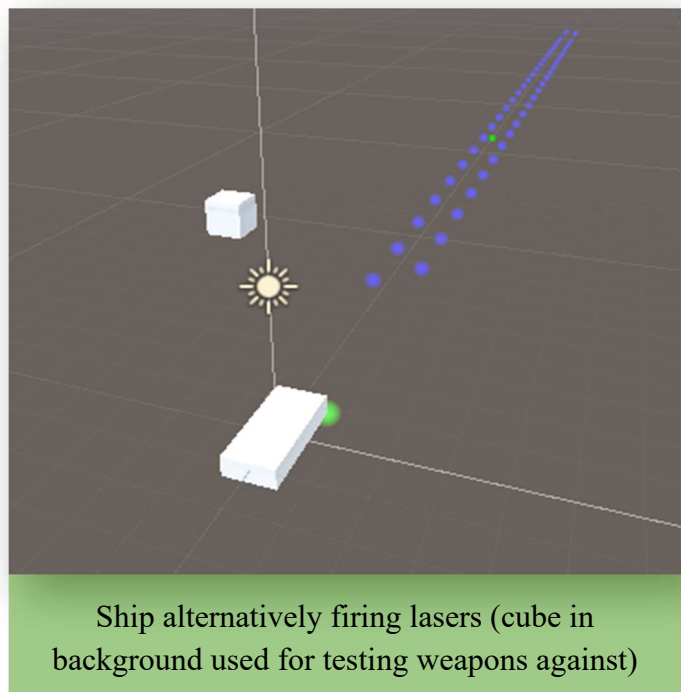
- Designed how HUD would look
- Made prototype throttle bar
- ◆ **Found ship was hard to precisely aim with button controls, which were unable to move the ship gradually, but with either full speed or no speed.**
- ◆ **Removed camera orbit and replaced with mirror view button, so mouse could be used to control gradual ship movement**
- ◆ **Made smooth ship control with mouse**
- Smoothed out ship rotations so they are constant irrespective of frame rate

14/03/20

- Began designing weapons
- Researched other spaceship games weapons systems
- Need to make a missile with physics, and a laser with raycasting

24/03/20

- Began prototype weapons system
- ◆ **Decided against making ray casting lasers (like real life lasers, will cast a ray from ship, and anything touching the ray during that instant is hit) as they were less interesting visually to look at, and were always 100% accurate and moved instantly, which negated the need for the player to see where the enemies were moving and predict where they might be shortly.**
- ◆ **Designed system to fire fast-moving projectiles which can alternate-fire lasers from twin laserguns**



25/03/20

- Learned how to reference variables from different scripts
- Created a missile that homes in on a target
- Began missile targeting system

TERM 2

16/04/20

- Made missile targeting system cast a sphere with a large radius and target the game object closest to the player ship that the sphere passes
- Learned how to reference different scripts

21/04/20

- Further worked on targeting system
- ♦ **Discovered because missile has Rigidbody, a built-into-Unity Physics component, it will collide with the player ship and come off at the wrong angle**
- ♦ **Can't replace Rigidbody with collider component because missileController script uses Rigidbody physics functions**

27/04/20

- ♦ **Moved spawn location so missile doesn't touch player ship**
- Added Enemy tag to enemies and adjusted targeting system to only register gameObjects with Enemy tag
- ♦ **Targeting is difficult to lock on at long distances and always chooses the closest object at close distances**
- ♦ Need to look into projecting all objects in the view frustum onto the viewing plane and check in screen space if they intersect with the search area

28/04/20

- Added GameController empty to scene to hold and run UIController script, which runs the game UI and new DamageController script, which calculates the damage taken and creates explosions when necessary (such as when a missile collides with a ship)

```
public class DamageController : MonoBehaviour
{
    DamageControl script, creates an explosion when Explosion method is called
    public GameObject explosionPrefab;
    public StatController sc;

    // Start is called before the first frame update
    void Start()
    {
    }

    // Update is called once per frame
    void Update()
    {
    }

    public void Explosion(Transform expTransform, float explosionRadius, float explosionDamage)
    {
        RaycastHit[] hitsWithinRadius = Physics.SphereCastAll(expTransform.position, explosionRadius, expTransform.forward);
        foreach(RaycastHit hit in hitsWithinRadius)
        {
            if (hit.transform.gameObject.tag == "Enemy" || hit.transform.gameObject.tag == "Player")
            {
                hit.transform.gameObject.GetComponent<StatController>().TakeDamage(explosionDamage);
            }
            else
            {
                Destroy(hit.transform.gameObject);
            }
        }
        //instantiate explosion prefab
        Instantiate(explosionPrefab, expTransform.position, expTransform.rotation);
    }
}
```

IT Multimedia – Spaceships!

- Made missiles create explosions on collision
- Explosions damage enemies and players, and destroy any other object (damage system not yet implemented) **Tweaked targeting system to prioritise targets closest to sphere cast instead of closest to player, now works much more smoothly**
- Need to make it so when objects collide each object's remaining health is subtracted from the other

14/05/20

- Added health system, where each ship has a set max shield strength that regenerates slowly over time. Any damage taken is subtracted from the shield, and once it goes below 0 the ship is destroyed.
- Changed system so when an object is destroyed by a missile, instead of being disabled, it destroyed completely to save processing power
- Missiles now create explosions which deal damage to everything within the radius
- Missiles will also explode after a certain amount of time passes since being fired
- ♦ **Lasers are supposed to deal damage upon collision, but the collision never registers for some reason and as a result no damage is taken**

1/06/20

- ♦ **Discovered game objects need Rigidbody components to register collisions, added Rigidbody to lasers**
- ♦ **After adding RigidBodies to ships the lasers now knock them around**
- ♦ **Turned lasers into triggers so that they don't physically interact with the ships, and instead pass through**
- Now ships aren't knocked around and can be destroyed with lasers and missiles
- Ideally, I would have previously used Unity's built-in physics system, had I known it existed, but the system I designed is equally functional and changing the system would require a lot of time, so I'll use mine as long as no other issues appear

2/06/20

- Looked into possible methods of designing AI: considered a neural network, state machines, and boid algorithms
- ♦ **Discovered by scaling the parent object of the ship change the shape (make the ship look rectangular) it was scaling the position and scale of all the children**
- ♦ **To fix this I removed the mesh component from the parent and reset the scales, and added a new child to the parent that carried the mesh**
- ♦ **Discovered missiles still knock enemy ships around**
- ♦ **Added a capsule collider to the missile that would detonate the missile before it touched the enemy ship**
- Began designing AI system using boids

5/06/20

- Continued coding FlockController, enemies will now create an array of all objects within their context radius (may be different for different enemies)

- ♦ **Arrays can't be changed easily after being initialised, so currently a new array is made every frame, which is very CPU intensive**
- ♦ **Replaced arrays of objects with a single list of objects, which can be easily changed, so enemies can be added and removed when they are created and destroyed.**

18/06/20

- Tested context checking code
- Wrote cohesion behaviour, which tells enemies to go closer to other enemies. When testing this causes any enemies near others to crash together (intended, as avoidance behaviour has not yet been written)



- Removed enemy ship rigidbodies, as they caused drifting upon collision, currently they merge together
- Wrote Alignment behaviour, tells enemies to align with nearby enemies
- Wrote Avoidance behaviour, which tells enemies to avoid other enemies
- ♦ **Currently only one behaviour can be selected at a time, and enemies snap to the direction given by the behaviour**
- Need to code composite behaviour, that incorporates all three behaviours and weights them accordingly. The averaging of directions will allow the enemy to gradually orient themselves to the desired direction, instead of snapping

IT Multimedia – Spaceships!

```
5 [CreateAssetMenu(menuName = "Flock/Behaviour/Cohesion")]
6 public class CohesionBehaviour : FlockBehaviour
7 {
8     public override Vector3 CalculateMove(FlockAgentController agent, List<Transform> context, FlockController flock)
9     {
10         //if no neighbours return no adjustment
11         if (context.Count == 0)
12         {
13             //Debug.Log(agent.name + " is zeroed");
14             return Vector3.zero;
15         }
16
17         //add all points together and average
18         Vector3 cohesionMove = Vector3.zero;
19         foreach (Transform item in context)
20         {
21             cohesionMove += item.position;
22         }
23         cohesionMove /= context.Count;
24
25         //create offset from agent position
26         cohesionMove -= agent.transform.position;
27         return cohesionMove;
28     }
29 }
30
```

CohesionBehaviour Script

```
5 [CreateAssetMenu(menuName = "Flock/Behaviour/Alignment")]
6 public class AlignmentBehaviour : FlockBehaviour
7 {
8     public override Vector3 CalculateMove(FlockAgentController agent, List<Transform> context, FlockController flock)
9     {
10         //if no neighbours maintain current alignment
11         if (context.Count == 0)
12         {
13             //Debug.Log(agent.name + " is zeroed");
14             return agent.transform.forward;
15         }
16
17         //add all alignments together and average
18         Vector3 alignmentMove = Vector3.zero;
19         foreach (Transform item in context)
20         {
21             alignmentMove += item.forward;
22         }
23         alignmentMove /= context.Count;
24
25         return alignmentMove;
26     }
27 }
28
29
```

AlignmentBehaviour Script

```
5 [CreateAssetMenu(menuName = "Flock/Behaviour/Avoidance")]
6 public class AvoidanceBehaviour : FlockBehaviour
7 {
8     public override Vector3 CalculateMove(FlockAgentController agent, List<Transform> context, FlockController flock)
9     {
10         //if no neighbours return no adjustment
11         if (context.Count == 0)
12         {
13             //Debug.Log(agent.name + " is zeroed");
14             return Vector3.zero;
15         }
16
17         //add all points together, average, and move away
18         Vector3 avoidanceMove = Vector3.zero;
19         int nAvoid = 0;
20         foreach (Transform item in context)
21         {
22             if (Vector3.SqrMagnitude(item.position - agent.transform.position) < agent.SquareAvoidanceRadius)
23             {
24                 nAvoid++;
25                 //OLD avoidanceMove += (agent.transform.position - item.position); // add item avoidance vector to avoidancemove
26
27                 //adds item avoidance vector multiplied by it's avoidance radius over its distance from the object, so the magnitude of that vector increases exponentially
28                 avoidanceMove += (agent.transform.position - item.position) * agent.SquareAvoidanceRadius / Vector3.SqrMagnitude(item.position - agent.transform.position);
29             }
30         }
31
32         if (nAvoid > 0)
33         {
34             avoidanceMove /= nAvoid; // divide avoidance move by number of avoidance vectors inside, to get average vector that moves away from items
35         }
36         return avoidanceMove;
37     }
38 }
39
```

AvoidanceBehaviour Script

28/06/20

```

5 [CreateAssetMenu(menuName = "Flock/Behaviour/Composite")]
6 public class CompositeBehaviour : FlockBehaviour
7 {
8     public FlockBehaviour[] behaviours;
9     public float[] weights;
10
11     public override Vector3 CalculateMove(FlockAgentController agent, List<Transform> context, FlockController flock)
12     {
13         //set up move
14         Vector3 move = agent.gameObject.transform.forward;
15         if (!PauseMenu.GameIsPaused)
16         {
17             if (weights.Length != behaviours.Length)
18             {
19                 Debug.LogError("Data mismatch in " + name, this);
20                 return Vector3.zero;
21             }
22
23             // reset move
24             move = Vector3.zero;
25
26             //iterate through behaviours
27             for (int i = 0; i < behaviours.Length; i++)
28             {
29                 Vector3 partialMove = behaviours[i].CalculateMove(agent, context, flock) * weights[i];
30
31                 if (partialMove != Vector3.zero)
32                 {
33                     if (partialMove.sqrMagnitude > weights[i] * weights[i] // if the length of sqr magnitude is greater than that behaviours weighting
34                     {
35                         partialMove.Normalize();
36                         partialMove *= weights[i]; // then partial move's length is set to that behaviours weighting
37                     }
38
39                     move += partialMove; // and that partial move is added to move
40                 }
41             }
42             return move;
43         }
44     }
45 }

```

CompositeBehaviour Script

- ◆ Scripted a composite behaviour that incorporates all three behaviours
- ◆ Currently causes ships to twitch constantly, cohesion might be cause as turning it off stops it
- ◆ Created new cohesion script that uses SmoothDamp, a Unity function that makes a vector takes a certain amount of time to move from one orientation to another
- ◆ Prevents twitching, but ships still move erratically
- Discovered Alignment Behaviour was adjusting the ships to align with each other's up direction, instead of each other's forward direction
- ◆ Adjusted composite behaviour so ships will generally stick together and not twitch
- Created script that causes ships to stay within a certain radius of the player, so they don't drift too far apart

```

5 [CreateAssetMenu(menuName = "Flock/Behaviour/PositionRadius")]
6 public class PositionRadiusBehaviour : FlockBehaviour
7 {
8     Vector3 positionRadiusVector;
9     public override Vector3 CalculateMove(FlockAgentController agent, List<Transform> context, FlockController flock)
10     {
11         float distanceToPlayer = Vector3.Distance(flock.player.transform.position, agent.transform.position);
12         Vector3 toPlayer = flock.player.transform.position - agent.transform.position;
13
14         if (distanceToPlayer < flock.innerPlayerRadius) // when inside inner radius, move away from player with linearly increasing strength inversely relative to distance from player
15         {
16             positionRadiusVector = toPlayer * (distanceToPlayer - flock.innerPlayerRadius);
17         }
18         else if (distanceToPlayer > flock.outerPlayerRadius) // when outside outer radius, move towards player with linearly increasing strength relative to distance from player
19         {
20             positionRadiusVector = toPlayer * (distanceToPlayer - flock.outerPlayerRadius);
21         }
22         else
23         {
24             positionRadiusVector = Vector3.zero;
25         }
26         return positionRadiusVector;
27     }
28 }

```

PositionRadiusBehaviour Script

- Replaced avoidance radius multiplier, which multiplied the context radius by a certain percentage, with a constant radius so different size ships can be given different avoidance radiuses easier
- Need to create script that makes AI aim at player
- Finished stat controller

```
7: public class GameOverMenu : MonoBehaviour
8: {
9:     public static bool GameIsPaused = false;
10:     public GameObject gameOverMenuUI;
11:     public GameObject hud;
12:     public TMP_Text currentWaveText;
13:     public TMP_Text highScoreText;
14:
15:     public void GameOver(int waveIndex)
16:     {
17:         Cursor.visible = true;
18:         hud.SetActive(false);
19:         gameOverMenuUI.SetActive(true);
20:         currentWaveText.text = "YOU REACHED WAVE " + waveIndex;
21:         if (PlayerPrefs.HasKey("highscore"))
22:         {
23:             if (waveIndex > PlayerPrefs.GetInt("highscore"))
24:             {
25:                 PlayerPrefs.SetInt("highscore", waveIndex);
26:             }
27:         }
28:         else
29:         {
30:             PlayerPrefs.SetInt("highscore", waveIndex);
31:         }
32:         highScoreText.text = "YOUR HIGHEST WAVE IS " + PlayerPrefs.GetInt("highscore");
33:     }
34:
35:     public void PlayAgain()
36:     {
37:         Time.timeScale = 1f;
38:         GameIsPaused = false;
39:         Debug.Log("Loading Game...");
40:         SceneManager.LoadScene("Game");
41:         Cursor.visible = false;
42:     }
43:
44:     public void LoadMenu()
45:     {
46:         Time.timeScale = 1f;
47:         GameIsPaused = false;
48:         Debug.Log("Loading Menu...");
49:         SceneManager.LoadScene("Menu");
50:     }
51:
52:     public void Quit()
53:     {
54:         Debug.Log("Quitting");
55:         Application.Quit();
56:     }
57:
58: }
59:
60:
```

GameOverMenu Script

1/07/20

- Received help from teacher breaking down the portfolio
- Found out markers have unlimited time to view actual project, previously I had misread the marking guidelines to assume the 6 minutes of media viewing was for the project itself
- Need to make one 6 minute narrated video that details the differences between each build
- Need to add descriptions under headings
- Comparison of products should use PMIs, positive minuses interests

- Highlight issues encountered in log with symbols or colour coding

6/07/20

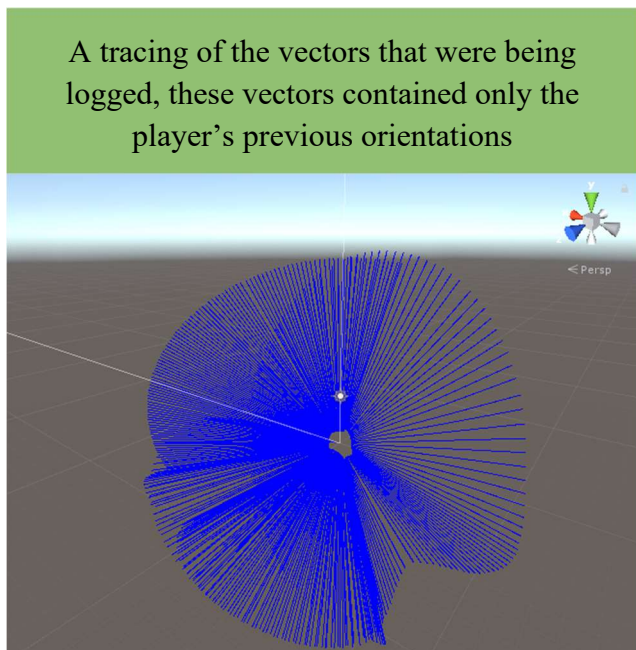
- Designed system that will allow ships to get into position behind the player, and pursue them
- Added position radius behaviour, that kept enemies outside of a small radius of the player and within a large radius of the player so that they could more easily shoot at the player
- ♦ **Need to add position centre behaviour, that samples the player's vector every x distance since last sample, and averages to make a position for the ship to travel towards, and direction for the ship to head**

7/07/20

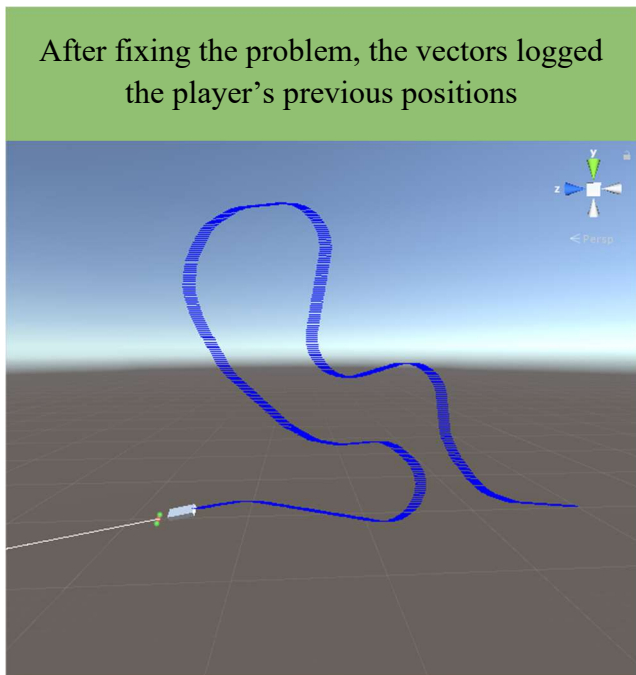
- ♦ **Attempted to track player path by logging samples in a list every set distance, found using vectors makes it difficult to find distance between**
- ♦ **List is not logging points as expected**

9/07/20

- ♦ **Attempted to track transforms instead of vectors, found that because we only need positions, and transforms require rotation, position and scale, transforms were not suitable**
- ♦ **Used Debug.DrawRay to trace the vectors that were being logged, discovered the orientations were being logged instead of the positions**
- Previously script added the players transform.forward vector to the list, now adds transform.position
- Added vector that is the average of the last 100 samples



- ♦ **Found that composite behaviour became unstable and twitchy when too many behaviours were added**
- ♦ **Attempted to make a new composite behaviour that was smoother than the previous one, that takes a specified amount of time to move from one orientation to another, new composite behaviour slowed movements but didn't fix issue**
- ♦ **Lowered new behaviours weightings and modified scripts so their strength grew exponentially. This way when the behaviour created a new direction for the ship to travel, it is ignored at first and as it grows in strength, the ship will slowly orient further in that direction**



10/07/20

- ♦ **Added descriptions of sections under each major title in portfolio**
- ♦ **Images and annotations were separating when changes were made surrounding images**
- ♦ **Grouped annotations with images in process diary to prevent them from separating**
- Added more images and annotations

11/07/20

- ♦ **Needed clear way of differentiating different logs in process diary**
- ♦ **Categorised bullet points into general, issues, and solutions**
- Went through process diary and began changing bullet points to suit their contents, and separating lines that had the problem and solution into separate lines

12/07/20

```

5 public class EnemyWeaponController : MonoBehaviour
6 {
7     public GameObject laserProjectile;
8     public GameObject leftLaserGun;
9     public GameObject rightLaserGun;
10    public GameObject laserInstance; // the instance of the last laser fired
11    private GameObject player;
12    public float laserFireRate = 0.2f; //secs per laser
13    public float angleDifference; // the minimum angle difference between the direction the ship is facing, and the direction the player is from the ship
14    public float safetyRadius; // The radius within which there must be no allies for the AI to shoot
15    private float nextLaserFire; // seconds until next fire
16    private bool shotIsLeft; // while true, next shot will be left, while false, next shot will be right
17
18    // Start is called before the first frame update
19    void Start()
20    {
21        shotIsLeft = true;
22        nextLaserFire = laserFireRate;
23        player = GameObject.FindGameObjectWithTag("Player");
24    }
25
26    void Update()
27    {
28        nextLaserFire += Time.deltaTime;
29
30        // if roughly facing player, and no allies are in the way, fire at player
31        Vector3 toPlayer = player.transform.position - (leftLaserGun.transform.position + rightLaserGun.transform.position)/2; // vector from average between laser guns, and player
32        if(Vector3.Angle(transform.forward, toPlayer) <= angleDifference) // if the ship is approximately facing the player
33        {
34            RaycastHit[] hits = Physics.SphereCastAll((leftLaserGun.transform.position + rightLaserGun.transform.position) / 2, safetyRadius, toPlayer, toPlayer.magnitude);
35            bool clearFiringPath = true;
36            foreach (RaycastHit hit in hits)
37            {
38                if(hit.collider.gameObject.tag == "Enemy" && hit.collider.gameObject != gameObject)
39                {
40                    clearFiringPath = false;
41                }
42            }
43            if (clearFiringPath && nextLaserFire >= laserFireRate)
44            {
45                if (shotIsLeft == true)
46                {
47                    laserInstance = Instantiate(laserProjectile, leftLaserGun.transform.position, Quaternion.LookRotation(toPlayer, transform.up));
48                    laserInstance.GetComponent<LaserController>().laserInstantiator = gameObject;
49                    shotIsLeft = false;
50                }
51                else
52                {
53                    laserInstance = Instantiate(laserProjectile, rightLaserGun.transform.position, Quaternion.LookRotation(toPlayer, transform.up));
54                    laserInstance.GetComponent<LaserController>().laserInstantiator = gameObject;
55                    shotIsLeft = true;
56                }
57                nextLaserFire = 0f;
58            }
59        }
60    }
61 }
62

```

EnemyWeaponController script

- Completed script that allowed enemies to shoot at player
- ♦ Found lasers were colliding with the ship that fired them if the ship was moving
- ♦ Modified laser projectile script so lasers wouldn't affect the ship that fired them
- Applied same fix to missiles
- ♦ Found missile was controlled by two separate scripts unnecessarily
- ♦ Consolidated two scripts

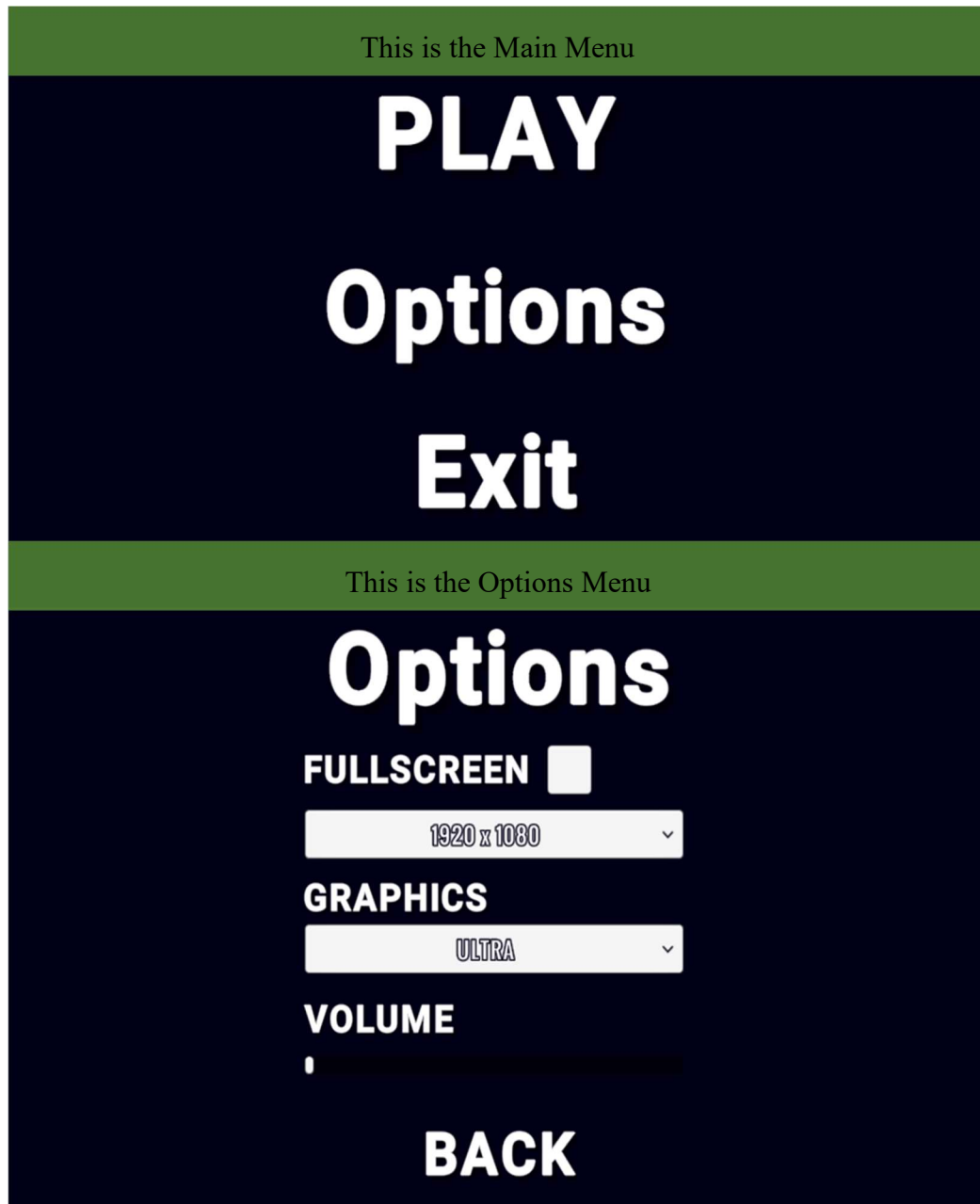
```

40    dmg.Expllosion(transform, explosionRadius, explosionDamage);
41    Debug.Log("Boom");
42    }
43    }
44    private void FixedUpdate()
45    {
46        missileRigidbody.velocity = transform.forward * missileVelocity;
47
48        if(independentLock)
49        {
50            //var missileTargetRotation = Quaternion.LookRotation(missileTarget.gameObject.transform.position - transform.position);
51            var missileTargetRotation = Quaternion.LookRotation(wc.targetCollider.gameObject.transform.position - transform.position);
52
53            missileRigidbody.MoveRotation(Quaternion.RotateTowards(transform.rotation, missileTargetRotation, turn));
54        }
55    }
56
57    }
58
59    private void OnTriggerEnter(Collider collider)
60    {
61        if (collider.gameObject != missileInstantiator) // if collider is not the ship that fired the laser's
62        {
63            //call explosion method
64            Debug.Log("triggered");
65            dmg.Expllosion(transform, explosionRadius, explosionDamage);
66        }
67    }

```

Two key sections of the consolidated MissileController script: movement and collisions

- Began scripting collisions in DamageController, so it could detect collisions between objects and decide if they bounce away, or one is destroyed and the other takes damage
- Found when using rear-view button the ship's reticle is in the way
- Moved button to be behind rear-view camera
- Created start menu, and script that opens game scene, options menu, and allows user to quit



13/07/20

- Scripted options menu so volume, graphics quality, resolution, and fullscreen toggle function
- Began researching game audio
- Began designing and scripting pause menu
- ♦ **Unity's built-in timescale system is supposed to allow you to speed up, slow down, or in this case pause the game, but with my game it doesn't fully work. Most systems stop, but the enemies' movement continues**

- Found that most movement systems (like the players) multiplies any desired movements by Time.deltaTime (the amount of time passed since the last frame) which equals 0 if the timescale is set to 0 (1 is normal speed, 0.5 is half, 0, Is paused...). These movements are paused with the timescale, but the enemy's AI can't be multiplied by Time.deltaTime as it works out where it should be each frame and independently teleports to that position
- ♦ **Modified enemy's AI so instead of calculating where it should be and going there every frame, it checks if the game is paused first, and if it is, does nothing that frame.**
- Began scripting wave and score system

14/07/20

- Browsed 3D model websites TurboSquid, Free3d, Sketchfab, and cgtrader for appropriate spaceship models
- Decided on ebal studios' Star Sparrow Modular Spaceship Pack, which included 13 ship models and 10 differently coloured textures. Fitted games modern arcade game style, buying a pack instead of separately ensured all were the same style, bright colours allow for quick and easy differentiation of units, files are low poly, which means it requires less computing effort to render them, and pack was free without royalties or permission required.
- ♦ **Need something for ships to spawn in, either a space station, or a warp gate. Can't find any suitable models**
- Continued establishing waves and points, designed for only one type of ship until other ships are ready

TERM 3

20/07/20

- Added Visual Design section to portfolio
- Expanded Artificial Intelligence section
- Expanded Comparison of Existing Products section
- Updated references

24/07/20

- ♦ Decided I'd have ships warp in on their own, as obtaining a model for a suitable space station proved too difficult
- Began framework for ship spawning code

29/07/20

- WaveController now spawns groups of ships of different types, in random locations around the player
- ♦ Ship formation only spawns facing north, regardless of orientation around player

4/8/20

- Began work on code breakdown video
- Broke down key scripts

7/08/20

- ♦ Spawned ships as children of a new parent GameObject, and face the GameObject towards the player. Children move as if attached to a parent, so they all pivot around the parent and face the player.

8/08/20

```

106 // Generate Random Wave
107 else
108 {
109     if ( waveIndex % 5 == 0 ) // is waveIndex divisible by 5
110     {
111         SpawnSquad(bossPrefab, waveIndex / 5);
112     }
113     else
114     {
115         int unitPoints = waveIndex; // set amount of ships to spawn, one each wave so far
116         while (unitPoints > 0)
117         {
118             int randomType = Random.Range(1, 4); // randomly select type of squad to spawn
119             int randomCount = Random.Range(1, unitPoints + 1); // randomly select how many ships in that squad,
120             unitPoints -= randomCount; // subtract points used from point count
121             switch (randomType)
122             {
123                 case 1:
124                     SpawnSquad(standardPrefab, randomCount);
125                     break;
126                 case 2:
127                     SpawnSquad(lightPrefab, randomCount);
128                     break;
129                 case 3:
130                     SpawnSquad(heavyPrefab, randomCount);
131                     break;
132             }
133         }
134     }

```

The script spawns a random amount of squads with a total amount of ships equal to the wave number

- Added random wave generator, so after wave 5 every wave is randomly generated
- Downloaded DinV Studio's Dynamic Space Background to use in a skybox as a background for the game
- ♦ **Found player's lasers weren't dealing damage to enemies, but enemies lasers worked fine**
- ♦ **Found mouse movement combined with wasd movement to move twice as fast**
- ♦ **Rewrote movement code so both controls affected the same movement value, instead of both moving the ship separately**

```

void Update()
{
    if (Input.GetAxisRaw("Roll") * Input.GetAxisRaw("Roll") == 1)
    {
        roll = Input.GetAxis("Roll");
    }
    else if (Input.GetAxis("SmoothRoll") * Input.GetAxis("SmoothRoll") >= 1)
    {
        roll = Mathf.Sign(Input.GetAxis("SmoothRoll"));
    }
    else
    {
        roll = Input.GetAxis("SmoothRoll");
    }

    if (Input.GetAxisRaw("Pitch") * Input.GetAxisRaw("Pitch") == 1)
    {
        pitch = Input.GetAxis("Pitch");
    }
    else if (Input.GetAxis("SmoothPitch") * Input.GetAxis("SmoothPitch") >= 1)
    {
        pitch = Mathf.Sign(Input.GetAxis("SmoothPitch"));
    }
    else
    {
        pitch = Input.GetAxis("SmoothPitch");
    }

    transform.Rotate(pitchSpeed * pitch * Time.deltaTime,
        yawSpeed * Input.GetAxis("Yaw") * Time.deltaTime,
        rollSpeed * roll * Time.deltaTime);
}

```

Reworked movement script preventing moving with keyboard

9/08/20

- Applied ship models
- Scaled hitboxes to fit models and relocated weapon positions
- Downloaded Pulsar Bytes' "Starfield Skybox" as background for game
- Applied space skybox



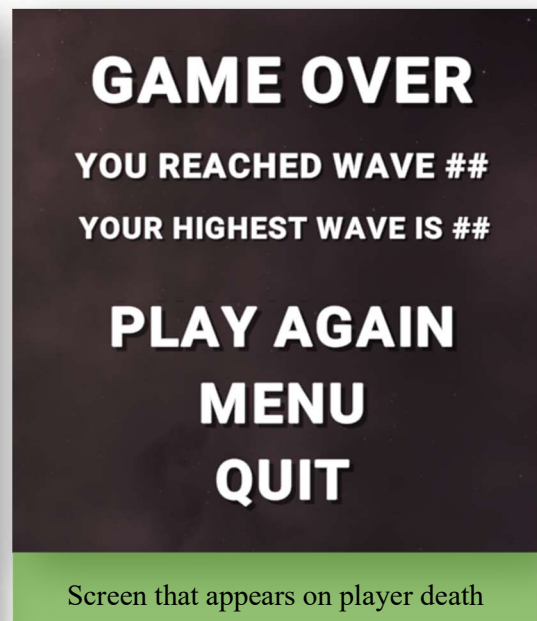
The different ship models and textures. From left to right: player, standard, heavy, light, boss

11/08/20

- Added save function, to save high score and settings
- Wrote start-up script that will update settings if there are any settings saved
- Began game over menu
- Found that the cursor doesn't disappear when playing game

```
12 void Start()
13 {
14     om = gameObject.GetComponentInChildren<OptionsMenu>();
15
16     if (PlayerPrefs.HasKey("volume"))
17     {
18         om.SetVolume(PlayerPrefs.GetFloat("volume"));
19     }
20     if (PlayerPrefs.HasKey("qualityIndex"))
21     {
22         om.SetQuality(PlayerPrefs.GetInt("qualityIndex"));
23     }
24     if (PlayerPrefs.HasKey("fullscreenInt"))
25     {
26         if (PlayerPrefs.GetInt("qualityIndex") == 1)
27         {
28             isFullscreen = true;
29         }
30         else
31         {
32             isFullscreen = false;
33         }
34         om.SetFullscreen(isFullscreen);
35     }
36     if (PlayerPrefs.HasKey("resolutionIndex"))
37     {
38         om.SetQuality(PlayerPrefs.GetInt("qualityIndex"));
39     }
40 }
```

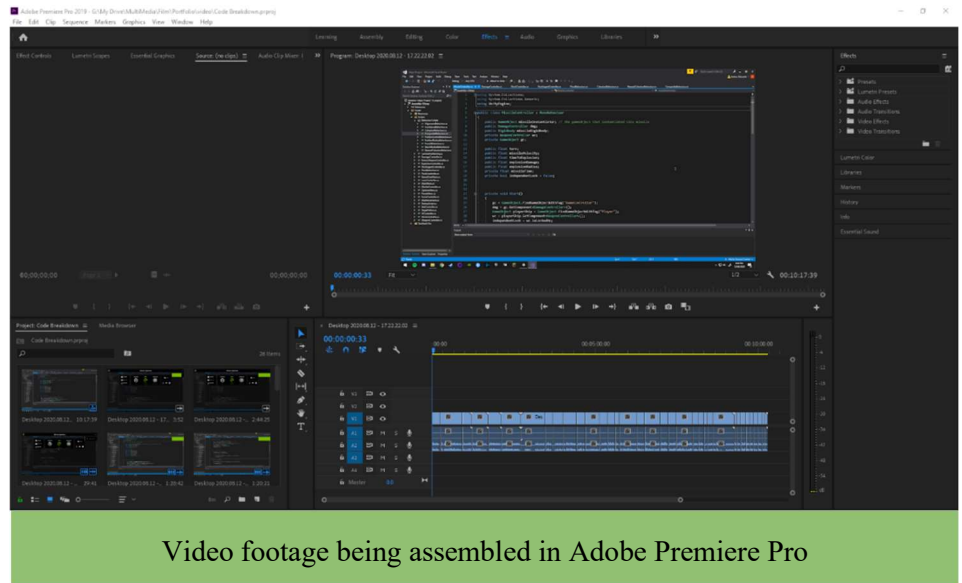
Script that runs on start-up that updates settings to what they were set to last time



IT Multimedia – Spaceships!

12/08/20

- Recorded the code breakdown script
- Assembled footage into video using Adobe Premiere Pro



15/08/20

- Fixed lasers by adding RigidBody to enemy ships
- Added explosion prefab to game

18/08/20

- Made cursor invisible while in-game
- Trimmed code breakdown footage from 10 minutes to 8 and a half
- ♦ Need to get video to 6 minutes or under

20/08/20

- ♦ Cut out damage controller script segment of video, removed 2 and a half minutes, need to cut another 13 seconds
- ♦ Increased video speed by 4% to shorten video by 13 seconds
- Finished video

22/08/20

- Learned how to add audio to Unity
- Inserted various sound effects into game
- Used Willewis' "Musket Explosion" audio clip: <https://freesound.org/people/Willewis/sounds/244345/>
- Used Aiwha's "Explosion" audio clip: <https://freesound.org/people/Aiwha/sounds/250712/>
- Used primeval_polypod's "Rocket Launch" audio clip: https://freesound.org/people/primeval_polypod/sounds/158894/
- Used PlymouthJCliffords' "Rocket Roar (looping)" audio clip: <https://freesound.org/people/PlymouthJCliffords/sounds/164842/>
- Used LimitSnap_Creations' "Rocket Thrust effect" audio clip: https://freesound.org/people/LimitSnap_Creations/sounds/318688/
- Used Lambich's "Small_Pulse_Cannon_Fire" audio clip: <https://freesound.org/people/Lambich/sounds/350579/>
- Set sound effects to occur in 3D space, adjusting volume according to direction and distance
- Added section to UIController that updates the player's thruster audio to match its strength

23/08/20

- Modified main menu to include game title
- Removed blue background so space skybox was visible
- Created camera rotator script that would slowly rotate the menu background
- **Found weapons could be fired while in pause menu**
- ◆ **Added line in WeaponController script that checks if the game is paused before you can fire a weapon**



```
40 void Update()
41 {
42     // Fire Lasers
43     nextLaserFire += Time.deltaTime;
44     if (Input.GetButton("PrimaryFire") && nextLaserFire >= laserFireRate && PauseMenu.GameIsPaused != true)
45     {
46         if (shotIsLeft == true)
47         {
48             laserInstance = Instantiate(laserProjectile, leftLaserGun.transform.position, transform.rotation);
49             laserInstance.GetComponent<LaserController>().laserInstantiator = gameObject;
50             shotIsLeft = false;
51         }
52         else
53         {
54             laserInstance = Instantiate(laserProjectile, rightLaserGun.transform.position, transform.rotation);
55             laserInstance.GetComponent<LaserController>().laserInstantiator = gameObject;
56             shotIsLeft = true;
57         }
58         nextLaserFire = 0f;
59     }
60     // Fire Missiles
61     nextMissileFire += Time.deltaTime;
62     if (Input.GetButtonDown("SecondaryFire") && nextMissileFire >= missileFireRate && PauseMenu.GameIsPaused != true)
63     {
64         missileInstance = Instantiate(missile, missileLauncher.transform.position, missileLauncher.transform.rotation);
65         missileInstance.GetComponent<MissileController>().missileInstantiator = gameObject;
66         nextMissileFire = 0f;
67     }
68 }
```

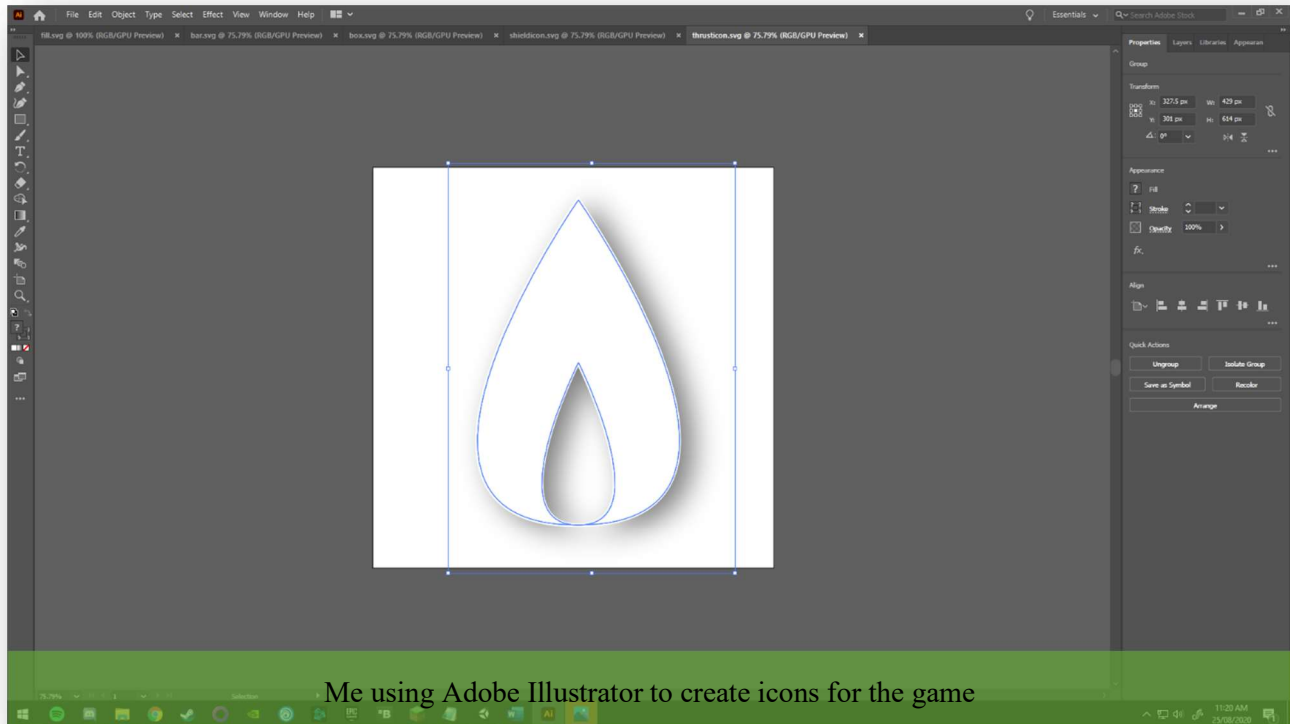
WeaponController script

```
5 public class MenuCamRotator : MonoBehaviour
6 {
7     Quaternion lastRotation;
8     Quaternion nextRotation;
9     float t;
10    public float secsPerRotation;
11    // Start is called before the first frame update
12    void Start()
13    {
14        lastRotation = Random.rotation;
15        nextRotation = Random.rotation;
16        gameObject.transform.rotation = lastRotation;
17    }
18
19    // Update is called once per frame
20    void Update()
21    {
22        t += Time.deltaTime * 1/secsPerRotation;
23        gameObject.transform.rotation = Quaternion.SlerpUnclamped(lastRotation, nextRotation, t);
24    }
25 }
```

MenuCamRotator script

24/08/20

- Redesigned HUD to match art style better
- Used Adobe Illustrator to create two icons for thrust and shield, and to create a background and fill bar for the sliders



- ♦ **Unity is not designed to handle vector images**
- ♦ **Added an extension that would allow me to use .svg files**
- ♦ **Unity extension was in early stages of development and was too unstable to use, kept causing images to break and crashes**
- ♦ **Converted .svg files into .png files, which could easily be implemented into the game as sprites without the use of the extension**

25/08/20

- Drafted final evaluation

26/08/20

- Finished final evaluation
- Proof read portfolio
- Updated page numbers
- Moved around, formatted, added, and sourced images
- Grammar checked portfolio

REFERENCES

Each of the references are dated, the dates correlate to the process diary so to see what I used the reference for, cross reference the dates with the process diary.

- YouTube. 2020. Let's Learn C++: Basic Tutorials - YouTube. [ONLINE] Available at: https://www.youtube.com/playlist?list=PL82YdDfxhWsCyZLsg_kXhH8sy5ixQNras [Accessed 8 September 2019].
- C# Station. 2020. Understanding the Differences Between C#, C++, and C - C# Station. [ONLINE] Available at: <https://csharp-station.com/understanding-the-differences-between-c-c-and-c/> [Accessed 15 September 2019].
- Game Training Pty Lt. 2020. Student Masterclass | Game Training Pty Lt. [ONLINE] Available at: <https://www.nsw.gamettraining.com.au/the-unity-masterclass>. [Accessed 28 September 2019].
- ACE Manual: ACE 9016 Submitted works and practical examinations: health and safety issues. 2020. ACE Manual: ACE 9016 Submitted works and practical examinations: health and safety issues. [ONLINE] Available at: <https://ace.nesa.nsw.edu.au/ace-9016> [Accessed 6 November 2019].
- Practical and performance exams | NSW Education Standards. 2020. Practical and performance exams | NSW Education Standards. [ONLINE] Available at: <https://educationstandards.nsw.edu.au/wps/portal/nesa/11-12/hsc/rules-and-processes/practical-performance-exams> [Accessed 6 November 2019].
- Unity Connect. 2020. Getting Started with VR - Unity Learn. [ONLINE] Available at: <https://learn.unity.com/tutorial/getting-started-with-vr#5c7f8528edbc2a002053b447>. [Accessed 8 November 2019].
- thetuvix. 2020. Unity development overview - Mixed Reality | Microsoft Docs. [ONLINE] Available at: <https://docs.microsoft.com/en-us/windows/mixed-reality/unity-development-overview> [Accessed 8 November 2019].
- cre8ivepark. 2020. Getting started with MRTK version 2 - Mixed Reality | Microsoft Docs. [ONLINE] Available at: <https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-getting-started> [Accessed 8 November 2019].
- thetuvix. 2020. Configure a New Unity Project for Windows Mixed Reality - Mixed Reality | Microsoft Docs. [ONLINE] Available at: <https://docs.microsoft.com/en-us/windows/mixed-reality/configure-unity-project> [Accessed 8 November 2019].
- Getting started with MRTK | Mixed Reality Toolkit Documentation . 2020. Getting started with MRTK | Mixed Reality Toolkit Documentation . [ONLINE] Available at: <https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/GettingStartedWithTheMRTK.html> [Accessed 8 November 2019].
- What is the Mixed Reality Toolkit | Mixed Reality Toolkit Documentation . 2020. What is the Mixed Reality Toolkit | Mixed Reality Toolkit Documentation . [ONLINE] Available at: <https://microsoft.github.io/MixedRealityToolkit-Unity/README.html> [Accessed 8 November 2019].
- GitHub. 2020. GitHub - microsoft/MixedRealityToolkit-Unity: Mixed Reality Toolkit (MRTK) provides a set of components and features to accelerate cross-platform MR app development in

- Unity. [ONLINE] Available at: <https://github.com/microsoft/MixedRealityToolkit-Unity>. [Accessed 8 November 2019].
- Unity Connect. 2020. Intro & Installing VRTK - Unity Learn. [ONLINE] Available at: <https://learn.unity.com/tutorial/installing-vrtk-beta?projectId=5cdc9e4cedbc2a1ef423426c#5ce5d45aedbc2a1da065e45d> [Accessed 8 November 2019].
 - What can I copy/communicate- . 2020. What can I copy/communicate- . [ONLINE] Available at: <https://www.smartcopying.edu.au/copyright-guidelines/what-can-i-copy-communicate-> [Accessed 11 November 2019].
 - Game Development Envato Tuts+. 2020. The Four Elements of Game Design - Envato Tuts+ Game Development Tutorials. [ONLINE] Available at: <https://gamedevelopment.tutsplus.com/series/the-four-elements-of-game-design--cms-1117> [Accessed 19 November 2019].
 - Digitizing audio. 2020. Digitizing audio. [ONLINE] Available at: <https://helpx.adobe.com/au/audition/using/digitizing-audio.html> [Accessed 19 November 2019].
 - Digitizing Sound. 2020. Digitizing Sound. [ONLINE] Available at: <https://cs.calvin.edu/activities/books/rit/chapter5/easy/sound.html> [Accessed 19 November 2019].
 - paaron301 . 2020. Windows Mixed Reality minimum PC hardware compatibility guidelines - Enthusiast Guide | Microsoft Docs. [ONLINE] Available at: <https://docs.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/windows-mixed-reality-minimum-pc-hardware-compatibility-guidelines> [Accessed 7 January 2020].
 - YouTube. 2020. Unity Camera Follow Player Tutorial (Free Assets & Scripts) - YouTube. [ONLINE] Available at: <https://www.youtube.com/watch?v=urNrY7FgMao> [Accessed 20 February 2020].
 - YouTube. 2020. Unity 3D Platformer - Learn to Make a 3D Action Platformer! - YouTube. [ONLINE] Available at: https://www.youtube.com/playlist?list=PLiyfvmtjWC_V_H-VMGGAzi7n5E0gyhc37 [Accessed 20 February 2020].
 - YouTube. 2020. C# Coding In Unity: Accessing Scripts/Variables from other Scripts(Game Objects) 2017.2. [ONLINE] Available at: <https://www.youtube.com/watch?v=XeOzp6KJlJ8> [Accessed 14 March 2020].
 - YouTube. 2020. 5 ways to Reference GameObjects in Unity3D – YouTube. [ONLINE] Available at: <https://www.youtube.com/watch?v=ymq2AUckws0>. [Accessed 25 March 2020].
 - YouTube. 2020. HOW TO MAKE HEART/HEALTH SYSTEM - UNITY TUTORIAL - YouTube. [ONLINE] Available at: <https://www.youtube.com/watch?v=3uyolYVsiWc>. [Accessed 14 May 2020].
 - YouTube. 2020. Flocking Algorithm in Unity - YouTube. [ONLINE] Available at: <https://www.youtube.com/playlist?list=PL5KbKbJ6Gf99UlyIqzV1UpOzseyRn5H1d>. [Accessed 6 July 2020].
 - YouTube. 2020. START MENU in Unity - YouTube. [ONLINE] Available at: https://www.youtube.com/watch?v=zc8ac_qUXQY. [Accessed 11 July 2020].
 - YouTube. 2020. SETTINGS MENU in Unity - YouTube. [ONLINE] Available at: <https://www.youtube.com/watch?v=YOaYQrN1oYQ>. [Accessed 12 July 2020].

- YouTube. 2020. PAUSE MENU in Unity - YouTube. [ONLINE] Available at: <https://www.youtube.com/watch?v=JivuXdrIHK0>. [Accessed 12 July 2020].
- YouTube. 2020. Introduction to AUDIO in Unity - YouTube. [ONLINE] Available at: <https://www.youtube.com/watch?v=6OT43pvUyfY>. [Accessed 13 July 2020].
- CGTrader. 2020. Star Sparrow Modular Spaceship 3D asset | CGTrader. [ONLINE] Available at: <https://www.cgtrader.com/free-3d-models/space/spaceship/star-sparrow-modular-spaceship>. [Accessed 14 July 2020].
- Unity Asset Store. 2020. Starfield Skybox | 2D Sky | Unity Asset Store. [ONLINE] Available at: <https://assetstore.unity.com/packages/2d/textures-materials/sky/starfield-skybox-92717>. [Accessed 9 August 2020].
- Unity Asset Store. 2020. Milky Way Skybox | 2D Textures & Materials | Unity Asset Store. [ONLINE] Available at: <https://assetstore.unity.com/packages/2d/textures-materials/milky-way-skybox-94001>. [Accessed 9 August 2020].
- Unity Asset Store. 2020. Dynamic Space Background Lite | 2D Textures & Materials | Unity Asset Store. [ONLINE] Available at: <https://assetstore.unity.com/packages/2d/textures-materials/dynamic-space-background-lite-104606>. [Accessed 9 August 2020].
- YouTube. 2020. Unity - How To Save Game / Data [PlayerPrefs] - YouTube. [ONLINE] Available at: <https://www.youtube.com/watch?v=y7RPVvwjrsA>. [Accessed 11 August 2020].
- Unity Asset Store. 2020. Unity Particle Pack 5.x | Asset Packs | Unity Asset Store. [ONLINE] Available at: <https://assetstore.unity.com/packages/essentials/asset-packs/unity-particle-pack-5-x-73777>. [Accessed 12 August 2020].
- Unity Asset Store. 2020. Fx Explosion Pack | Fire & Explosions | Unity Asset Store. [ONLINE] Available at: <https://assetstore.unity.com/packages/vfx/particles/fire-explosions/fx-explosion-pack-30102>. [Accessed 12 August 2020].
- YouTube. 2020. Introduction to AUDIO in Unity - YouTube. [ONLINE] Available at: <https://www.youtube.com/watch?v=6OT43pvUyfY>. [Accessed 26 August 2020].
- Freesound. 2020. Freesound - "Musket Explosion" by Willlewis. [ONLINE] Available at: <https://freesound.org/people/Willlewis/sounds/244345/>. [Accessed 26 August 2020].
- Freesound. 2020. Freesound - "Explosion" by Aiwha. [ONLINE] Available at: <https://freesound.org/people/Aiwha/sounds/250712/>. [Accessed 26 August 2020].
- Freesound. 2020. Freesound - "Rocket Launch" by primeval_polypod. [ONLINE] Available at: https://freesound.org/people/primeval_polypod/sounds/158894/. [Accessed 26 August 2020].
- Freesound. 2020. Freesound - "Rocket Roar (looping)" by PlymouthJCliffords. [ONLINE] Available at: <https://freesound.org/people/PlymouthJCliffords/sounds/164842/>. [Accessed 26 August 2020].
- Freesound. 2020. Freesound - "Rocket Thrust effect.wav" by LimitSnap_Creations. [ONLINE] Available at: https://freesound.org/people/LimitSnap_Creations/sounds/318688/. [Accessed 26 August 2020].
- Freesound. 2020. Freesound - "Small_Pulse_Cannon_Fire.wav" by Lambich. [ONLINE] Available at: <https://freesound.org/people/Lambich/sounds/350579/>. [Accessed 26 August 2020].

FINAL EVALUATION

The final product for my Year 12 Multimedia major project is a complete game I created through Unity.

In terms of the goals I set out in my statement of intent, to create a fun game with intuitive controls and complex AI or procedural generation, I was, for the most part, successful. While the game didn't turn out to be all that much fun, it had all the aspects a working game should have, including simple controls and functioning AI. In the end it turned out to be more like a tech demo, displaying all the core aspects that could be in a game without the volume of content a marketable game might have. Although what I accomplished was satisfactory, with more time I would have liked to expand on the game to give a greater sense of progression, and to include more content to give the game greater depth.

My prior knowledge of programming and video games were extremely helpful, as they shortened the learning curve required for Unity and C#. My interest in video games was also a great source of motivation. I was able to extract elements of other games and implement them in my own, such as the colour coding from Galaxian, and the intense space combat of Elite Dangerous.

My limited resources (time, manpower, and software) turned out to be sufficient due to the way I designed the game. I planned the game out in a way that minimised the need for artistic elements such as modelling and graphic design, and emphasized the need for the game's mechanics, such as AI. I originally wanted to include some procedural generation, but the game idea I ended up developing didn't suit the concept and had no reason for me to add it, so I left it out. The AI turned out very well, the enemies were able to interact with each other, travel in formation, and evade, pursue, and attack the player.

The greatest shortfall of the game was the audio. I initially thought that sourcing the audio wouldn't require much time, and that creating my own audio would have been unnecessary, thus I didn't allow myself sufficient time to work on it. As a result I was only able to acquire low quality audio and no sound track, and while it was sufficient for the games purpose, I realised late in development that there was a lot of room for improvement in the audio and I could have demonstrated more skills had I allowed time to develop audio myself. I would have liked to record some sound effects myself, acquire higher quality audio for anything I couldn't create myself, and personally write and record a soundtrack for the game. I also have an interest in music and can play the piano and guitar reasonably well, my experience writing music would have enabled me to create a soundtrack for the game given enough time. Writing some music for the game would have been an interesting and fun challenge that would have contributed to the intensity and overall experience of the game.

There were other things that I didn't consider, such as the need for knowledge of physics and mathematics in game development. As I developed the game I found myself needing to research certain topics that I hadn't expected. When I was creating the ship movement scripts, I learned that unity used a vector system to track the location, rotation, and movement of game objects. I had to learn how vectors worked and how I could use them in my game. Then I found out that object rotations are managed with not only vectors, but another type of number system called a quaternion, which measured rotations around axes x, y, z, and w. I had to learn about gimbal locks, and why it

was necessary for the introduction of a fourth axis to define a three-dimensional object, and how to apply that knowledge in my game. These unexpected topics required massive amounts of additional research, I found myself learning about not only about maths but about different coding elements, like abstract classes, about Unity features, like scriptable objects, and about various other little things, like vector graphic illustration.

The game turned out as well as I had hoped, there were areas for improvement but I've learned a lot about game development and I would be able to do a much better job if I tried again

Aside from the actual game development, I gained a lot of valuable experience in project management. There were three main areas I managed, time management, finance, and risk control.

In terms of time management, I did a good job planning out the sequence of development processes, and scheduling each step. As I was faced with certain challenges, I was able to replan my schedule to make up for lost time. The COVID-19 lockdown was the biggest challenge, I was unable to work efficiently as I had limited access to teacher's support, and because I had to share my computer with my family which greatly reduced the amount of time I could spend working on my major project. During lockdown worked to the best of my ability but found myself falling behind, but once lockdown was released, I was able to get back on schedule by outsourcing certain less important aspects of game development, such as the models and textures for the ship.

My finance management was less successful, although it didn't cause any issues for my game development. I had originally not planned on spending any money, as I was using only free software and hardware that I already owned, and I was developing the game independently. As I was working on my project however, I came across a few unexpected costs, such as a Bluetooth adapter and my multimedia fees. Fortunately, these costs were minor, so they didn't hinder my progress.

The final area of project management was risk management; thanks to my risk control I was able to develop my game safely. The dangers in an office place aren't often life threatening, but things like improper ergonomics and muscle strain can cause long term health issues. Because I identified the risks and applied controls, I was able to effectively prevent any such injuries.

Overall, my project management was successful, and the few mistakes I made will serve as learning experiences for future project management