University of Plymouth

School of Engineering, Computing and Mathematics



COMP3000

Final Stage Computing Project

2021/2022

*(Defend & Capture)*

James Yarnold

*(10621871)*

BSc (Hons) Computing & Games Development

# Acknowledgements

I would like to thank friends and family; especially Leah Humphries for always being there for me through each year of this course and my parents at home who always provided me with support and guidance.

Thank you to my friends who helped test my game and gave me helpful constructive criticism.

I would also like to thank Mr James Hayter and Mr Paul Watson for their efforts on teaching Unity and project semantics thought out the key stages of the course.

# Abstract

This report will address the aspects of planning and project development around the game Defend and Capture. For full context the game was created during the final year of university and represents the knowledge of four years of education. The game is a small scale RTS genre based around the toy soldier’s aesthetic and aimed and newer players to the genre, it features a combat system and base building mechanic, combined with AI systems that help bring the game to life.

The report starts of by outlining the core concepts that started the journey of project management, beginning with project objectives from the perspective of the author and then the objectives of the game itself, as well as defining the methods of approach toa chieve these goals.

The body talks about the project sprints and what was notable features were added or experienced in each one. The project cycle consists of seven sprints that were approximately two weeks to a month long.

It concludes with a reflection of the project, which attempts to highlight some issues with development but also reflect on where it succeeded.

Table of Contents

[1 Acknowledgements 2](#_Toc103585850)

[2 Abstract 2](#_Toc103585851)

[4 Table of Figures 4](#_Toc103585852)

[5 Word Count 5](#_Toc103585853)

[6 Links 5](#_Toc103585854)

[6.1 GitHub 5](#_Toc103585855)

[6.2 Itch.io 5](#_Toc103585856)

[6.3 teams 5](#_Toc103585857)

[7 Introduction 5](#_Toc103585858)

[7.1 Project Definition 5](#_Toc103585859)

[7.2 Purpose 6](#_Toc103585860)

[7.3 Objectives & Functional Requirements 6](#_Toc103585861)

[7.3.1 Minimum viable product 6](#_Toc103585862)

[7.3.2 Minimum awesome product 7](#_Toc103585863)

[8 Method of Approach 7](#_Toc103585864)

[8.1 Project Management 9](#_Toc103585865)

[8.1.1 Microsoft Teams 9](#_Toc103585866)

[8.1.2 Testing 10](#_Toc103585867)

[8.1.3 Version Control 10](#_Toc103585868)

[8.2 Technologies 11](#_Toc103585869)

[8.2.1 Visual Studio, Unity and C# 11](#_Toc103585870)

[8.2.2 Navigation mesh 11](#_Toc103585871)

[8.2.3 Ray casts 12](#_Toc103585872)

[9 Legal, Social, Ethical and Professional 12](#_Toc103585873)

[9.1 Legal 12](#_Toc103585874)

[9.2 Social 13](#_Toc103585875)

[9.3 Ethics 13](#_Toc103585876)

[9.4 Professional 13](#_Toc103585877)

[10 Implementation 14](#_Toc103585878)

[10.1 Sprint 0: 26/11/2021 – 10/12/2021 14](#_Toc103585879)

[10.1.1 Overview 14](#_Toc103585880)

[10.2 Sprint 1: 10/12/2021 – 26/01/2022 16](#_Toc103585881)

[10.2.1 Overview 16](#_Toc103585882)

[10.2.2 Tasks 16](#_Toc103585883)

[10.2.3 Implementation 17](#_Toc103585884)

[10.3 Sprint 2: 26/01/2022 – 22/02/2022 18](#_Toc103585885)

[10.3.1 Overview 18](#_Toc103585886)

[10.3.2 Tasks 19](#_Toc103585887)

[10.3.3 Implementation 19](#_Toc103585888)

[10.4 Sprint 3: 22/02/2022 – 17/03/2022 21](#_Toc103585889)

[10.4.1 Overview 21](#_Toc103585890)

[10.4.2 Tasks 21](#_Toc103585891)

[10.4.3 Implementation 21](#_Toc103585892)

[10.5 Sprint 4: 17/03/2022 – 05/04/2022 23](#_Toc103585893)

[10.5.1 Overview 23](#_Toc103585894)

[10.5.2 Tasks 24](#_Toc103585895)

[10.5.3 Implementation 24](#_Toc103585896)

[10.6 Sprint 5: 05/04/2022 – 19/04/2022 28](#_Toc103585897)

[10.6.1 Overview 28](#_Toc103585898)

[10.6.2 Tasks 29](#_Toc103585899)

[10.6.3 Implementation 29](#_Toc103585900)

[10.7 Sprint 6: 19/04/2022 – 03/05/2022 33](#_Toc103585901)

[10.7.1 Overview 33](#_Toc103585902)

[10.7.2 Tasks 33](#_Toc103585903)

[10.7.3 Implementation 33](#_Toc103585904)

[10.8 Sprint 7: 03/05/2022+ 35](#_Toc103585905)

[10.8.1 Overview 35](#_Toc103585906)

[10.8.2 Tasks 35](#_Toc103585907)

[10.8.3 Implementation 35](#_Toc103585908)

[11 End of Project post-mortem 36](#_Toc103585909)

[12 Conclusion 36](#_Toc103585910)

[13 Bibliography 36](#_Toc103585911)

[14 Appendices 37](#_Toc103585912)

[14.1 Appendix 1: Games Design Document 37](#_Toc103585913)

[14.1.1 Game Design Document – Defend & Capture 38](#_Toc103585914)

[14.1.2 Assets 42](#_Toc103585915)

[14.1.3 Unity Store Assets 43](#_Toc103585916)

[14.2 Appendix 2: Sprints and Retrospective tasks 46](#_Toc103585917)

[14.3 Appendix 3: Scrapped UI designs 56](#_Toc103585918)

[14.4 Appendix 4: questions and results 57](#_Toc103585919)

# Table of Figures

[Figure 1, showing my Kanban style board during sprint 7 8](#_Toc103585682)

[Figure 2, showing my Sprint Retrospective at the end of sprint 7 9](#_Toc103585683)

[Figure 3 visualises the reason why convex shapes are used in baking unity navigation meshes. 11](#_Toc103585684)

[Figure 4 documentation created to plan the system of building and upgrades 15](#_Toc103585685)

[Figure 5 blender scene of base and buildings 16](#_Toc103585686)

[Figure 6 blender scene of soldiers and guns 16](#_Toc103585687)

[Figure 7 code snippet of camera controls 17](#_Toc103585688)

[Figure 8 Current Menu Design 18](#_Toc103585689)

[Figure 9 Soldier Prefab 20](#_Toc103585690)

[Figure 10 Selection Logic 20](#_Toc103585691)

[Figure 11 Units exiting base after spawning 22](#_Toc103585692)

[Figure 12 logic of selection sphere 23](#_Toc103585693)

[Figure 13 logic of selection sphere 24](#_Toc103585694)

[Figure 14 ray casts visualised 25](#_Toc103585695)

[Figure 15 Unity inspector showing hoe the relocate targets were applied 26](#_Toc103585696)

[Figure 16 unity hierarchy showing how the relocate targets are children of the main unit 26](#_Toc103585697)

[Figure 17 Showing the distance of one of the relocate targes from the main unit 27](#_Toc103585698)

[Figure 18 code that uses ray casts to see if the unit can see an enemy or not 27](#_Toc103585699)

[Figure 19 Relocate logic 28](#_Toc103585700)

[Figure 20 Blender models Check list 29](#_Toc103585701)

[Figure 21 Tanks rotate angle located first raycast 30](#_Toc103585702)

[Figure 22 while the second ray cast activates the helicopter and soldiers rotate angle logic 30](#_Toc103585703)

[Figure 23 Rocket projectile and tag 31](#_Toc103585704)

[Figure 24 Missile projectile and tag 31](#_Toc103585705)

[Figure 25 bullet projectile and tag 31](#_Toc103585706)

[Figure 26 health and damage for soldier 32](#_Toc103585707)

[Figure 27 code for making enemy AI move into range of player unit if another Enemy in range spotted it 34](#_Toc103585708)

[Figure 28red buttons for useability 34](#_Toc103585709)

# Word Count

# Links

## GitHub

<https://github.com/Dracknid10/Defend-Capture>

## Itch.io

<https://theletterpurple.itch.io/defend-capture>

## teams

<https://teams.microsoft.com/l/team/19%3aecKoxcoIuq21EdUilBes13uVypzTjXk8I-vHWK0CcB81%40thread.tacv2/conversations?groupId=efa916e3-b129-4e4c-b67e-9bd5b6138c05&tenantId=5437e7eb-83fb-4d1a-bfd3-bb247e061bf1>

# Introduction

This report details my experience in taking on a large solo project that attempts to showcase my skills and understanding of Unity’s libraries as well as incorporating many of the aspects that were taught to me over the duration of the course. Specifically, I drew inspiration from COMP2007 where I built a 3D game and 3D scene for the first time. I was also inspired by my experience with COMP3013 where I was project manager. During this time, I spent more time with project development strategies and became more familiar with how to operate a large-scale project from a planning level. I intended to utilise the experience I have gained in these modules, to plan and enact a project of appropriate size and scale that can be used as an important stand out portfolio piece, which includes hall marks of industry standard criteria that employers like to see. With significant focus on Unity and partner software with Unity such as Blender, I hoped to link this to the current COMP3014 module to achieve this portfolio piece.

## Project Definition

Defend & Capture is a 3D single player, artificial intelligence driven, faster paced, real time strategy game with a casual simulation twist on the combat. It is built on the Unity game engine, specifically, using Unity’s navmesh AI system, raycast networks and lightweight custom built sorting algorithms. The game features custom-built assets using Blender and free assets imported from the Unity asset store that creates an overarching ‘toy soldiers’ aesthetic to the game. This reflects a casual theme that aims to welcomes a younger audience to the RTS; several RTS focused features and several important AI mechanics designed for engaging gameplay. The gameplay loop is designed around a ‘rock paper scissors’-based unit creation where unit types of infantries, tanks and helicopters counter each other. The player must manage their base supplies and buildings while trying to stay on top of the unit counters the enemy AI is creating. The main objective for the player is to get to the AI base and destroy it to complete the game while protecting theirs.

## Purpose

The purpose of the game was to provide a creative platform for me to push my skills and knowledge of Unity, in a way that’s challenging and introduces new concepts to my skill base to ultimately show to employers. On a gameplay level, Defend and capture hopes to stretch these skills in trying to produce a valuable and comprehensive AI using Unity’s libraries. Making the AI challenging and synergistic was an important aim to make the journey of playing an experience centred around determination so players can feel they did well even if they lost, resulting in rewarding gameplay.

## Objectives & Functional Requirements

Upon project conception, the functional requirements were established to give a good foundation to the backlog and actualise concepts into preliminary planning stages. The objectives that aligned with my existential goals were:

* Use a ‘Kanban’ style of framework for project management that parallels with key agile project management manifesto core values.
* Deliver an error/ bug free experience to promote this in future works.
* Explore the ways of production of a custom-built AI in the Unity game engine
* Use the knowledge of previous modules to combine a gaming experience worthy of a portfolio showcase.
* To utilise testing to deliver a better project based on feedback

Core manifesto values reinforce an idea of not getting bogged down in extensive and exhaustive documentation but streamlining these into user stories. This will begin the project sooner and without delay. The manifesto also stresses the use of ‘customer collaboration’ which brings the potential customers into the development process, to give developers useful insight as its being built and then accepting that changes might be helpful to the project’s solution. (Wrike, n.d.).

My functional requirements were split into my minimum viable product (MVP) and my Maximum Awesome Product (MAP)

### Minimum viable product

* Player can build buildings and each of them effect the game in their specific way.
* Players can control troops using the single click system.
* Player units and enemy units fight and can hurt each other.
* Player can destroy the enemy base and win, and Players can have their base destroyed and lose (Conquest).
* Enemy Ai moves around the map randomly.
* Game has no errors and functions on mid to low spec PCs.

### Minimum awesome product

* Players can control the troops using single click, group select, select all of one type of units and select all units.
* Sounds for all different aspects.
* Player can upgrade existing units using the building upgrades system.
* Units fight each other and rock paper scissors system is completely integrated.
  + Flanking troops (if a troop is shot in the back) they take extra damage.
* Players can decide which units fight which unit by clicking on the unit then the enemy unit.
* Enemy AI uses tactics to fight for example, flanking and counter flanking, distractions, counter units, rushes/ turtling, scouting, directly fighting troops weak to the unit, or holding strategic areas of the map.
* More than one map.
* More than one game type, conquest, and dominion (hold areas to gain points, first to x points wins).

Full context of these can be seen in the games design document at [appendix one](#_Appendix_1Games_Design)

# Method of Approach

The main goal with project management was to display a level of knowledge around agile project management that would be satisfactory or exceptional to the level of industry standards. To adhere to an Agile strategy, the project would be iteratively assessed and dynamically adapted to achieve my functionality targets, with the addition of deliverable retrospectives on a biweekly or triweekly basis depending on task sizes, and importance of depth and discovery of problematic errors. These ‘Sprints’ would dictate the flow of progress and task lists for that section. Adjusting the length of them dynamically was important, given some tasks of the project were bigger and more important to complete than others but limiting it to a week extra meant tasks weren’t overextending into areas out of scope and a refocus was possible upon the sprint ending.

To help with the flow of tasks, a Kanban style board was used to keep track of large “parent” tasks which were defined in user stories. By splitting these down into smaller tasks, meant that could be picked and chosen as and when they were necessary for development. They formed the basis of checklists that reflected the MVP and MAP goals. Sprints would be followed with a retrospective that would portray what was achieved in the sprint, and what problems arose. The tasks for the next sprint could be assessed using the information of the last sprint and new tasks could be assigned more accurately with the context of the qualitative information presented in past sprints retrospective. This helped the natural flow between the end of one sprint and the start of another.

This was shown to help during the process of deciding and pulling tasks from the backlog or the “to do” into the “doing” column where I would decide on high priority task’s I believe I could get done in the two to three weeks allotted for sprints. During this, uncompleted tasks would be identified in the retrospective, to be given extra thought on when moving to the next sprint. Tasks would be either sent back to the back log or would stay in the task list if priority was important enough.

The board, featured in figure one, and the retrospective, featured in figure two, acted codependently on gaining insight when choosing tasks for the next sprint. The board provided an over arcing title of larger tasks and their break downs, while the retrospective provided an understanding on the progress of smaller tasks and if they needed more in-depth strategies applied to them. In the sprint report these over arcing tasks from the board are in green, and errors or added tasks during the sprint are in red. These tasks and errors were ticked and checked off on the board if they were completed when the sprint ends, if they weren’t, the task was moved down into the next task list and the parent task stayed in the “doing” column. Sprints and their tasks, as they are on teams, can be seen at [appendix two](#_Appendix_2:_Sprints) as well as checklists for tasks in the same area.

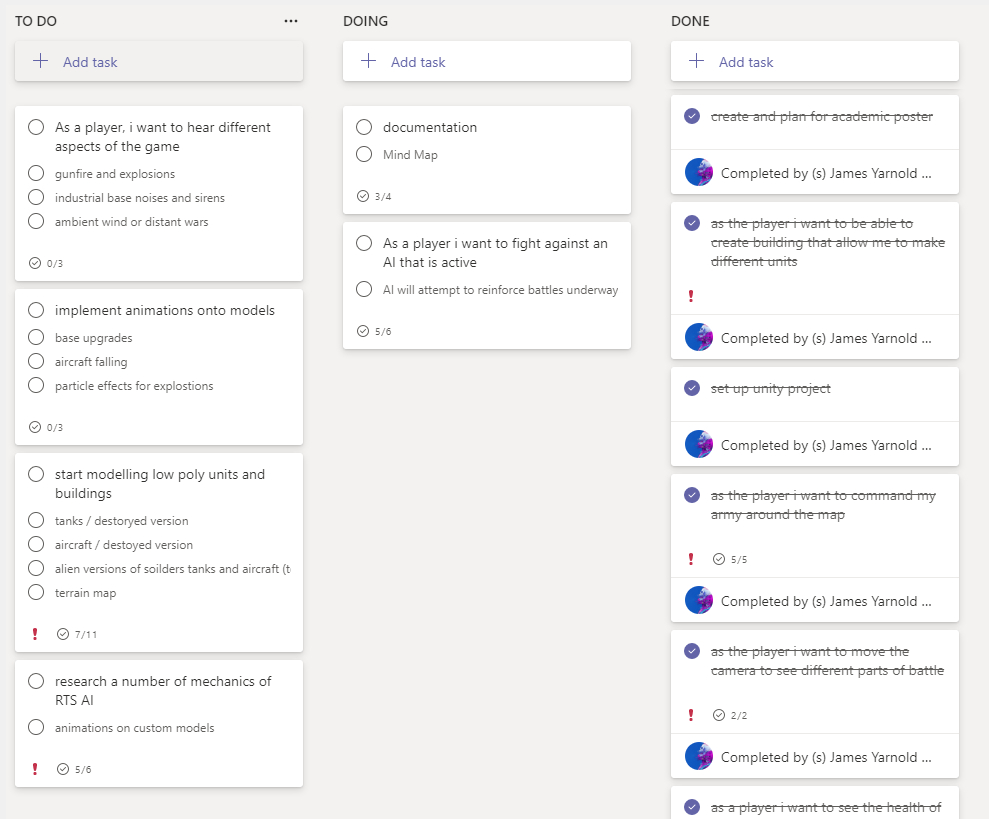


Figure 1, showing my Kanban style board during sprint 7

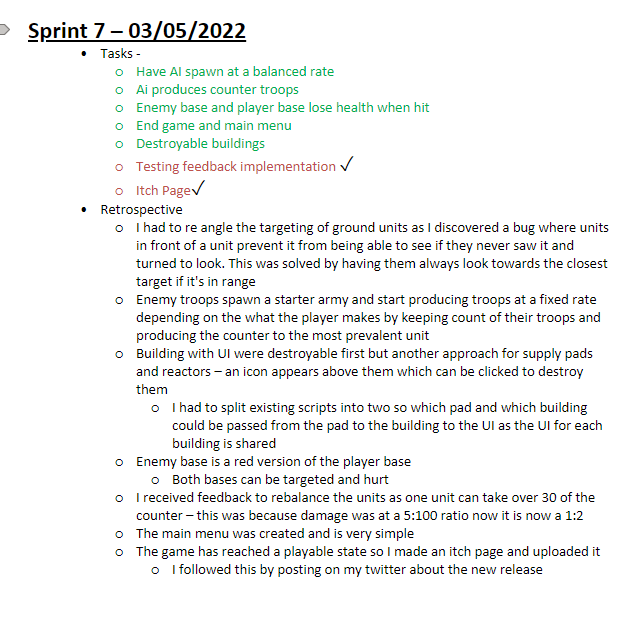


Figure 2, showing my Sprint Retrospective at the end of sprint 7

Research and development was an important step in creating the project Systems. They were written and broken down with the use of diagrams and charts, centralised into the GDD, as per manifesto guidelines, and derived the projects backlog which allowed the project to be separated down in MVPs and MAPs. This was important as understanding the aims around creating the different parts of the game would set out the paths and avenues that would guide future sprints, therefore planning centred around revision and research would be more effectively achieved with more specific conceptual points being brought to the forefront.

To achieve these methodologies, the project needed a viable understanding of platforms in which to launch these project management strategies. There was a number of applications and packages used for project management. The aim was to keep the amount of these down, to centralise progression and development into apps that specialised in project management and development fields also, having connections to past experience at university, meant time spent adjusting to new software and APIs wasn’t needed. I also needed a version control system with analogous criteria for project redundancy.

## Project Management

### Microsoft Teams

The project uses Teams to host the project board and documentation because it met many of my prerequisites for choosing an API. I was familiar with the software as I’d used it twice before in two separate modules, where I’d acted as project manager for those projects and used the planner add on to gain access to a Kanban style customisable board. This is where I created user stories to form the back log tasks that were further broken down into check lists of tasks which were reviewed upon sprint finish and sprint start. I also used the ‘One Note’ add on for teams which allowed for a better storage of information, as I intended for retrospectives on each sprint, I chose to have the board and retrospective writing area on teams mainly for consistency and centralisation as well as sharing documentation around system mechanics and contacting my supervisor.

### Testing

There were two types of testing adhered to during the course of the project. Technologically driven compilation errors and user driven perspective analysis. Google forms was used to host the questionnaire to complete testing around the mechanics focusing on the base and resources as well as menu GUI integration and information flow and useability. Specifics on the questions and implementation can be seen in sprint six and seven in the implementation section of this report.

The main way to test and debug for problems was to have serialized fields in the Unity inspector, allowing variables and numbers to fluctuate visibly in real time. Specifically observing variables when certain mechanics were supposed to be active or triggered, allowed deductions to be made towards problems or successes during the course of development for each mechanic and sub mechanic. By increasing variables, such as health, to exponentially higher than normal gameplay conditions, bugs could be more accurately and efficiently be seen and solved with the added time to reflect on active gameplay. Bugs were often reported in the sprint retrospectives as to allow more time to be dedicated in other sprints, to figure out or avoid errors.

To stick to the agile manifesto, useful qualitative user tests were conducted and involved in the development of the project with the purpose of reflecting and receiving feedback given on work up to certain points. This ensures the project ends up being suitable for the target audience and more errors are spotted and corrected before it is declared completed and before, the MVP is reached.

User perspective is also important because it leads to the discovery of end user defined, logical problems. This lifts the proverbial veil that knowing the full complete context of the game give to the developer perspective. Relieving this perspective blind spot better helps identify problems and oversights it causes.

### Version Control

I primarily used GitHub for online cloud storage to back up my work and save sprint progress. I used the GitHub desktop API due to its ease of integration and use in projects and on local systems. I used it frequently to save progress during sprints and at the end of sprints, leaving an appropriate title and description for each commit.

## Technologies

### Visual Studio, Unity and C#

Unity is the game engine the project was built on and while being popular in the indie game development community for its ease of use and ready accessibility, it provides two intrinsic systems built into it to achieve mechanics around movement and around the AI of player-controlled troops and AI controlled troops that this project practises to achieve goals around the creation of its AI.

### Navigation mesh

The Unity Navmesh or ‘navigation mesh’ is how developers can declare walkable areas for characters in their game scenes. On one level it collects all available meshes that are rendered and generates a surface on them to represent areas that are walkable for characters called agents. This is important as baking the mesh is a costly process, and it is more effective when done once and on objects that won’t be moving at run time. It also separates the surface into convex polygons, making it so paths between vectors are confirmed clear and cannot have an obstacle between them, it also then stores which polygons are next to each other. This allows wider understanding across the entire mesh when searching for the route to the agent’s destination. (Unity, 2021)

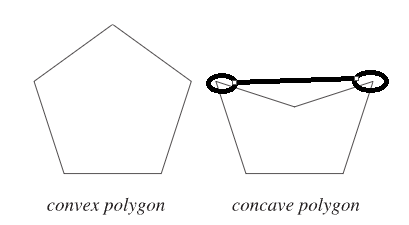


Figure 3 visualises the reason why convex shapes are used in baking unity navigation meshes.

With the information gathered at a global level, individual agents that exist on the navmesh can use these shapes and follow the edges to reach a destination in an efficient way at a local level.

Within the Unity libraries, methods exist for manipulating different important factors of the component attached to the agent. Effects such as its navigation abilities, like steering and obstacle avoidance, can be changed using these methods and properties. The project uses the methods associated with Set Destination and stopping Distance for the majority of its manipulation, so that for the AI in conjunction with ray casting to provide more information about individual agents at run time.

Set destination triggers Unity to calculate and path for the game object towards a vector3 location and set allows the agent to move towards it. It contains “pathPending” which can be called on as it acts as a Boolean and returns true if a path is being calculated, this is because set destination may take a couple of frames to calculate and move the agent.

Stopping distance refers to the agent’s acceptable radius of achieving its targeted position which is trying to move to. As other agents may be taking up the space and will be continually blocking its ability to move there.

### Ray casts

Raycasting is the process of shooting invisible lines, or spheres, from a point in world space at a predefined rotation and often a predefined range to identify colliders and game objects that intercept with the invisible rays. This is extremely useful when trying to ascertain what is around a primary game object because developers can access secondary objects the rays hit as the ray’s store information about what they hit. These hits allow the manipulation of the game scene further in ways depending on instances of distances, tags, or directions that the game object may define.

# Legal, Social, Ethical and Professional

## Legal

Defend and capture was developed on unity and therefore needs a licence to reflect its size and scope in the industry which was chosen as the personal licence as in this state in maintains free usage as Unity charges licencing fees to users that can produces games that generate revenue of USD100,000 per previous fiscal year. Unity hosts a community workshop where people can upload their 3d assets or other works for people to download and use. Defend and Capture uses assets from the unity store but uses appropriate licencing that allows use in even commercially motivated games but remains free to use and publish with your game. These assets include:

* Too Many Livings Room Props (DEV, 2018)
* Low Poly Helicopter Pack Free (Ganti, 2018)
* Cartoon Tank Free (Comeback, 2020)
* Voxel Furniture Free (VIRTI, 2020)
* Too Many items: Kitchen Props (DEV, 2018)
* 15 low poly models (luo, 2021)

The project uses navmesh addons that were hosted on a project sponsored by unity, while the project itself was unused, but the components located in the ‘NavMeshComponents’ were imported into Unity as these were the most recent uploads of the components research led to finding. (Brackeys, 2018)

The Pan European Game Information board or PEGI is the UKs game content rating system which is industry standard across games sold and distributed across multiple platforms in the UK. Developers would fill out a questionnaire on participating storefront platforms such as the Apple store or google play store and the PEGI rating board would quantify their rating and display it with the game. PEGI themselves are a member of the IARC or international age rating coalition that brings together rating systems from around the world so when developers fill this questionnaire out a rating is given in each region that reflects cultural norms and expectations. (PEGI, n.d.). Defend & Capture is estimated to be a 12 to 16 rating due to a simulated death around humanoid figures with minor implied violence, this makes it accessible to the age demographic it is targeted at.

## Social

To get an accurate understanding of the game’s reception in the public sphere a round of testing and the resulting implementation was completed. Testing needed to be done contextually with the target audience to have the most viable solution implemented into the final version. It was however difficult to find a sample audience that matched the ideal younger age of the target audience of 16 or above as accessible networks yielded subjects of ages 18 or above, while this was inopportune, the main objective for the game was to act as an introductory platform to the genre so lesser experience to strategy games was equally important. Every person who took part in the testing was 18 or above but with little or no experience in RTS games.

## Ethics

To conform to the university Code of Ethics when using human play testers, each tester had the purpose explained to them about what their role would be during it, and what their results would be for the game. I made it clear to testers that what they said or wrote was all constructive and appreciated. As per university policy, consent forms were sent out and returned by the end of the sprint and stored locally.

Defend & Capture is a classic real time strategy game focused on fighting and the simulated war of a battlefield. On the surface this is a fitting idea for a game, but as part of the target audience is the ages 16 years or above, it is arguably adding to a sense of desensitisation that the younger generation may be being exposed to, or at the very least, portraying war in an unrealistic way. Only showing parts that aim to be fun and ignoring the human cost real life wars inflict on people. With the recent war in Ukraine that has moved many people into action against war, it was a responsibility and ethical decision to use the games platform on itch.io to promote impartial charitable works that help the ongoing humanitarian crisis the Russo-Ukraine war has caused. This is why there is a link to the red cross, a fundamentally unbiased organisation, on the Itch.io page. Impartiality was an important factor in deciding a charity as the goal, of the presence of the link, is to help, not to alienate anyone on the basis of perspective, as the war at the time of writing is a highly controversial topic. (British Red Cross, 2022). But the need for humanitarian aid isn’t in dispute.

## Professional

Defend and Capture aims to be a standout portfolio piece as the biggest project currently on display and so takes the top centre spot, so it’s seen and read easier by visitors. The portfolio lists a number of skills developed with the project so site visitors can see what skills were necessary to build it, as well as hosting a link to the itch.io page that gives a greater insight into the project by having more pictures, a video, and a more detailed description of the game.

# Implementation

All sprints and their respective retrospectives are located at [appendix two](#_Appendix_2:_Sprints)

## Sprint 0: 26/11/2021 – 10/12/2021

### Overview

The objectives for this sprint were to first conceptualise and then document many of the game ideas and features on paper as well as set up the potential platforms of project management and research avenues of success for the features planned for the game. The modelling of assets on Blender was also an important step to begin with once the conceptual ideas were created, and to start getting a quick foot hold on the feel of the project in the Unity setting. This foothold is extended by research into potential solutions to many of the main RTS mechanic staples the project can expect to have.

#### Tasks

* Record ideas for upgrade system
  + Diagrams
  + Skill tree tables
* Research different mechanics
  + Camera controls
  + Pathfinding
  + Unit controls
  + Modelling effectively
* Start blender models
  + Solider
  + HQ
  + Buildings

#### Implementation

Defend and Capture features several mechanics designed to be beginner friendly due to its targeted audience. The main mechanic is a rock paper scissors style combat where units the player and AI make counter specific to other units. This is planned to cause fluctuations in the heat of battle to spark a level of user engagement and concentration. This mechanic also ties in the player resource and base management as the player must control their spending of resources to build buildings that produce that correct unit counter of troops against the AI. This is limited as a power reactor must be built in order to build some of these buildings. Having the base be at a single location on the map ties together both mechanics of combat and base management to create a loop of gameplay that can best be described as such:

1. Building and resource management.
2. Creating units and manoeuvring them to battle.
3. Watching the battle and clocking the units you’ll need to make in the future to counter the AI units.
4. Repeat.

At the time of conception, when the features were in this planning stage, upgrades to each building were to be planned for and implemented at a later stage as stretch goals, once the foundation of at least the first level buildings had been applied. Figure four shows the conception of the functionality for each building as well as the upgrades for each building as well as the types of troops and what they are weak/powerful against.

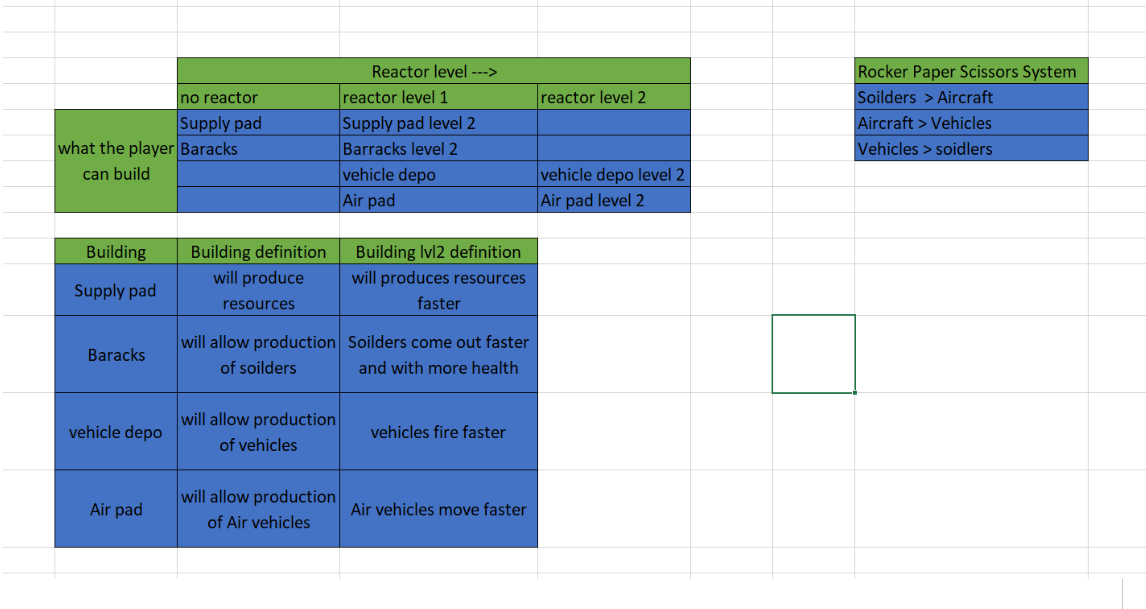


Figure 4 documentation created to plan the system of building and upgrades

This rock paper scissors system is not new to RTS games as Defend and Capture gained inspiration from the works of Ensemble Studios, the creators of Age of Empires and Halo Wars. Halo Wars was made for Xbox consoles in 2007 and featured new unit control mechanisms that were made simplified so players had an easier time using the Xbox controller to move units, they stated that they felt that while porting Age of Empires to Xbox they “first learned about many problems that come from porting a game to the console” they said this in reference to their attempts at using PC oriented control schemes (Remo, 2008). This ‘build from the ground up’ way of designing the controls for their games was used as a strategy which the project implemented when it came to researching controls and movement for this task. To build upon the strategy Ensemble used for its controls, the project planned on moving these simplified controls back onto PC to fit the target audience of newcomers to the genre.

In this sprint modelling for the buildings for the main base and soldiers was completed (final versions shown in figure [5] and figure [6]). This was the priority as the project flow would start at the beginning of the gameplay loop. Starting with the creation of buildings and the ability to create troops. Blender was used as the 3D modelling software as low poly assets could be created for free. The main base (far left on figure five) was designed to have seven pads around the outside as areas to place buildings while the soldiers were modelled with degrading health versions and upgraded versions, for their assault weapons I used a model I’d previously made for their assault rifle, and I modelled a rocket launcher for them to have on their backs.

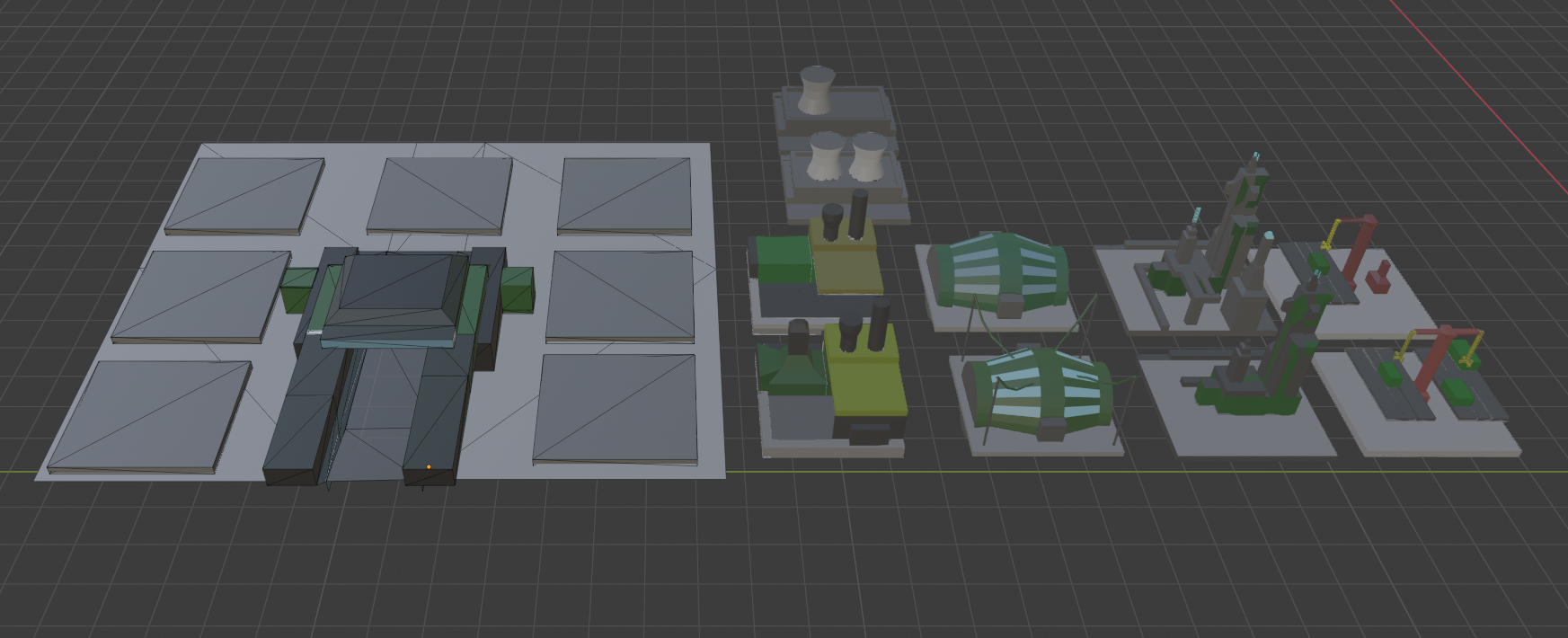


Figure 5 blender scene of base and buildings

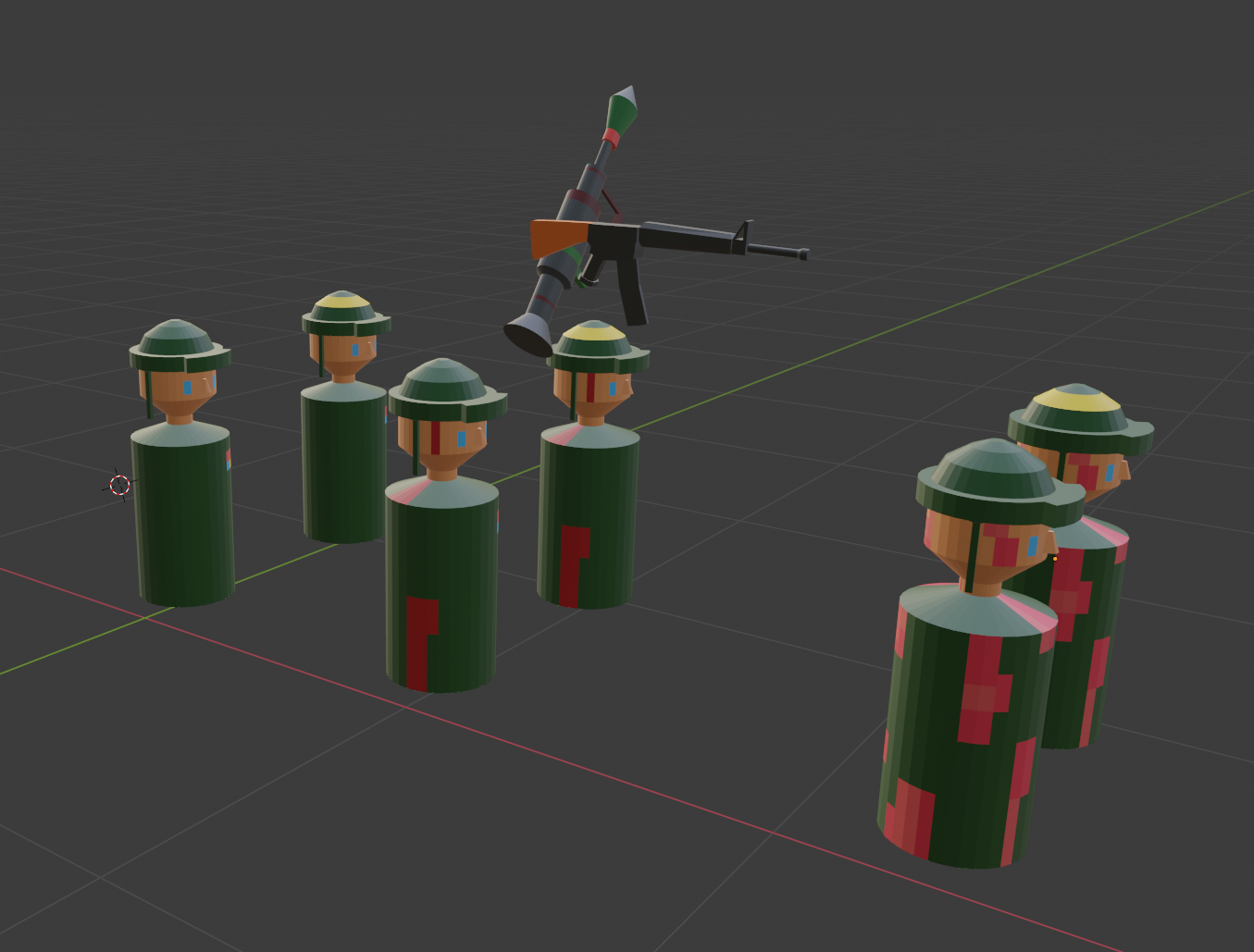


Figure 6 blender scene of soldiers and guns

## Sprint 1: 10/12/2021 – 26/01/2022

### Overview

In this sprint the goal was to begin the programming of the camera movement as well as creating the UI for different buildings and having the option to spawn these buildings on the selected pads. There were several iterations of how GUI would appear after the player clicked a building or pad, and these variants were saved and can be seen at [appendix x]

### Tasks

* Camera controls
  + WASD QE movement
* Create UI for different buildings
  + Have this UI open when the building is clicked on
* Create UI for each pad
  + Spawn the correct building on the correct pad

### Implementation

The implementation of the camera movement was important to start with, as future additions to the game may require the movement of the camera for bug fixing, and it allows for better context for the starting platform the UI systems would utilise when they come to be applied further in the sprint.

The final version of the camera behaviour was attached to a parent game object as the camera itself was to be tilted at an angle, this made the local directions of the camera different to being purely on the X and Y axis. The parent object is straight and can be more accurately translated across the axis.

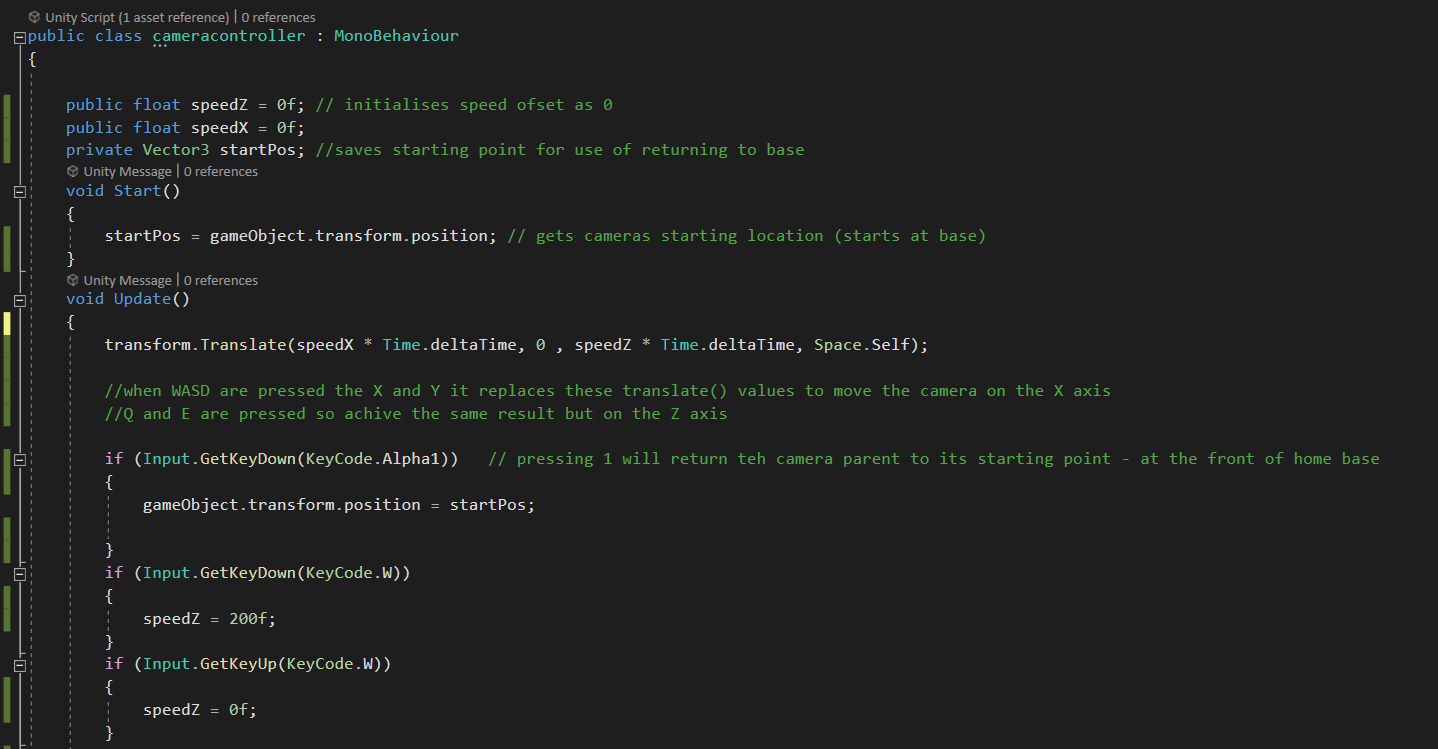


Figure 7 code snippet of camera controls

The GUI of the pads and buildings had the aim to be unobtrusive and quick to access as the game was planned to be fast paced, with mass units being created, and a lot of backwards and forwards from the main base with the camera work. Initial prototypes for the UI were made so buttons were smaller and used images instead of words to convey messages quickly but an error with the odd shaped buttons, ([seen in appendix 3](#_Appendix_3:_Scrapped)) made it so in the circular shaped UI design buttons overlapped and were not accurate to select because of their proximity to one another. This led to a redesign into a more conventional UI, where the UI swiftly animates on screen, where buttons had names and hovering over them gave a description of their uses.



Figure 8 Current Menu Design

Following on from this, the next task was to instantiate the selected building on the pad. A solution was investigated into having a parent script that connects the pads and the UI together. Information about which pad to build on is given when clicking on the pad for the first time, and what type of building to create is gathered through the UI button (as seen above) when it is clicked. To conjoin these collectors, a script was introduced to centralise the information and act an intermediary between the pad script and the UI script. this introduces a flow to the code that encapsulates these three scripts together to do what it does. The pad script has the notable inclusion of whether or not it had been built on. This is to prevent two buildings existing on the same pad as exiting the menu and clicking on the pad again before the coroutine was finished allowed the play to do this until this condition was added. The building instantiation also works by connecting two pivot points together, located in the middle of the pad and the other located at the very bottom centre on the building. This puts the building directly in the centre of the pad.

## Sprint 2: 26/01/2022 – 22/02/2022

### Overview

At this stage the core concepts of an RTS game and portions of the minimum viable product were being considered when allocating and prioritising tasks. The previous tasks have set up some important factors such as the base building, camera controls and Ui designs. To progress, several tasks on the troop manoeuvring and troop controls were decided on and added to the task list from the planner board. It was important to get this feature prototyped as controlling troops around the world space would be an important player interaction and would need to be tested to get a better understanding of how it could improve. Leaving enough time to complete the development of these mechanics and test them meant purposefully developing it earlier in the sprint phases was tactically sound.

### Tasks

* Create navigation mesh controls for units
* Selected units are highlighted and can move to points

### Implementation

The troop navigation was implemented via the navmesh agent component functionality that unity has embedded into it. The nav mesh agent component needs something to detect walkable surfaces, and while the option to bake surfaces into these walkable areas using object meshes is available on the standard version of Unity, it was predicted that extra utility, that the standard version omits, would be necessary further into the project. Unity provides an add on to these via GitHub that provides the extra functionality that was required. The extra functionality that the add on offers is the ability to apply filters to surface to allow a control of which agents and layers can use this surface. This was important as the games ‘rock paper scissors’ system required helicopters as a unit. For realism, these helicopters would be able to fly over objects that other units could not traverse, the additional function’s these addons would provide would be planned to go towards helping this idea become standard in game. The files for these addons were downloaded and added as a new folder into the project and became accessible via the inspector once added to an object.

The unit selection system was split into two systems. In this sprint a simple, basic control scheme was implemented that was purposefully close to ground level and simple for the player to understand. Clicking on troops will highlight them to the player so they stand out, then right clicking on the map will move them to that space, the player can also click more than one unit and move them all to a point on the map. The script only had to be successfully implemented once on one type of unit as other units would work the same way, and use the same scripts, using the same logic as the initial soldier model. Relative to this task, the only differing values being their model and size of the colliders for them.

The selection works by utilising unities mono behaviour lifecycle method, on trigger enter, combined with a Kinematic ridged body component and box collider set to not apply physics collisions (this was enabling kinematic allows) as the navmesh components sort this behaviour themselves. As seen in figure nine a gold circle and a spotlight were applied and would be removed and reapplied to signify the unit had been selected using the code seen in figure ten. The scale of the gold circle was also animated to fluctuate and the light dim in intensity and brighten again, this design was employed to increase the juiciness factor of the user experiences during the game’s progression. The script keeps track of a number count to see how many time the unit has been selected and uses the odd numbers and even numbers of this count to decide its selected status. When the number turns into odd or even, the unit is added to a parent script containing all the other units that have been selected. The point and click script use’s a ray cast to get the point in space where the player has clicked, then it iterates through this list of selected units and sends them to the point.

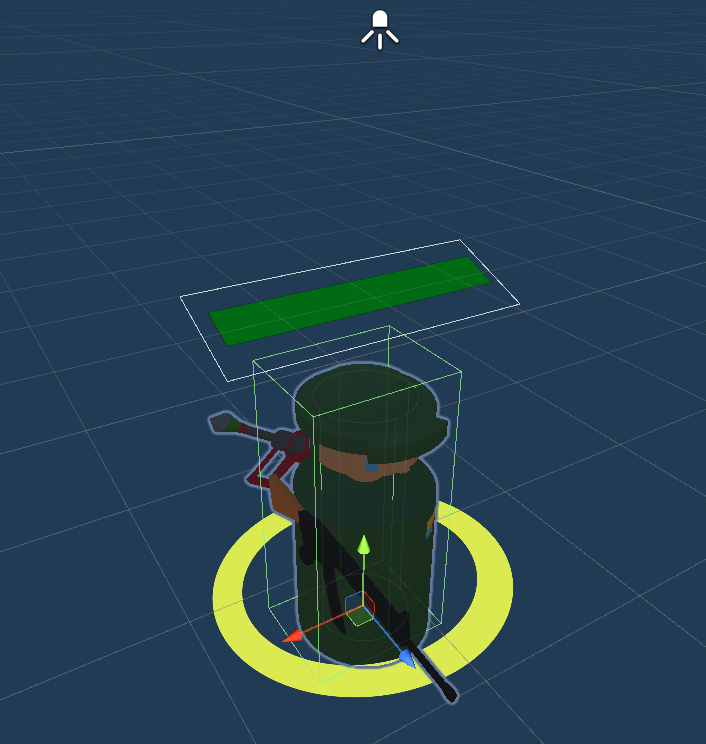


Figure 9 Soldier Prefab

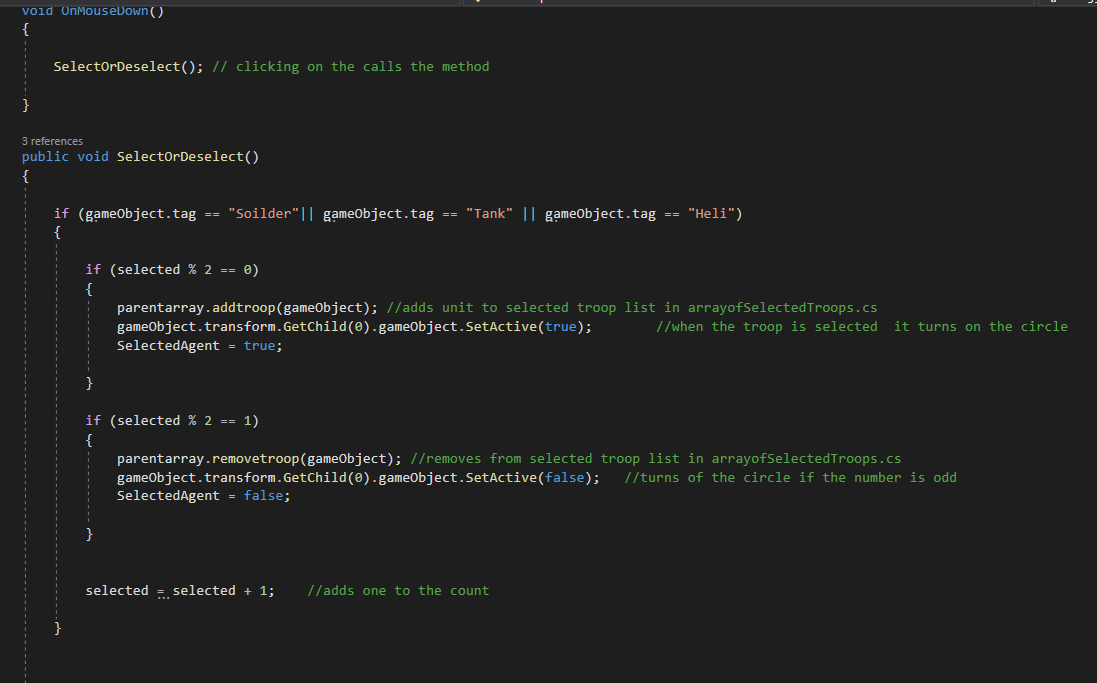


Figure 10 Selection Logic

Once the troop movement function was successfully added, a crucial bug developed when testing with more than a few units, and even less of what the player would experience. All the units tried to get to the same spot at the same time so they would all get stuck on each other trying to get to the exact point of the raycast and spin around. To solve this, I utilised a mechanic the navmesh agents’ component has that allows a quantifiable yield towards other agents labelled as the agent’s priority. A small hot fix randomises the agent priority, every 8 seconds, while they are not at their target destination. This allows other units with higher priority to reach it. When they do reach the destination, their priority becomes lower than is possible with the randomisation allowing other agents to push to get to the acceptable space. This was later expanded upon in future sprints to manipulate the units stopping distance, both in tandem work to eliminate this bug and cause a more fluid troop movement for the player to experience.

## Sprint 3: 22/02/2022 – 17/03/2022

### Overview

In this sprint, the goals were to continue the development in several aspects of already added foundational elements and update these works with functionality equivalent to the MVP requirements. The systems of the solider would be replicated onto the other systems, just as the systems on the buildings that create units could be replicated. This sprint was the effort of providing this system for the buildings. The building functionality took a shorter time that expected so extra thought and time was redirected during the second half of the sprint into improving the control systems for troops. Using blender to create the new units was not achieved in this sprint as the priority to improve the systems was greater and the Unity asset store has prebuilt models to use

### Tasks

* Have barracks and supply pad do their functions
* Model a simple tank and helicopter and map
* Implement selection circle for mass section of units

### Implementation

The base systems have been constructed to be heavy on the side of the UI with the model itself only providing access to that UI, it contains the code to instantiate units and return to the battlefield. To this end three user interfaces were designed to represent each building type that would instantiate the correct units at the correct place. In practise, the script for opening the menu (to get access to the create troop buttons) existed on the building and during this time in the project the script also existed on the menu itself, because of access to functions in the script for the buttons and the buildings themselves being prefabs. This would cause problems down the road when trying to add the remove building function, the script would be overhauled and eventually be split into two once this discrepancy was discovered. The three-unit spawners shared a spawn location and used a simple button that spawns a unit but to avoid having it pop into existence the units would spawn in the space in the base, as seen in figure eleven, and move out of that space towards a rally point. The goal of this effect was to highlight the importance of the main bases’ hub as an important part of the process of creating troops. This was important to convey as the type of game would involve the player protecting their base and destroying the Ais, so in this sense, the main base providing such an important function, reinforced the idea to protect it.

Reactors and supply pads did not have their own UI at this stage, but the functionality was complete for both. The parent script for these was the “statmanager.cs” script. The existence of the supply pad adds to an integer variable on this script labelled as supplies, this provided a centralised area for other scripts to access the number of supplies and power level. This was necessary as the building scripts system needed to know the number of supplies and power level before the player was allowed to build that building.

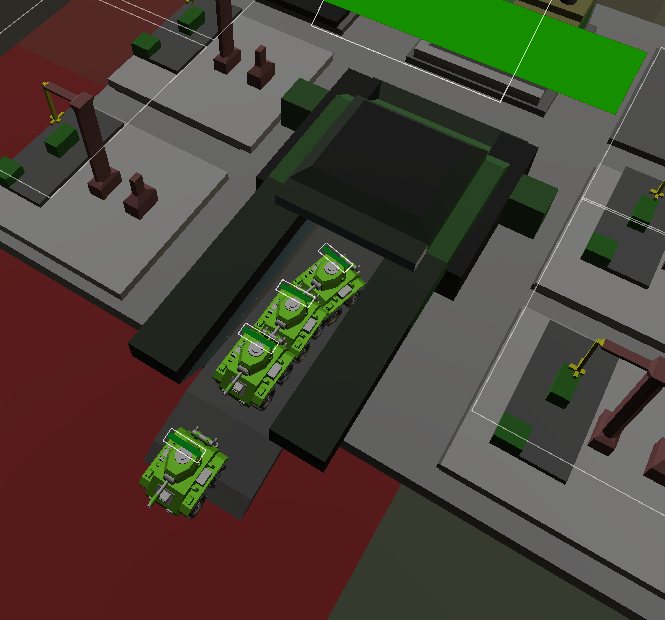


Figure 11 Units exiting base after spawning

This is the first instance of where units are moving without expressive permission from the player and the first steps into creating an AI for units on the battlefield. This initial movement sets a precedence within the player-controlled units AI that gives them partial self-reliance. Seeing them move by themselves should help establish them as their own entity withing the game, separating them from the base, its assets and even the player. The aim of this is to have the player feel like they’re commanding an army with their own loyalty towards the player and they’re commanding this army that can be dependable. Following this line of thinking, it was a understood a balance between these units moving on their own without player interaction was to be kept in as too much would be intrusive and make the units harder to control, completely breaking the effect this and future mechanics would aim to achieve.

After completing this functionality, more time was spent on expanding the options of unit control. This stretch goal from the MAP was brought forward after developer testing pointed towards a tedium when clicking many troops to send an order to all of them, then clicking them all to deselect them. This tedium needed to be solved with a faster way to select and deselect, as the fast-paced environment that was being moulded was threatened by this. As well as addressing this issue, extra time gained from implementing the building functionality quickly, combined with a lower priority level to create models for tanks and helicopters meant time could be allocated in this sprint to solving this issue.

The design for the new controls, at a base level, was a circular area that could be moved over units to select them, then moved back over them to deselect them. This gave a wider circumference to hit units instead of clicking them individually with the mouse cursor. Using a similar raycast system to the moving of troops, an invisible navmesh agent with a large spherical collider was introduced, it would be updated to the mouse position when it is active and while inactive its functionality would be irrelevant, it has the tag “Selector” and units would register this tag on a collision to activate its methods to select or deselect the individual. The additional functionality of tapping shift would quickly deselect all units that were already selected so the only units that would be selected were the ones this circle moved over.

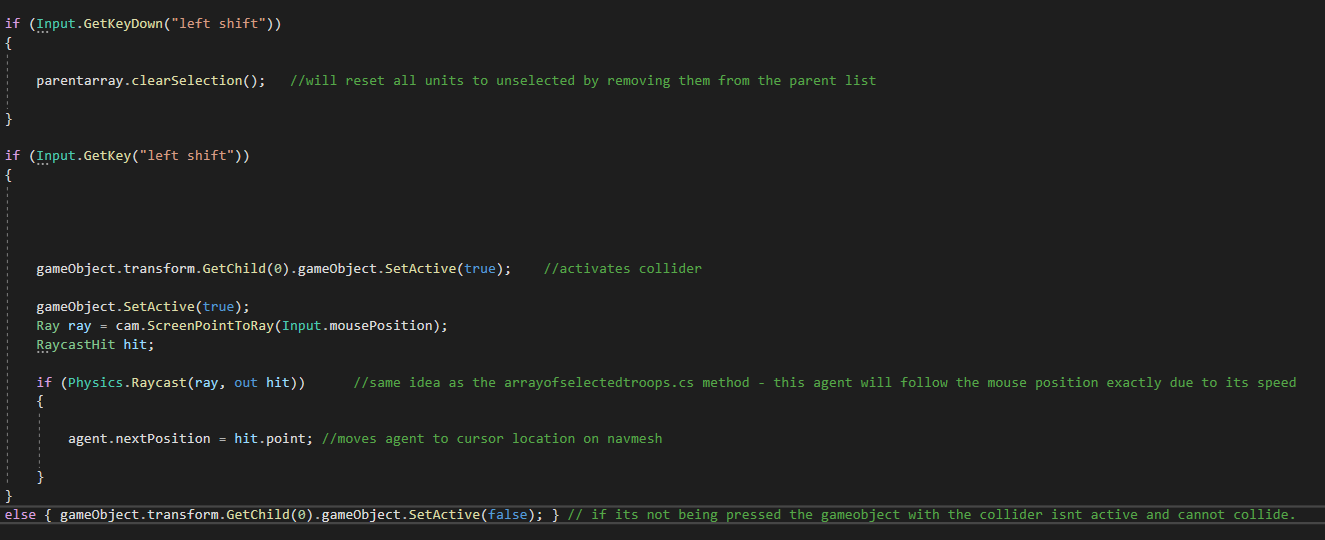


Figure 12 logic of selection sphere

Before the end of this sprint, it is noted that an error was occurring when the shift button was pressed. The original way of removing units from the parent list of selected units was causing this error, it was noted in the retrospective at the end of the sprint for consideration of tasks in the next sprint.

## Sprint 4: 17/03/2022 – 05/04/2022

### Overview

With the creation of troops and troop creation added the next phase of implementation revolved around player-controlled troops and their reaction when fighting and targeting enemy units. The combat would be a large part of this game, so more time was allocated into this sprint to fully explore the concepts and ensure quality work was being done and implemented. While this was the main focus of this sprint, bugs around the selection circle were blocking the projects main objectives and MVP objectives from being achieved. The modelling for the tanks and helicopters were also pushed back to this sprint but instead of modelling these, the unity asset store was used to get free assets to replace them, this saved time to fix more bugs and work on the player units AI.

### Tasks

* Fix selection circle
* Model a simple tank and helicopter and map
* Ally troop movements – realistic movement to avoid friendly fire

### Implementation

To solve the list manipulation error, a mass deselect method was created to reset each individual units’ values (values that determine the unit’s status of selection) back to their original state, following this it clears the selected troops list fully resetting what is selected. And removing the list after the fact meant the iterative for loops that deselected wouldn’t reference a null value. After solving this issue, the task was ticked in the retrospective so it wouldn’t be brought up again.

Breaking the combat system down into aiming, health and damage and AI allowed development to follow a path in the most logical way and build each part on top of the other, in this sprint aiming and some AI additions were completed. For aiming a temporary enemy was made for units to focus towards. It was declared that as enemies spawn, they are added to a list located in the Game Manager making this game manager the parent script for the targeting script attached to the player unit and enemy behaviour script attached to the enemy.

The player unit uses a range variable to sort through the list of enemies and store the enemies in range in a local list. Once this has been completed it iterates through these game objects to find the closest out of these objects, this is saved as the closest enemy which can be used in later methods. To improve efficiency, this code was put inside a coroutine that applies every two seconds, as having this method activate every frame would be taxing and unnecessary. The code seen in figure thirteen demonstrates this approach.

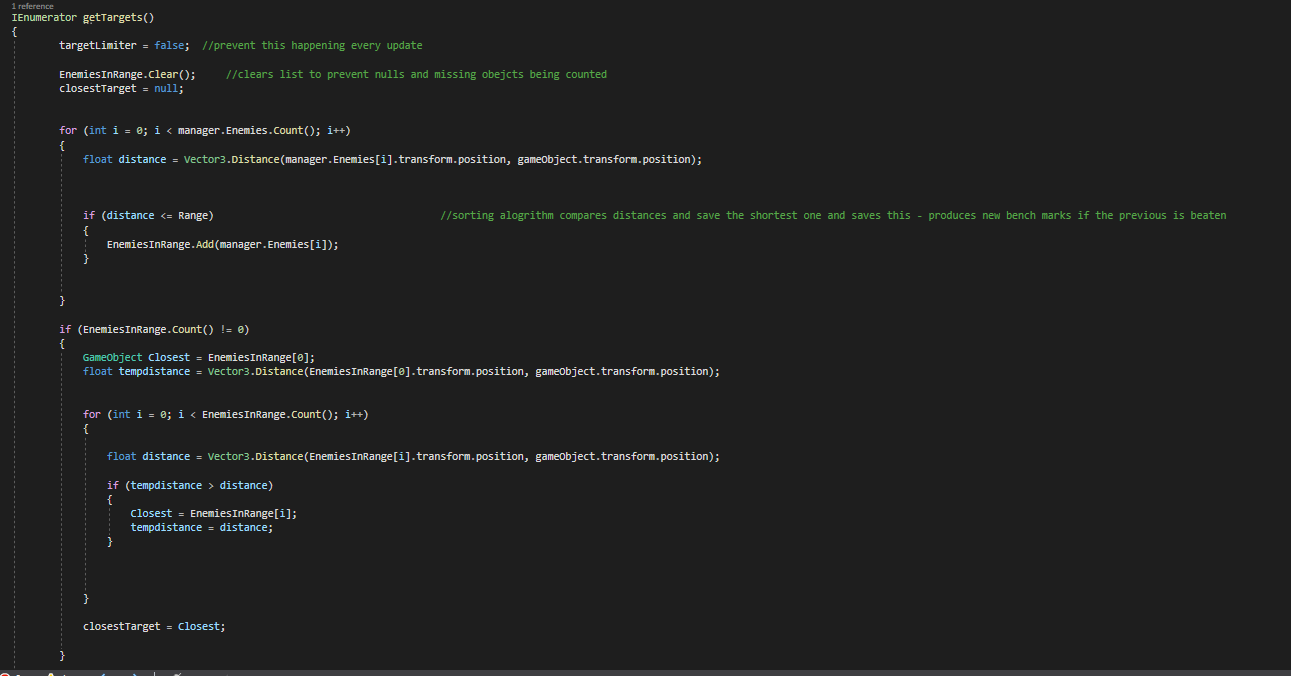


Figure 13 logic of selection sphere

Getting the closest target was important step in the combat systems approach as this meant the unit knew what enemies were around and could respond to this information. The next section of development was building off of this and making the unit face the closest enemy and analyse what’s in front of it.

Player controlled unit AI was beginning to take shape but to avoid unintuitive automatic manoeuvring and promote helpful synergistic movements a system of ray casts would be implemented to give troops a more intelligent analysis of what is around it. An initial ray cast is sent out from the unit towards the closest enemy, seen as yellow lines in figure fourteen. This allowed the unit to know if there are any obstacles between it and its closest enemy in range. Whether or not the unit could see the enemy was also tracked with a Boolean called “CanSee”. Storing this crucial information would be important for the second ray cast that is introduced. This ray cast, seen in green in figure fourteen, shows the range of the unit, which way its facing and what is directly in front of them. With all this information, the unit is able to tell if its in combat, if an ally is in combat, if what they are aiming at is friend or foe and consequently if they should fire or not. With all this information decisions can be made in a tactical setting without being obtrusive to the player.



Figure 14 ray casts visualised

The first level of this AI was a rotating angle around the unit that the unit could reposition to if there was an enemy in its radius, but there was an ally in the way. This was important as units could end up being bunched up when told to go to a point due to the nature of nav mesh agents set destination method, so initializing combat from a bunched-up cluster without a certain level of dispersion would result in the front units shooting, and the units in the back ranks not getting involved in the fight without being told directly to move and reposition to have a target. This level of micromanagement is unwelcomed in this beginner friendly game and removes the simulation-based angle combat was aimed to be like.

To start utilising this information and allowing this rotational based strategy, units were given four child waypoints around them that followed where they went, as seen in the hierarchy and inspector of unity in figures fifteen and sixteen. These were applied to the game objects prefab to be used in the code as vector locations and consequently locations the agent could move to in its immediate vicinity. A method was to be called when an ally in front could see an enemy, but the unit itself couldn’t, (figure seventeen) this is gathered when the relative forward facing raycast hits the ally and it returns the game objects tag as an ally allowing this method to activate.

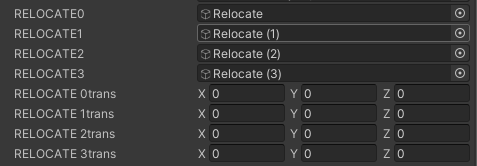


Figure 15 Unity inspector showing hoe the relocate targets were applied

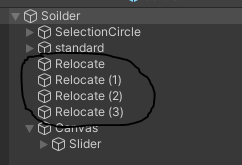


Figure 16 unity hierarchy showing how the relocate targets are children of the main unit

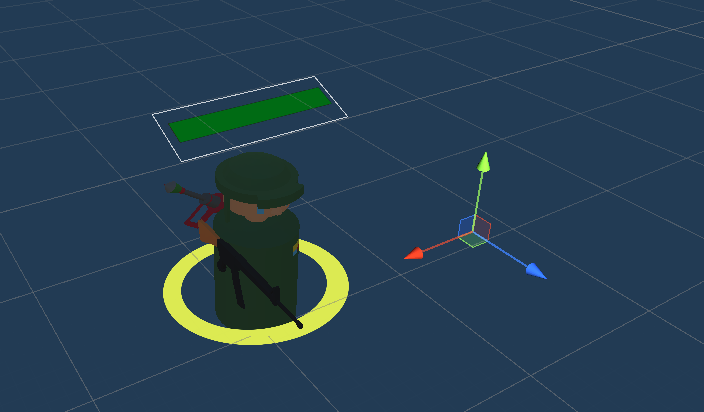


Figure 17 Showing the distance of one of the relocate targes from the main unit

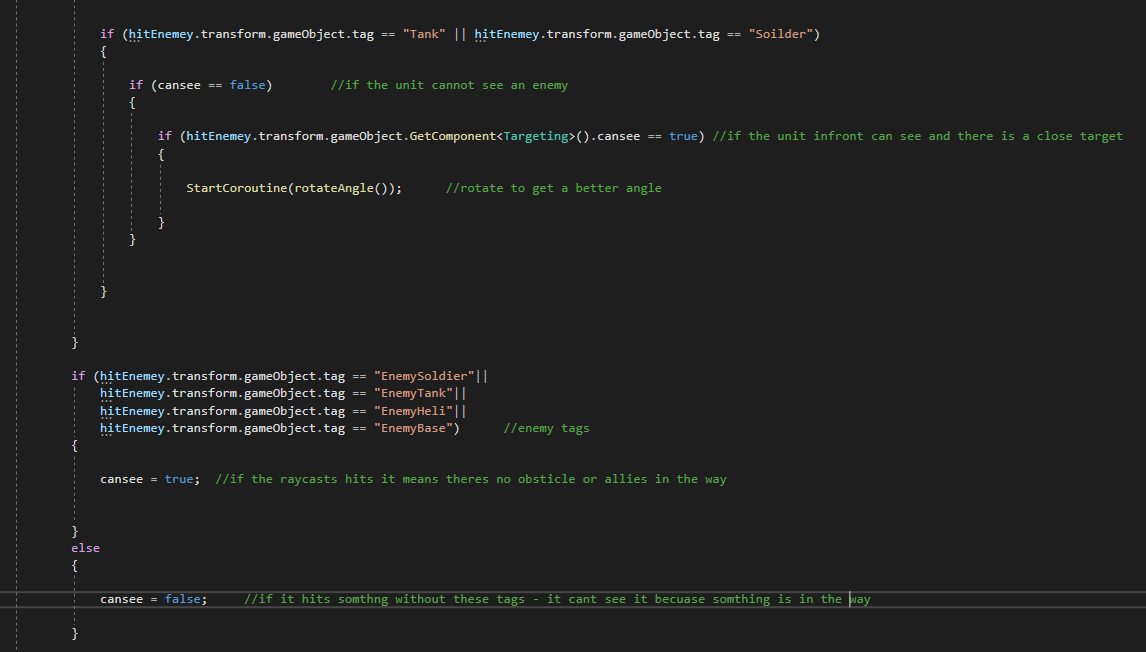


Figure 18 code that uses ray casts to see if the unit can see an enemy or not

The rotate angle coroutine is called every five seconds are met to reduce strain on the local system and to allow the unit to reach is relocate target. The method itself, featured in figure nineteen, uses a list of these targets and refreshed them every use to update the vectors with the new positions, it then randomly selects on to send the unit too. If this process is successful, the units will step from behind their allies and engage in combat and create a radial firing squad around the enemy unit.

This creates a dynamic battlefield where units are moving around and have the intelligence to get involved without user input but in a restricted way that does oppress or dominate the players own plans.

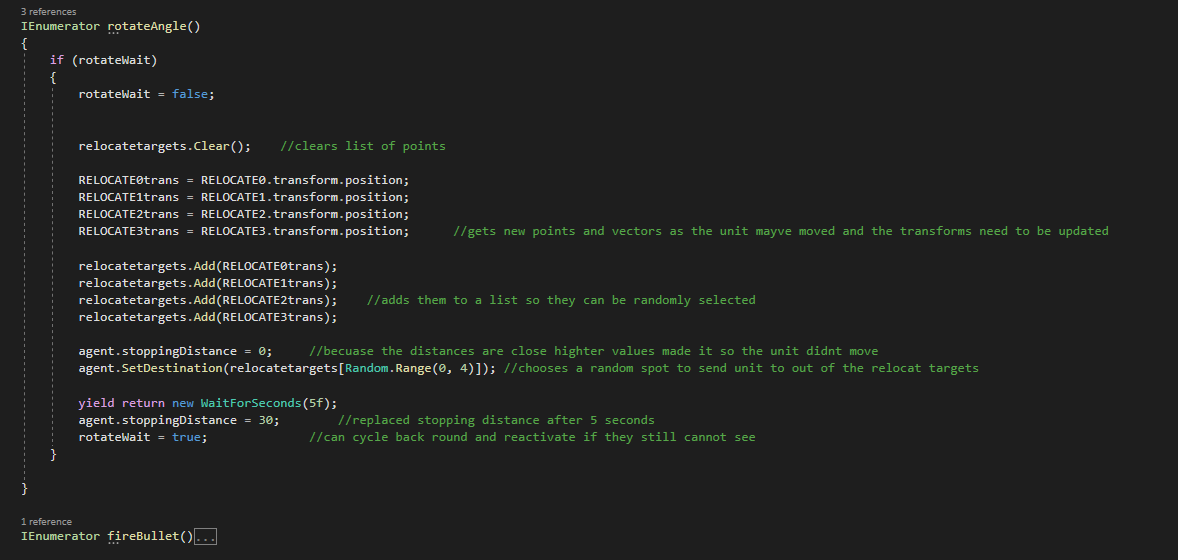


Figure 19 Relocate logic

## Sprint 5: 05/04/2022 – 19/04/2022

### Overview

This sprints focus was the continuation with the combat mechanics. As aiming and AI had been implemented into the game for the first of the player-controlled units, the first main object was to have player unit’s hurt enemies and shoot projectiles at their targets. Secondly, as up to this point in the project only the player soldiers had been created, the aim was to Replicate the logic of the barracks and its ability to create soldiers for the vehicle depot and air pad to create tanks and helicopters respectively. Similarly, replicating the logic of the soldier to create player-controlled tanks and helicopters that share the same AI functions. As a prerequisite to these tasks, assets would need to be obtained from the unity store. Creating the tank, helicopter and map assets had been a low priority task in previous sprints due to the Unity stores availability and time requirements for the building of said assets becoming quickly out of scope.

After initialising the main combat system for the player, plans were made to use the logic of the player-controlled units as a foundational system for the enemy AI. In that context, the first level of testing could begin that encapsulated; the main base building and its UI system; the resources and how they restrict troop creation; troop manoeuvres and controls and the feel of fighting into one questionnaire that aimed to examine one cycle of the gameplay loop.

Figure [20] shows the teams checklist for the amount of estimated assets I would’ve had to build, this task remains in the “to do” column despite having useable assets for the map tanks, and helicopter because these unchecked tasks equated to stretch goals that could delegated later in the project’s creation, even post university.

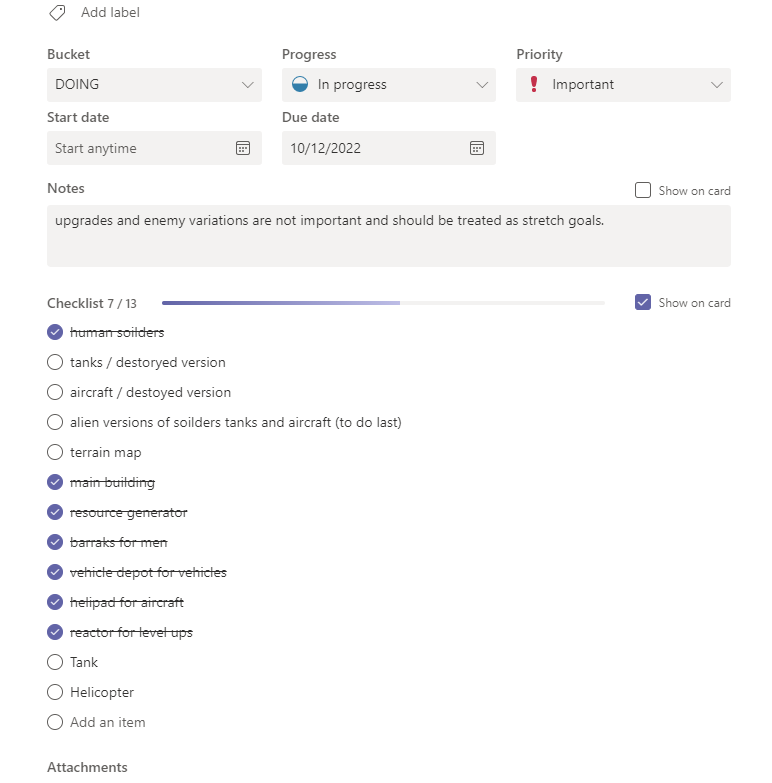


Figure 20 Blender models Check list

### Tasks

* Have troops hurt enemies.
* Find Unity store assets to replace homemade tanks and planes and terrain.
* Implement their mechanics using the finished friend infantry troop as a base.
* Create Questionnaire.

### Implementation

Two models from two different asset packs were used, and became the tank and helicopter design, notably for both the tank turret and the helicopter rotors were separate objects which was desirable for both of their designs. The tank would aim and shoot from its turret while its body stayed at the angle of movement, and the helicopter rotors would be animated and smoothed to create the illusion of its flight. The logic is shared between the unit types, however minor changes were necessary to accommodate the new tags and game objects.

The tanks slightly different way of shooting meant that ray casting from its forward position was less relevant so the rotate angle methods as discussed could be moved to the initial ray cast to achieve the same effect. To see this difference in the code, one can inspect figure [21] which shows a red circled area of code that applies to the tank only. Inspecting figure [22], the red underlined area applies to the helicopter and soldier as they both require to be facing the same way they shoot for the logic to apply to them while the tank does not experience this.

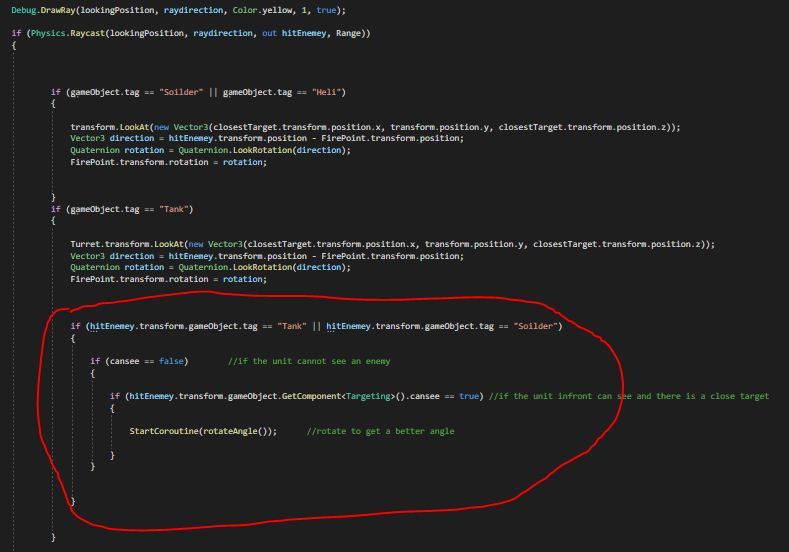


Figure 21 Tanks rotate angle located first raycast

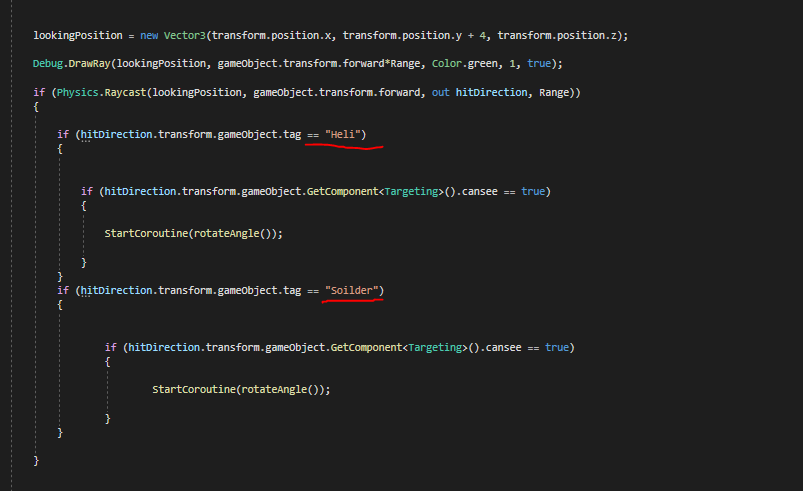


Figure 22 while the second ray cast activates the helicopter and soldiers rotate angle logic

With the implantation of each unit, different projectiles could be shot from each one. The projectiles are their own prefabs and react differently to each unit due to tags that have been applied to the projectile types, these can be seen in figures 23 to 25.



Figure 23 Rocket projectile and tag

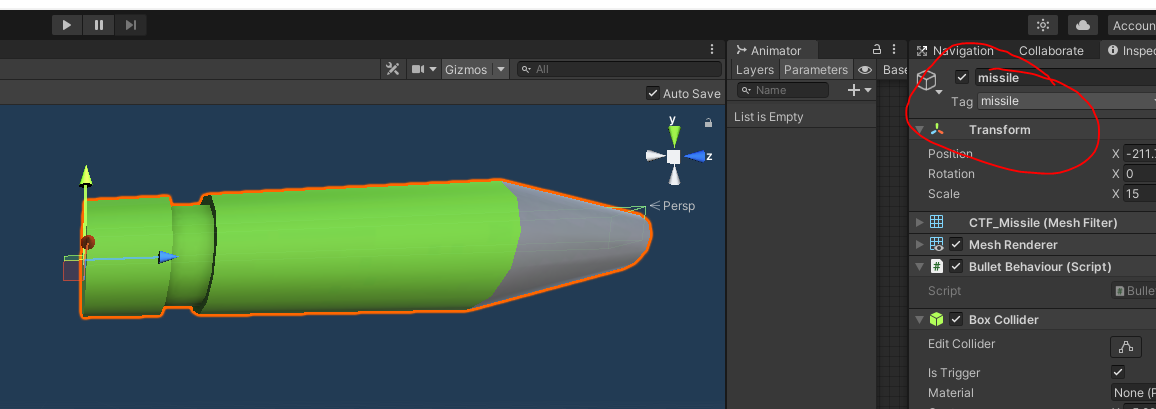


Figure 24 Missile projectile and tag

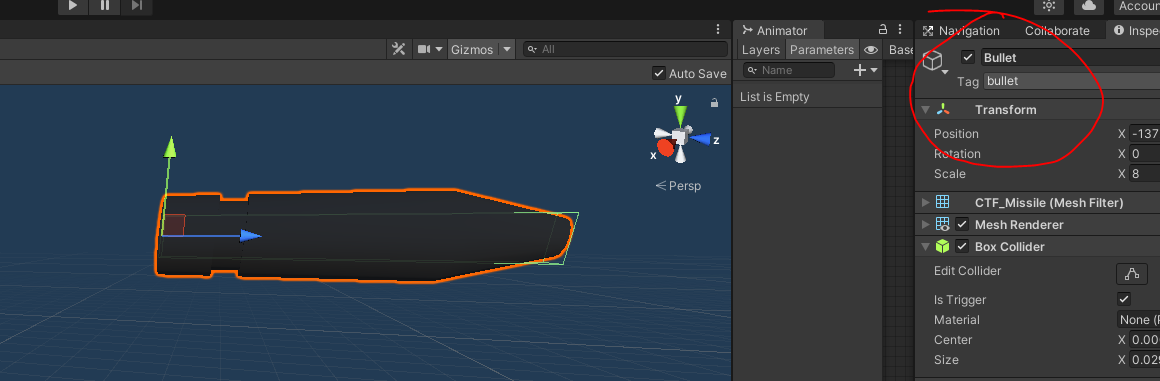


Figure 25 bullet projectile and tag

Using unities tag system, the ‘Rock paper scissors’ combat system that was planned to be implemented, starts to take shape with the implementation of these assets. More damage can be applied to units that get hit with these projectiles to give damage advantages and disadvantages based on its projectile. Unit types will only fire one type of projectile. This can be decided and declared, using the tag system the “OnTriggerEnter ()” method in Unity mono behaviour. This method triggers when there’s a collision and can take an argument about the object its collided with. Figure [25] shows the damage differentials of soldier unit. Notably it takes more damage from missiles, which is a projectile only the tank fires.

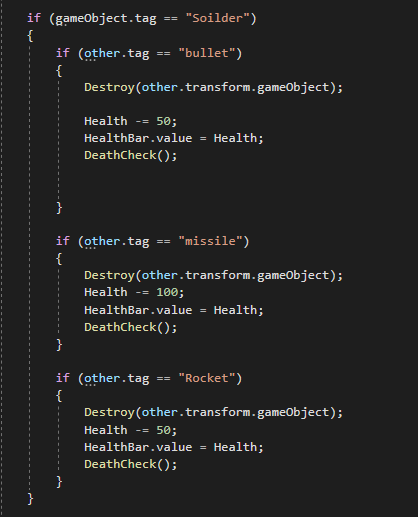


Figure 26 health and damage for soldier

The next Objective in this sprint was to plan a questionnaire. Using Google forms, seven questions were formulated to get the best quality qualitative and Quantitive feedback that could be translated into actionable tasks in the following sprint. Questions that could judge the accuracy of the tester compared to the target audience, were important to gauge their level of prior understanding of the genre, and questions about what they’re expecting to happen during UI sequences were also important, as quickly understanding the mechanics around the menus was crucial to keep a faster paced tempo to the game. Lastly questions focused around of ease of use were significant, as there are times players expect to be told certain things and others where they expect to be shown certain things. Knowing when to show players text verses when to signify mechanics was important to increase accessibility and ease of use in the menus. Questions and their specific results can be found at appendix[x].

## Sprint 6: 19/04/2022 – 03/05/2022

### Overview

This sprint was used as time to implement the enemy Ai. The concept of the enemy AI would use the finished product of the player AI as a foundation and build upon it to replace the player input with additional functions. Consequently, the conversion of the assets for the enemy AI involved scripting of the same logic and a changing of materials from green to red to signify the differences in teams. This sprint was also used to as time to gather testers and evaluate responses to questions, and plan reactions to reasonable requests. The fact that many of the systems in place could be reverse engineered to save time and achieve the same symmetrical effect allowed extra time for additional features to be implemented on the AI to make the game more enjoyable.

### Tasks

* Have troops hurt allies
* Have enemies turn and shoot at player troops
* Have enemies hunt and roam
* Implement mechanics using the finished friend infantry troop as a base to enemy troops
* questionnaire and interview people about the mechanics

### Implementation

At this stage much of the logic around shooting and targeting has already been implemented and to convert into logic the enemy AI could use, re-coding tags and retargeting of the lists to focus on all enemy troops instead of all ally troops when searching for targets in unit areas. This new implementation was separated into a separate script called Enemy Behaviour. There were several new additions added to the logic that performed analytical tasks per unit for it to discover its surroundings.

Firstly, the enemy needed to know the units, on its team, that were nearby so they could share information about what they’re experiencing around, for example, if an enemy AI unit goes into range of a player’s unit the other enemy units would need to know this to avoid them leaving the area and not fighting. The solution to this that was executed was a similar logic to the sorting algorithm the player AI used to get the enemies in range, except this is used by the AI to get the allies in its radius, it stores these in a list and then looks through these to see if the variable that gives them the status of seeing an enemy is triggered, it then uses them to get the closest target to them and moves towards them, stopping in its radius using an inflated stopping distance value before resetting it after enough time has passed. Figure [27] conveys the most important part of this code.

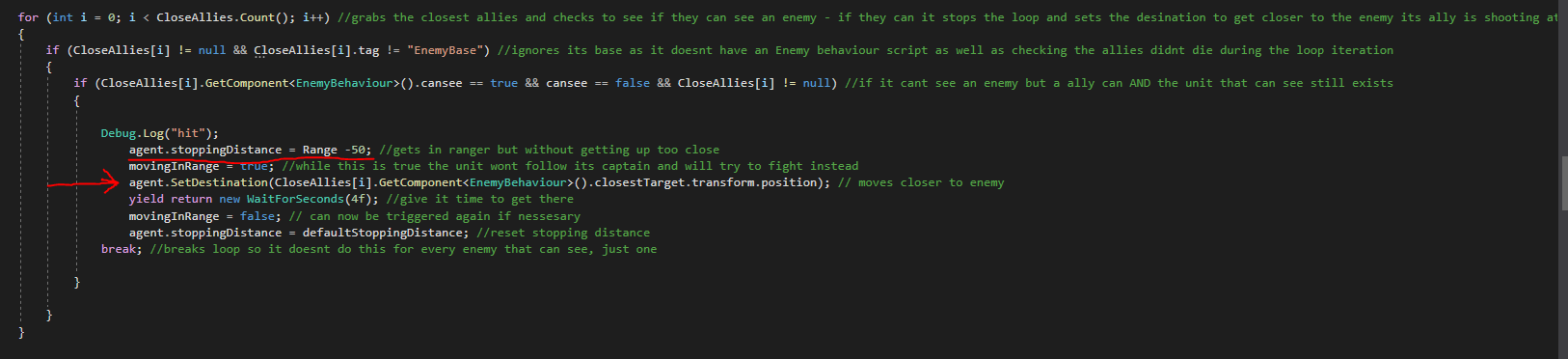


Figure 27 code for making enemy AI move into range of player unit if another Enemy in range spotted it

In this sprint testers gave feedback about many on the features in the game, and conclusions were drawn about how to improve the menu system, these tasks were added to the task list for the next sprint.

Crucial feedback on the user interactions with the menus was understood, as well as how well mechanics could be understood. To increase useability as per suggestions, text on the building screen would be made to turn red to signify it couldn’t be built. this new feature can be seen in figure [28]

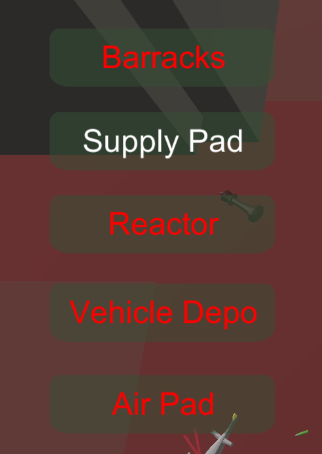


Figure 28red buttons for useability

## Sprint 7: 03/05/2022+

### Overview

The last sprint focused on finishing touches to the enemy AI and game type. This included adding an enemy base with spawn points for it, and the ability for troops to aim and fire at the enemy base, deal damage, and destroy it. Then finish with an end game screen. As well as the last finishing touches to the player base as destroyable reactors and supply pads were not yet implemented but needed to be to have a base building system that wouldn’t lock the player into having a base of supply pads or reactors. The remainder of this sprint was dedicated to report writing.

### Tasks

* Have AI spawn at a balanced rate
* Ai produces counter troops
* Enemy base and player base lose health when hit
* End game and main menu
* Destroyable buildings
* Testing feedback implementation
* Itch Page

### Implementation

The achieve the spawning for the AI, it was broken down into 3 phases; the beginning of the game where it’s assumed the player it introduced and to the game and can figure out the base’s menus, this phase lasts around twenty seconds where the enemy doesn’t create any units. After this phase, an initial army of twenty soldiers exits the base to protect and explore the map. It is after this phase the AI starts to analyse what the player is making and starts to create the counter units to fight better. It does this by tracking how many numbers of troops the player is making and actively counters the most of one troop the player has. For example, if the player makes five soldiers and ten tanks the AI starts to make helicopters because helicopters counter tanks.

# End of Project post-mortem

The development of the game took an ambitious stance when initial planning took place, believing that the initial base game fundamentals such as the combat and building could be completed quicker than it was, and time for the AI would be longer allowing a more expansive AI to be planned and thought about. However, while several notable AI features were implemented, the development turned to focus on useability and the experience of the player, as opposed to fully developing the AI, this shift in focus strayed away from the main objective of creating a fully comprehensive Ai package, but still achieved in becoming a standout portfolio piece and for the developer and still got a foothold on all the objects of the minimum viable product and some extended goals. Shifting focus like this is also beneficial to any end user that may end up playing this game, it was targeted at newer players to the genre so if they are introduced into an experience with useability at the forefront, it may cause them to really get into the genre, which is good for the industry as a whole.

The added time spent creating main functions in the game, lead to perhaps unorganised and inefficient code with hotfixes and not much room for future expansion without an overhaul or dedicated sprint to clean some scripts. While this doesn’t affect the game as of Alpha release, it may affect future development, specifically code around the enemy AI decisions, as this became a complicated system of ray casts, Boolean variables and parent lists that may be hard to introduce to a fresh developer to optimise.

Another prominent feature that needs implementation is audio. The game lacks any sort of audio which can be used in areas of useability, accessibility, and immersion. This is a major omission works against the game and future sprints should prioritise this before any more additions to the Ais or concepts.

# Conclusion

To conclude, Defend and Capture explored many aspects of industry standard agile development systems and provided important experience on how they should work and how to implement them properly and what to expect from the industry. The game provides a fun casual outlet to new players of the genre, but more importantly it represents the potential of a compelling RTS game that embodies the four years of university and the resulting passion this author has for the industry.

# Bibliography

Brackeys, 2018. *NavmeshComponents.* [Online]   
Available at: https://github.com/Brackeys/NavMesh-Tutorial/tree/master/NavMesh%20Example%20Project/Assets/NavMeshComponents

British Red Cross, 2022. *What we stand for.* [Online]   
Available at: https://www.redcross.org.uk/about-us/what-we-stand-for#:~:text=The%20Movement%20makes%20no%20discrimination,most%20urgent%20cases%20of%20distress.  
[Accessed 2022].

Comeback, 2020. *Cartoon Tank Free.* [Online]   
Available at: https://assetstore.unity.com/packages/3d/vehicles/land/cartoon-tank-free-165189

DEV, Q., 2018. *Too Many Items: Kitchen Props.* [Online]   
Available at: https://assetstore.unity.com/packages/3d/props/too-many-items-kitchen-props-127635

DEV, Q., 2018. *Too Many Items: Living Room Props.* [Online]   
Available at: https://assetstore.unity.com/packages/3d/props/furniture/too-many-items-living-room-props-129097  
[Accessed 2022].

Ganti, P., 2018. *Low Poly Helicopters Pack Free.* [Online]   
Available at: https://assetstore.unity.com/packages/3d/vehicles/air/low-poly-helicopters-pack-free-121151

luo, R., 2021. *15 low poly models.* [Online]   
Available at: https://assetstore.unity.com/packages/3d/props/15-low-poly-models-202061#description

PEGI, n.d. *How we rate games.* [Online]   
Available at: https://pegi.info/page/how-we-rate-games  
[Accessed may 2022].

Remo, C., 2008. *E3: Halo Wars' Rouse: Age Of Mythology Console-Controller Prototype Informed Game's Genesis.* [Online]   
Available at: https://www.gamasutra.com/php-bin/news\_index.php?story=19465  
[Accessed may 2022].

Unity, 2021. *Inner Workings of the Navigation System.* [Online]   
Available at: https://docs.unity3d.com/Manual/nav-InnerWorkings.html  
[Accessed 10 May 2022].

VIRTI, 2020. *VOXEL Furniture FREE.* [Online]   
Available at: https://assetstore.unity.com/packages/3d/props/furniture/voxel-furniture-free-170365

Wrike, n.d. *What Is the Agile Manifesto?.* [Online]   
Available at: https://www.wrike.com/agile-guide/agile-manifesto/#the-four-agile-manifesto-values  
[Accessed 23 April 2022].

# Appendices

## Appendix 1: Games Design Document

### Game Design Document – Defend & Capture

#### Genre & Goals

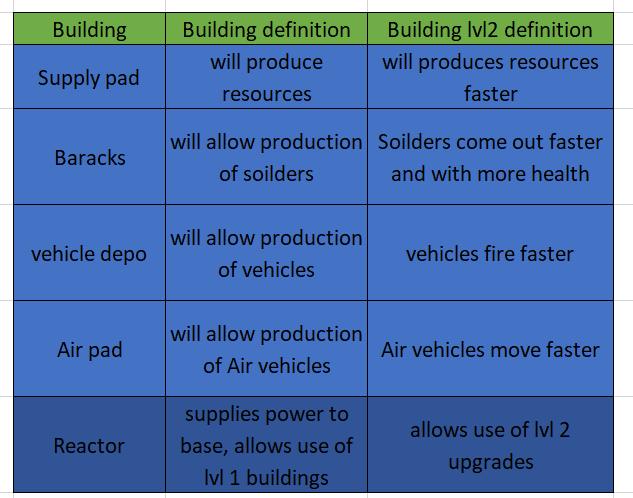
Defend & Capture will be an entry level, small scale RTS game targeted at an audience that is younger, more casual, or new to the RTS genre. The game attempts achieve this by using a low poly art style that is friendly for younger players and isn’t gritty or matured to the eye making it more applicable for these audiences. Although currently in early stages of development, character designs and the art style are established to be for these audiences with the first map being a toy soldiers aesthetic design.

The game represents a game of rock paper scissors between units, the player controls an army they’ve made to push and destroy an enemy base. An enemy AI pushes to do the same to the player. It attempts to set up a simulation style combat with AI aspects on the player side that helps guide fights, as well as enemy controlled AI that presents a challenging task to overcome for these newer/casual players. There are several main mechanics that are key to understanding:

* Base construction,
* Resource management,
* Troop controls and manoeuvres,
* Combat centric mechanics.

#### Base construction

The empty template base is given to the player at the start, it contains seven ‘pads’, these pads, indicated as grey platforms around the central part of the base, provide spaces for the player to spend resources to build one building on these spaces. The buildings allow for resource production, the unlocking of other buildings or the production of troops. All of which are important to utilise and link to the other main mechanics, but with a limited number of pads the player has to manage these buildings and discover which is the most prevalent at the time to overcome the AI.



*Figure [1]*

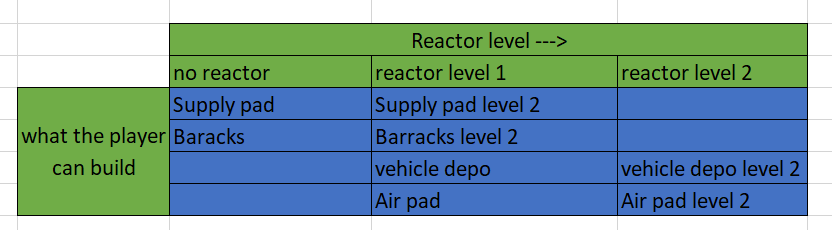
#### Resource management

Resources are split into 3 types and relate back to building that increase their counts (from left to right in figure [2]):

* Supplies
  + These are received when the player has supply pads. The more supply pads a player has the quicker they’ll get supplies. If a supply pad is upgraded, they’ll receive more supplies per pad.
  + Payer can spend these supplies on other buildings and upgrades, but primarily will use them to produce troops using other buildings.
  + Players receive these in the hundreds making it the most abundant resource.
* Power
  + Power dictates what buildings can be built at a specific time. A reactor existing on the base will provide 1 power, an upgraded reactor proves another power for a total of 2. Specificities are located in figure [3]
* Population
  + A player can create up to 50 troops as of early development, this may fluctuate depending on tests.
  + This reduces strain and keeps the game light. The more troops the more hectic it’ll be and the more power it’ll require on the PC to perform the AI protocols.
  + Barracks, vehicle, and air depots will create one unit per unit bought, this will increase the population by 1 until the player cannot make more.



*Figure [2]*

*Figure [3]*

#### Troop controls and manoeuvres

The player troops will be able to be controlled using the mouse and keyboard and will primarily use ray casts and unity navmesh system and custom arrays systems to navigate the map. The 3 ways of the player interacting controlling the troops will be:

* Single click
  + Single clicking a unit will highlight them in a gold circle to let the player know they are selected and ready to move, clicking the again reverts this and ‘unselects’ that unit
  + Clicking on another unit while another is selected selects this as well and they can be moved together.
* Group selects
  + Pressing shift will bring up a circle on the floor pressing shift will deselect all currently selected troops, this button can be tapped to unselect all troops
  + All troops that aren’t selected that enter the circle, become selected.
* Map clicks
  + Right clicking on the map while units are selected cause the units to move to that space.

During tight battles the player is expected to micromanage the position of units so they can get a better angle and can shoot past their friends at the enemy, the goal of this is to make battles more engaging, although, if the majority of your troops don’t get involved because they’re waiting on placement in battles, battles would become frustrating. To solve this player-controlled units AI will analyse if they’re in combat and attempt to move and get in range/ get an angle on enemy troops to shoot at without player input. Units will still prioritise commands over their own AI.

#### Combat centric mechanics.

Combat is a rock paper scissors style fight between each unit, using the system in figure [4]. In this stage of development combat is semi simulated with units attacking the closest units in range and shooting their own projectiles towards this enemy. The projectiles they shoot are what determines the damage and weaknesses thus creating the rock paper scissors system.

* Soldiers shoot bullets
* Tanks shoot missiles
* Aircraft shoot rockets



*Figure [4]*

The Enemy will attempt to utilise this rock paper scissors mechanic just like the player should, the AI can see what the players making by fighting them in combat, it will then send the relevant troops over to the battle to counter the players troops, the player will have to keep on top of this to win.

#### Minimum viable product

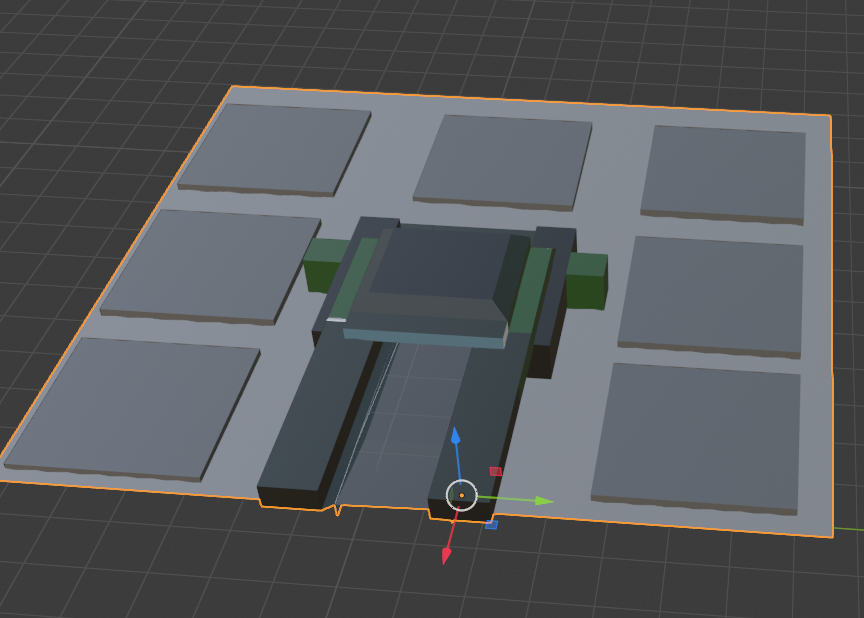
* Player can build buildings and each of them effect the game in their specific way.
* Players can control troops using the single click system
* Player units and enemy units fight and can hurt each other
* Player can destroy the enemy base and win, and Players can have their base destroyed and lose (Conquest)
* Enemy Ai moves around the map randomly

#### Minimum awesome product

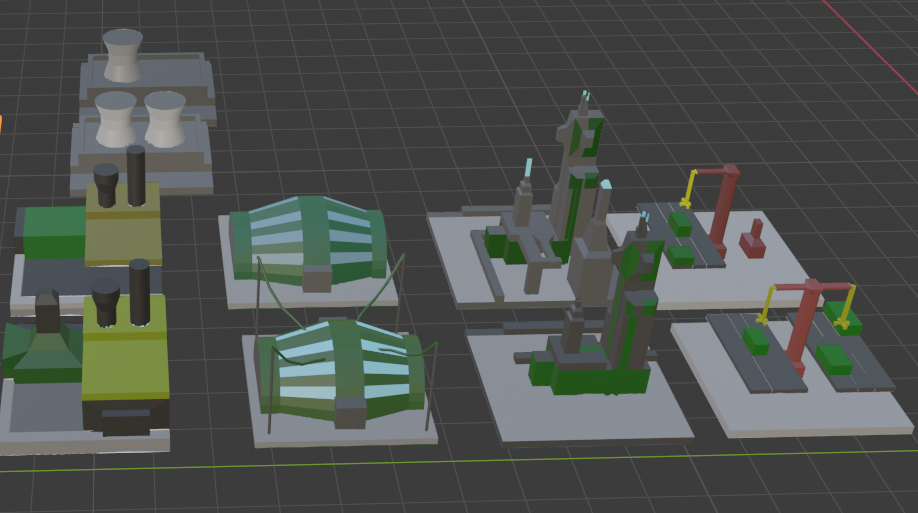
* Player can build buildings and each of them effect the game in their specific way.
* Players can control the troops using single click, group select, select all of one type of units and select all units
* Sounds for all different aspects
* Player can upgrade existing units using the building upgrades system
* Units fight each other and rock paper scissors system is completely integrated
  + Flanking troops (if a troop is shot in the back) they take extra damage
* Players can decide which units fight which unit by clicking on the unit then the enemy unit
* Enemy AI uses tactics to fight for example, flanking and counter flanking, distractions, counter units, rushes/ turtling, scouting, directly fighting troops weak to the unit, or holding strategic areas of the map.
* More than one map
* More than one game type, conquest, and dominion (hold areas to gain points, first to x points wins)

### Assets

#### Custom built Blender Assets



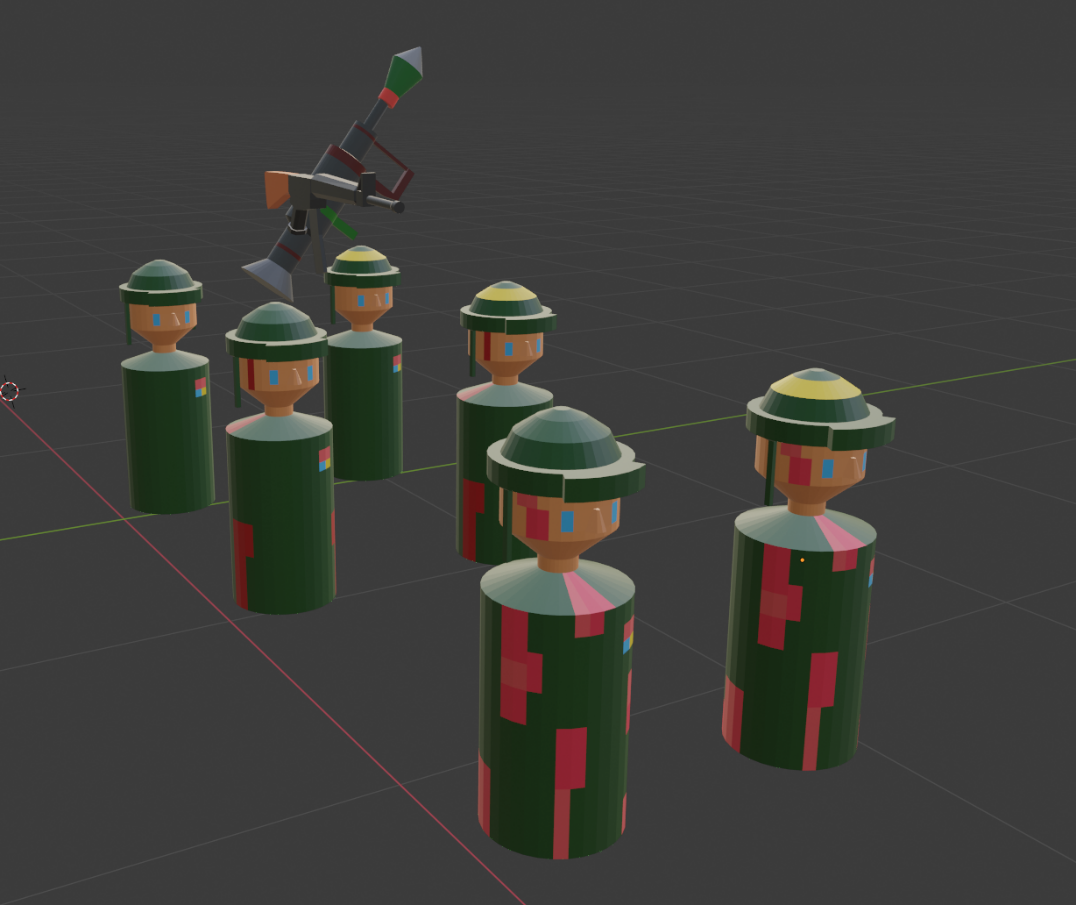
Home base



🡨reactor level 1 and 2

🡨level one buildings

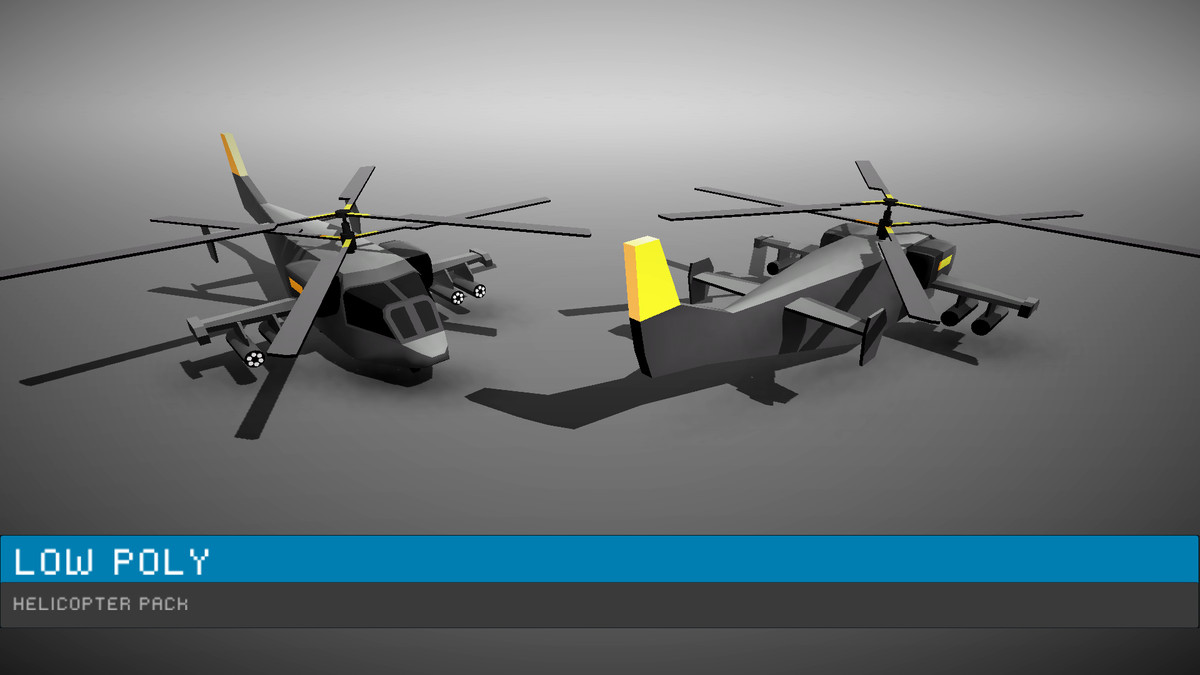
🡨level 2 buildings

solider models with decreasing health represented with bloodier models

gun and rocket launcher.

### Unity Store Assets

Unity Store assets were selected to save time in modelling. High quality low poly assets that were free to use and credit in projects were used to.



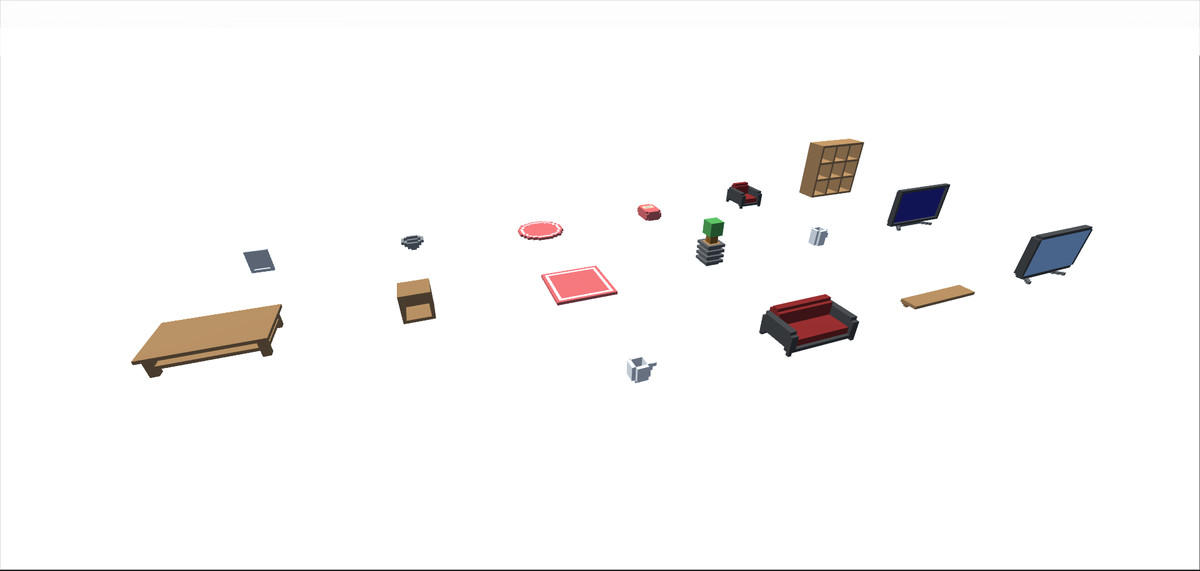
<https://assetstore.unity.com/packages/3d/vehicles/air/low-poly-helicopters-pack-free-121151>



<https://assetstore.unity.com/packages/3d/vehicles/land/cartoon-tank-free-165189>



<https://assetstore.unity.com/packages/3d/props/furniture/voxel-furniture-free-170365>



<https://assetstore.unity.com/packages/3d/props/furniture/too-many-items-living-room-props-129097>



<https://assetstore.unity.com/packages/3d/props/too-many-items-kitchen-props-127635>



<https://assetstore.unity.com/packages/3d/props/15-low-poly-models-202061>

## Appendix 2: Sprints and Retrospective tasks

**Sprint 0 - 26/11/2021**

Task -

* + Record ideas for upgrade system
    - Diagrams
    - Skill tree tables
  + Research different mechanics
    - Camera controls
    - Pathfinding
    - Unit controls
    - Modelling effectively
  + Start blender models
    - Solider
    - HQ
    - Buildings
* Retrospective
  + I was able to create a chess piece looking solider based on inspiration from a YouTube video I found during research. [Unity RTS - Movement and Pathfinding Groups](https://www.youtube.com/watch?v=4UKMa4JqTTY&t=389s).
  + I was able to model a rocket launcher using an image as reference. <https://pt.vecteezy.com/arte-vetorial/354194-icone-de-vetor-de-lancador> and use a previous assault rifle model I'd made to gear him up.
  + I created the buildings id identified. I planned to have 3 upgrades for the buildings, meaning 3 variants of each, but made the decision on scaling back to 2 variants to help with project scope.
  + I spent several hours researching videos I believed could help and guide when it came to make various mechanics in the game and added them to my unity help playlist on YouTube which I could refer back to when needed.

**Sprint 1 – 10/12/2021**

* Task -
  + Camera controls
    - WASD QE movement
  + Create UI for different buildings
    - Have this UI open when the building is clicked on
  + Create UI for each pad
    - Spawn the correct building on the correct pad
* Retrospective-
  + While getting rotation to work was an initial challenge, I was able to go through several iterations of camera movement and settled on a version I was happy with. The first issue had WASD and no rotation but the player was able to move directionally by have their cursor go nearer the screen edges; another had rotation but it was based on world angle vs the cameras therefore even if the camera was rotated 90 degrees, pressing W would still move north as opposed to east. The current version has rotation and directional.
  + Worthy of note is how I've chosen to focus on these core systems before moving onto combat in contrast to the Gantt chart, this is so I have a solid foundation to work from when I do get to making the AI.
  + Sprint was overrun (Christmas & driving up to Worcester) was able to get a partially functioning UI for choosing a building. 2 iterations of the UI design was manufactured but after losing progress and discovering bugs with the UI shapes a simple format was adopted.
  + Ui will animate onto the screen and animate away – code was also set up so it records which pad was chosen to build and which building was chosen to build.
    - solution for instantiating buildings
    - After this I will move onto path finding and spawning troops

**Sprint 2 – 26/01/2022**

* Task -
  + Create Namesh controls for units
  + Selected units are highlighted and can move to points
* Retrospective
  + Very slow sprint while getting to grips with other modules
  + Completed nav mesh control – units can move but very basic and needs work
  + Selected units receive a gold circle and lose it when they're unselected

**Sprint3 – 22/02/2022**

* Tasks -
  + Have barracks and supply pad do their functions
  + Model a simple tank and helicopter and map
  + Implement selection circle for mass section of units
* Retrospective
  + Barracks functions
    - Create soldiers
  + Supply pad functions,
    - adds supplies
  + Units turn to face enemies
    - Targeting system without using collision
      * Looks for targets in range
  + Selection circle activates gold circles but errors around removing units from selected units arrays

**Sprint 4 – 17/03/2022**

* Tasks -
  + Fix selection circle✓
  + Model a simple tank and helicopter and map
  + Ally troop movements – realistic movement to avoid friendly fire
* Retrospective
  + Troop AI – moves so they don’t shoot each other in the back
    - I had to create a custom AI using raycasts so the troop could analyse what's in front of them, if it's a team member and the team member is in combat they'll move to get an angle on the enemy
    - They shoot and bullets come out and collide with the enemy
    - They know where their closest enemy is and what's in front of them using raycasts, they look for targets every two seconds as to not do it every frame
  + The sphere selection accurately works and doesn’t miss any troops
    - Selecting the troops removes it from a parent array and I was getting bugs that when shift is pressed to bring up the circle it couldn’t go through an activate the select/deselect because it edits the array and when the for loops comes back the array has changed so it bugs, I made a mass remove from list method that resets the selection back to their unselected modes

**Sprint 5 – 05/04/2022**

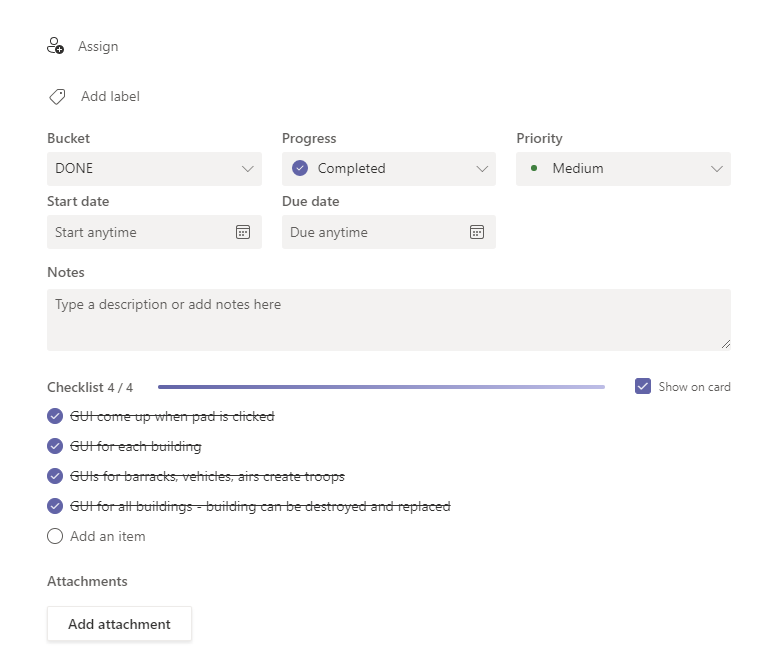
* Tasks -
  + Have troops hurt enemies
  + Find Unity store assets to replace homemade tanks and planes and terrain✓
  + Implement their mechanics using the finished friend infantry troop as a base
  + Create Questionnaire ✓
* Retrospective:
  + I was able to implement tank and helicopters to the friendly team
    - I used some free assets from the unity store and modelled a map using them
    - I found a good tank and helicopter model to use
  + I started the write up and plan to review as I write – this will increase time but be worth it
  + I created a questionnaire so I can get some feedback on what I have so far
    - Specifically to test the useability of the menus around creating troops and the resources ease of understanding

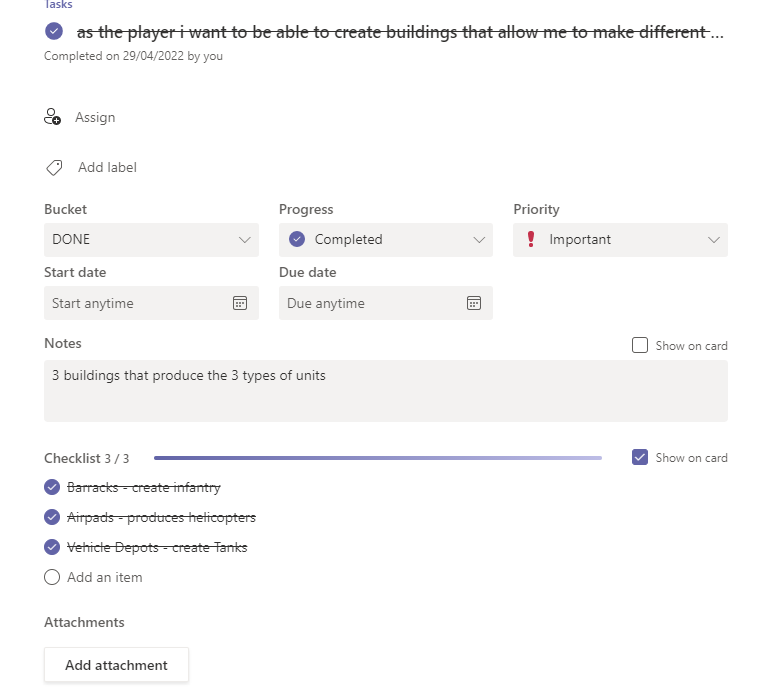
**Sprint 6 – 19/04/2022**

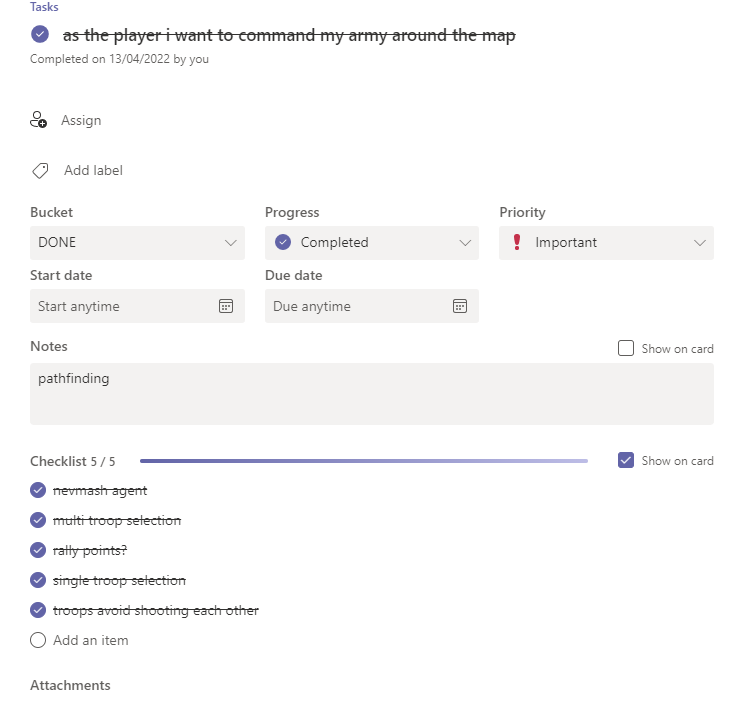
* Tasks -
  + Have troops hurt allies
  + Have enemies turn and shoot at player troops
  + Have enemies hunt and roam
  + Implement mechanics using the finished friend infantry troop as a base to  enemy troops
  + questionnaire and interview people about the mechanics
* Retrospective:
  + I was able to implement health and damage to enemy members and ally members so they can kill each other
  + I started the write up and plan to review as I write – this will increase time but be worth
    - 5 people were gathered to test the game - I received some useful feedback mainly on the useability of the menus and action feedback and information display

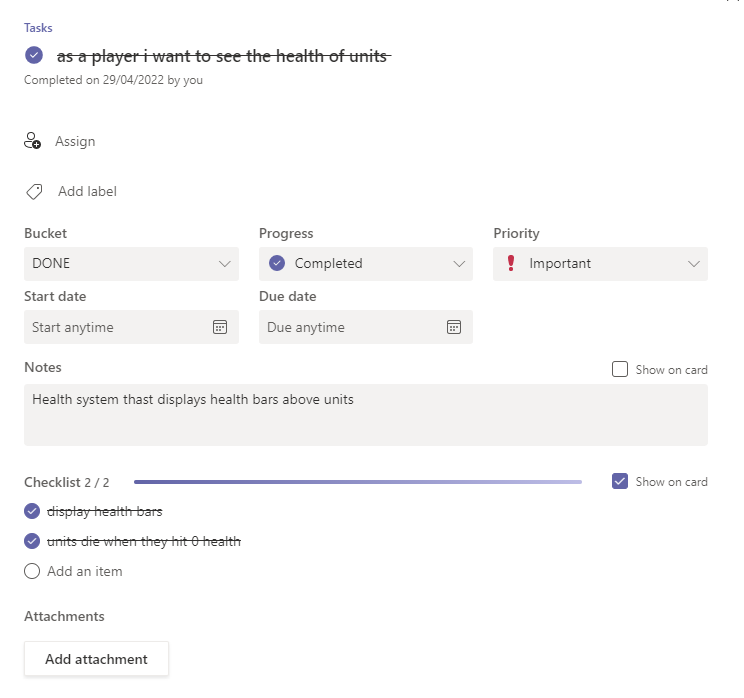
**Sprint 7 – 03/05/2022**

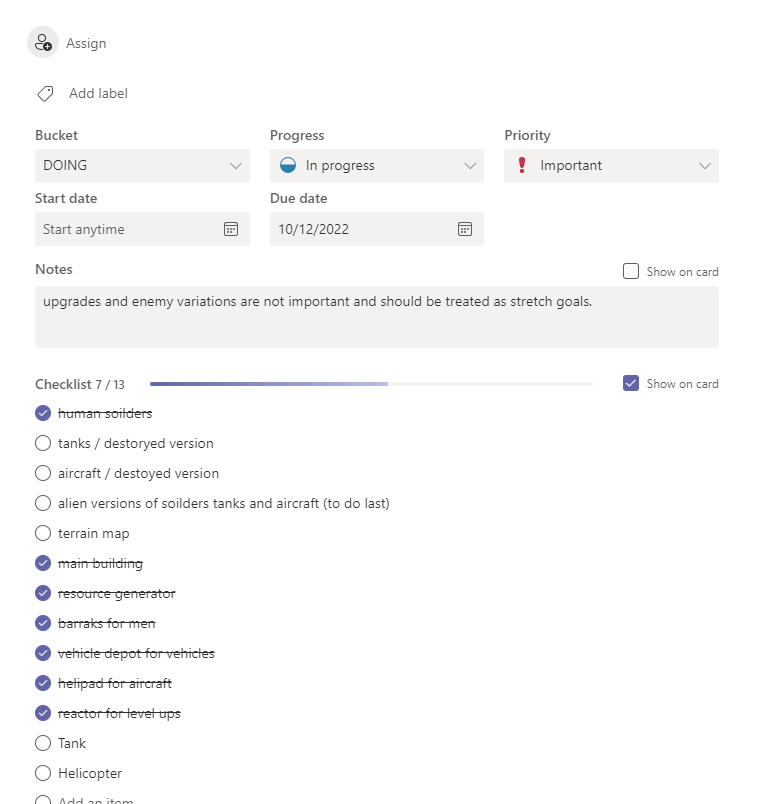
* Tasks -
  + Have AI spawn at a balanced rate
  + Ai produces counter troops
  + Enemy base and player base lose health when hit
  + End game and main menu
  + Destroyable buildings
  + Testing feedback implementation ✓
  + Itch Page✓
* Retrospective
  + I had to re angle the targeting of ground units as I discovered a bug where units in front of a unit prevent it from being able to see if they never saw it and turned to look. This was solved by having them always look towards the closest target if it's in range
  + Enemy troops spawn a starter army and start producing troops at a fixed rate depending on the what the player makes by keeping count of their troops and producing the counter to the most prevalent unit
  + Building with UI were destroyable first but another approach for supply pads and reactors – an icon appears above them which can be clicked to destroy them
    - I had to split existing scripts into two so which pad and which building could be passed from the pad to the building to the UI as the UI for each building is shared
  + Enemy base is a red version of the player base
    - Both bases can be targeted and hurt
  + I received feedback to rebalance the units as one unit can take over 30 of the counter – this was because damage was at a 5:100 ratio now it is now a 1:2
  + The main menu was created and is very simple
  + The game has reached a playable state so I made an itch page and uploaded it
    - I followed this by posting on my twitter about the new release

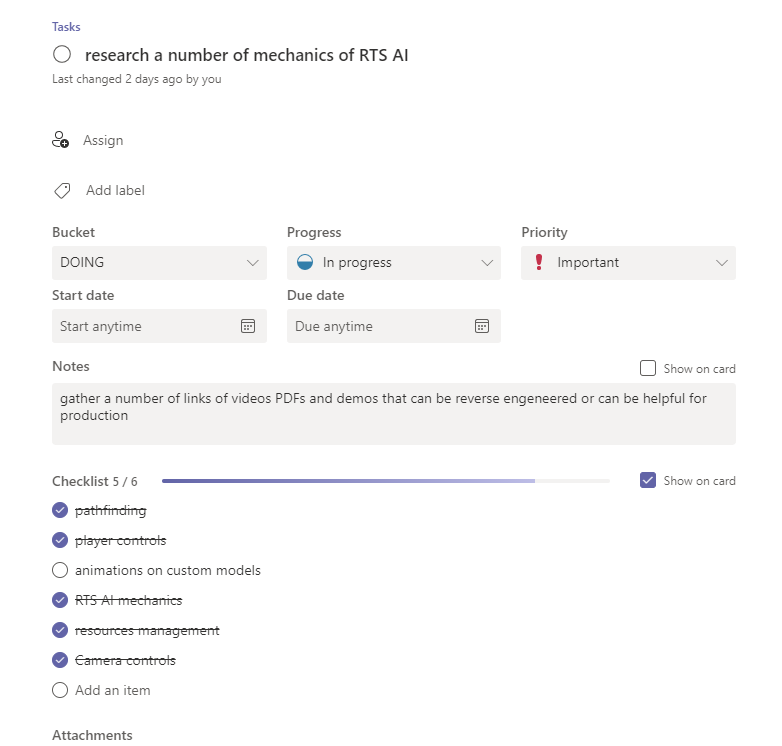






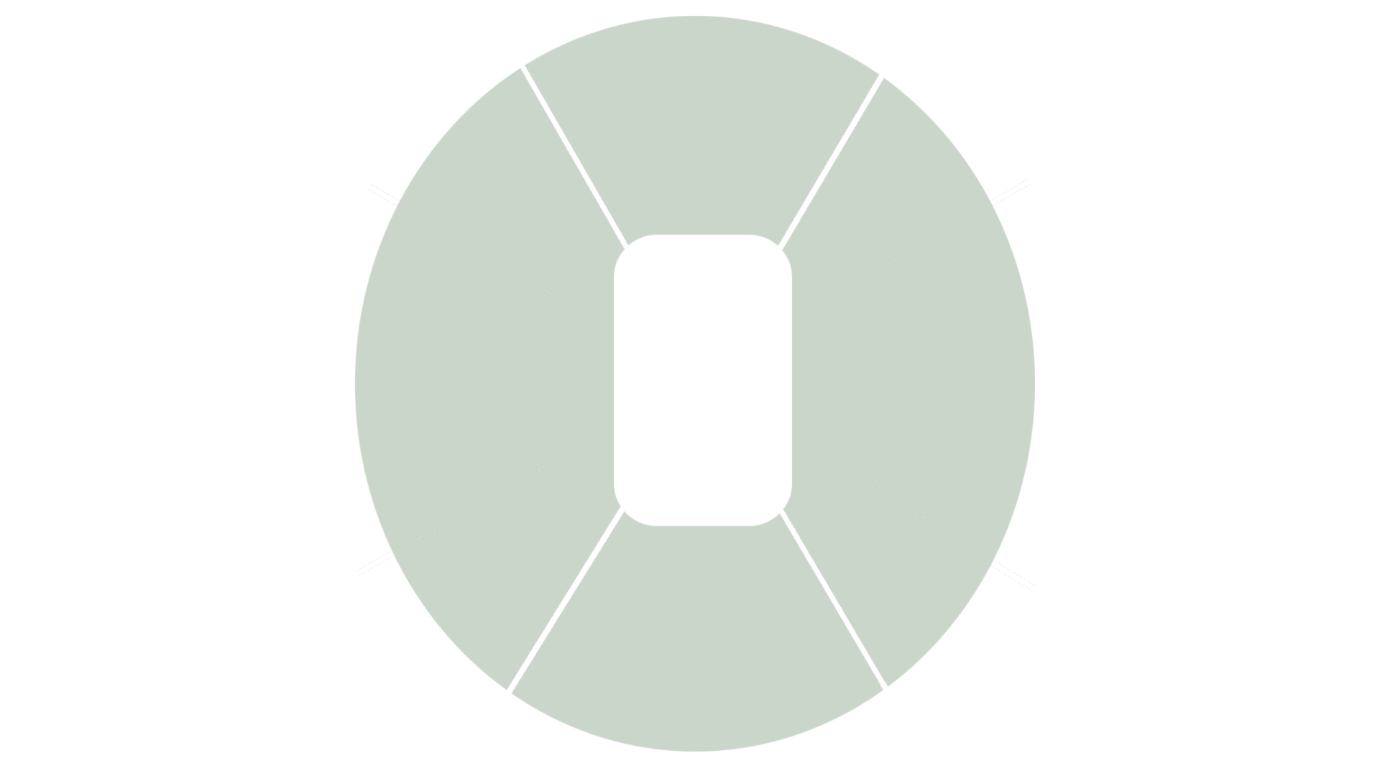






## Appendix 3: Scrapped UI designs







## Appendix 4: questions and results

<https://forms.gle/gn8ihnruUZ1Aq1fq9>

