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 my girlfriend Leah Humphries for always being there for me through each year of this course especially when times were low and giving me the strength to keep going and my parents at home who always provided me with support and guidance.

Also, thanks go 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

Table of Contents

[Acknowledgements 2](#_Toc103347619)

[Abstract 2](#_Toc103347620)

[Word Count 3](#_Toc103347621)

[Links 3](#_Toc103347622)

[Introduction 3](#_Toc103347623)

[Project Definition 3](#_Toc103347624)

[Purpose 4](#_Toc103347625)

[Objectives & Functional Requirements 4](#_Toc103347626)

[Minimum viable product 4](#_Toc103347627)

[Minimum awesome product 5](#_Toc103347628)

[Method of Approach 5](#_Toc103347629)

[Project Management 7](#_Toc103347630)

[Microsoft Teams 7](#_Toc103347631)

[Testing 8](#_Toc103347632)

[Version Control 8](#_Toc103347633)

[Technologies 9](#_Toc103347634)

[Visual Studio, Unity and C# 9](#_Toc103347635)

[Navigation mesh 9](#_Toc103347636)

[Ray casts 10](#_Toc103347637)

[Legal, Social, Ethical and Professional 10](#_Toc103347638)

[Legal 10](#_Toc103347639)

[Social 11](#_Toc103347640)

[Ethics 11](#_Toc103347641)

[Professional 11](#_Toc103347642)

[Implementation 11](#_Toc103347643)

[Sprint 0: 26/11/2021 – 10/12/2021 12](#_Toc103347644)

[Overview 12](#_Toc103347645)

[End of Project Report 12](#_Toc103347646)

[Project Post-Mortem 12](#_Toc103347647)

[Conclusion 12](#_Toc103347648)

[Bibliography 12](#_Toc103347649)

[Appendices 13](#_Toc103347650)

# Word Count

# Links

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

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

# 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 as well as Comp3013 where I was project manager and 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 id 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 that includes hall marks of industry standard criteria that employers would 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 as well as to feel fulfilled that my career at university lead to this, and I can be proud of it.

## 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, built primarily using Unity’s navmesh AI system, raycast networks and lightweight custom built sorting algorithms. The game features some custom-built assets using Blender and free assets imported from the Unity asset store that creates an overarching ‘toy soldiers’ aesthetic to the game realising 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 counter other unit types, the player must manage their bases 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 win.

## Purpose

on a personal level, 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, 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 as 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 as to begin the project sooner and without delay from the documents; they also stress 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 projects 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… [insert appendix number]

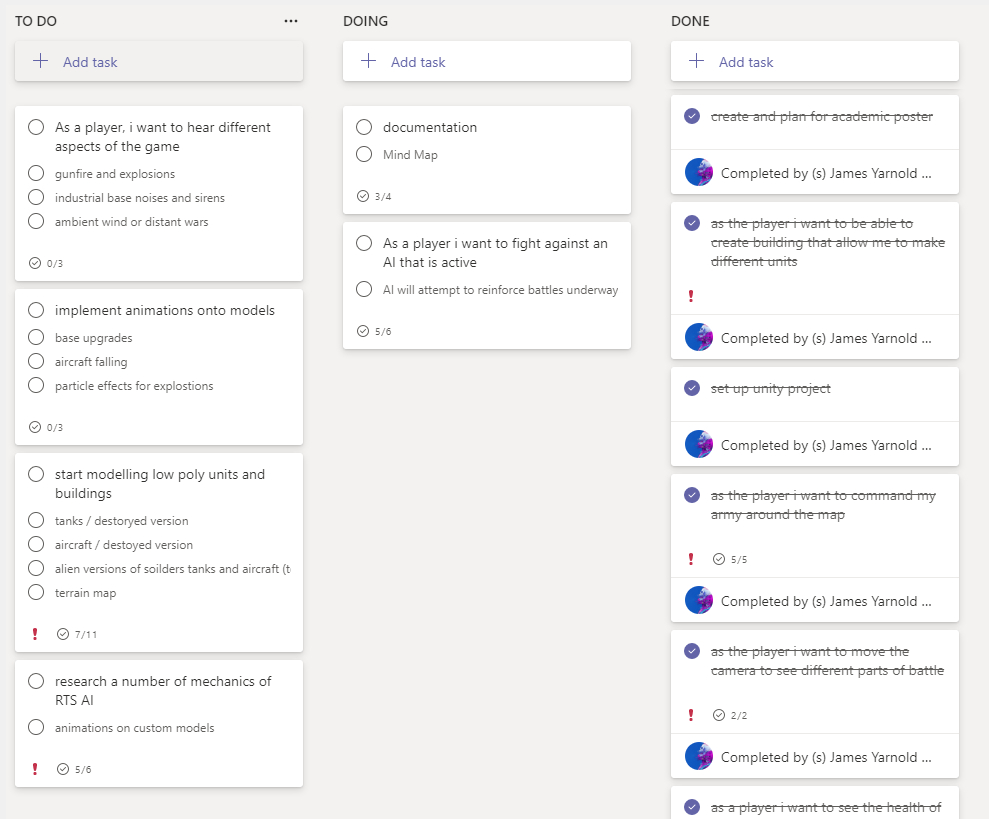
# Method of Approach

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, 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 could refocus upon the sprint ending.

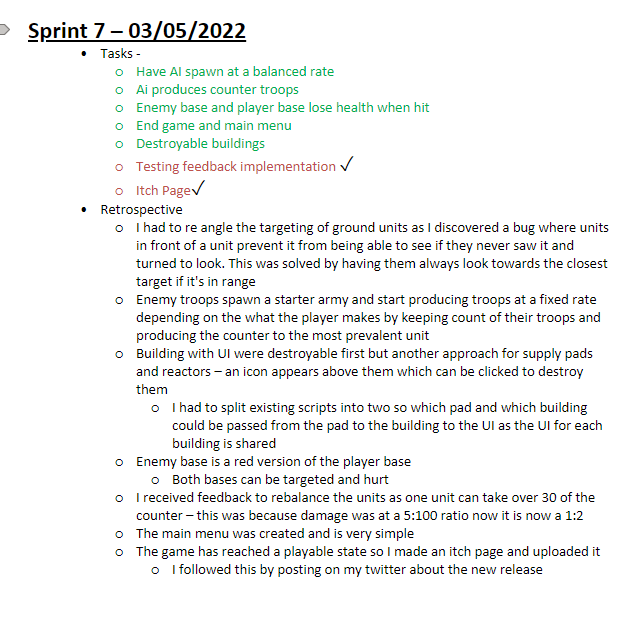
To help with the flow of tasks, a Kanban style board was used to keep track of large “parent” tasks which were defined in user stories. By splitting these down into smaller tasks that could be picked and chosen as and when they were necessary for development, they formed the basis of checklists that 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, helping the natural flow between the end of one sprint and the start of another.

This helped more specifically 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 while 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 with the board providing over arcing larger tasks and their break downs while the retrospective providing understanding on the progress of smaller tasks as well as 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: [insert appendix number].



*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, they are centralised into the GDD as per manifesto guidelines and derived the projects backlog and 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, but the aim was to keep the amount of these down to centralise progression and development into apps that specialised in the project management and development field as well as having connections to past experience at university, so 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 that 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 the project 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 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 the errors id found.

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 that 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.

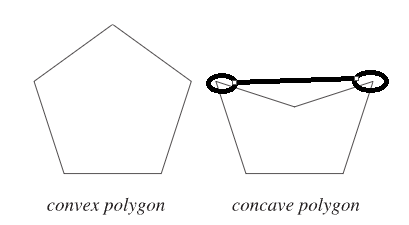
## Technologies

### 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 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 which 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 the 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 the local level.

Within the Unity libraries, methods exist for manipulating different important factors of the component attached to the agent called the nav mesh agent component. Effects such as its navigation abilities, steering and obstacle avoidance can be changed like this. The project uses the methods associated with Set Destination and stopping Distance for the majority of its manipulation to 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 bool 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 the components located in the “NavMeshComponents” project 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, and 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 in the name of criticism and it 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, which on the surface is a fitting idea for a game, but as part of the target audience is the younger person 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, page on the matter on the Itch.io page. The fact of 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 on there and so takes the top centre spot so its seen and read easier. 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 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 x]

## 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 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

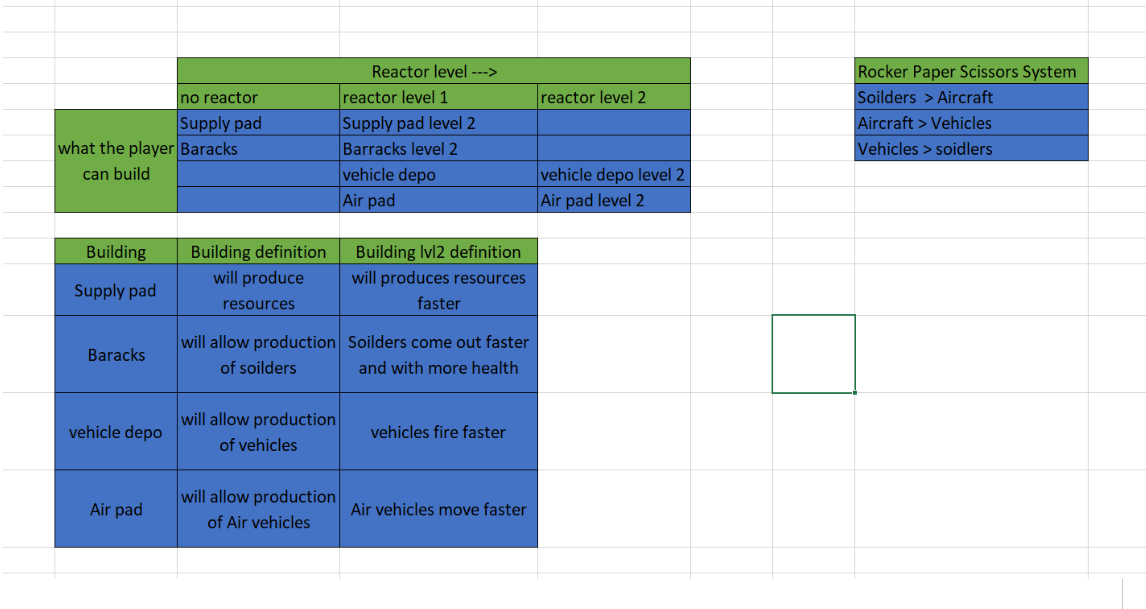
* Record ideas for upgrade system
  + Diagrams
  + Skill tree tables
* Research different mechanics
  + Camera controls
  + Pathfinding
  + Unit controls
  + Modelling effectively
* Start blender models
  + Solider
  + HQ
  + Buildings

#### 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 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 players 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.

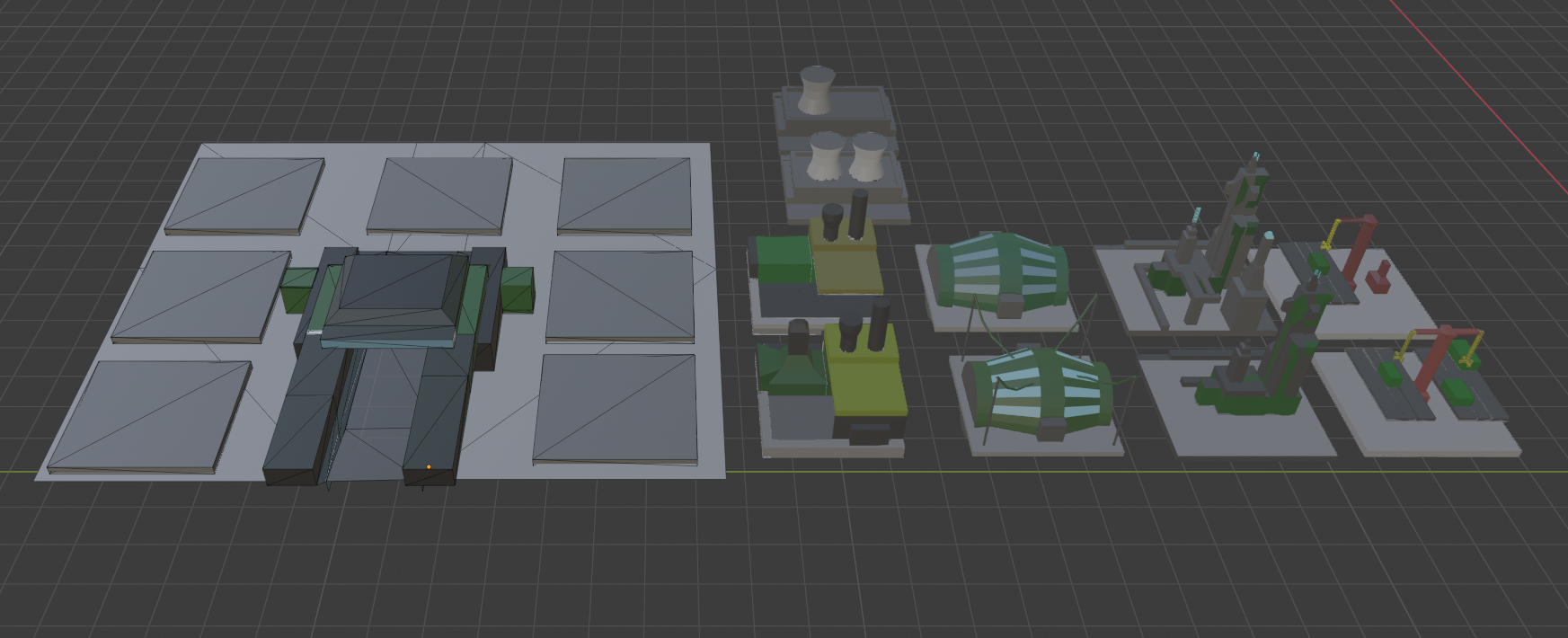
At the time of conception, when the features were in this planning stage, upgrades to each building were to be planned for and implemented at a later stage as stretch goals once the foundation of at least the first level buildings had been applied. Figure four shows the conception of the functionality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



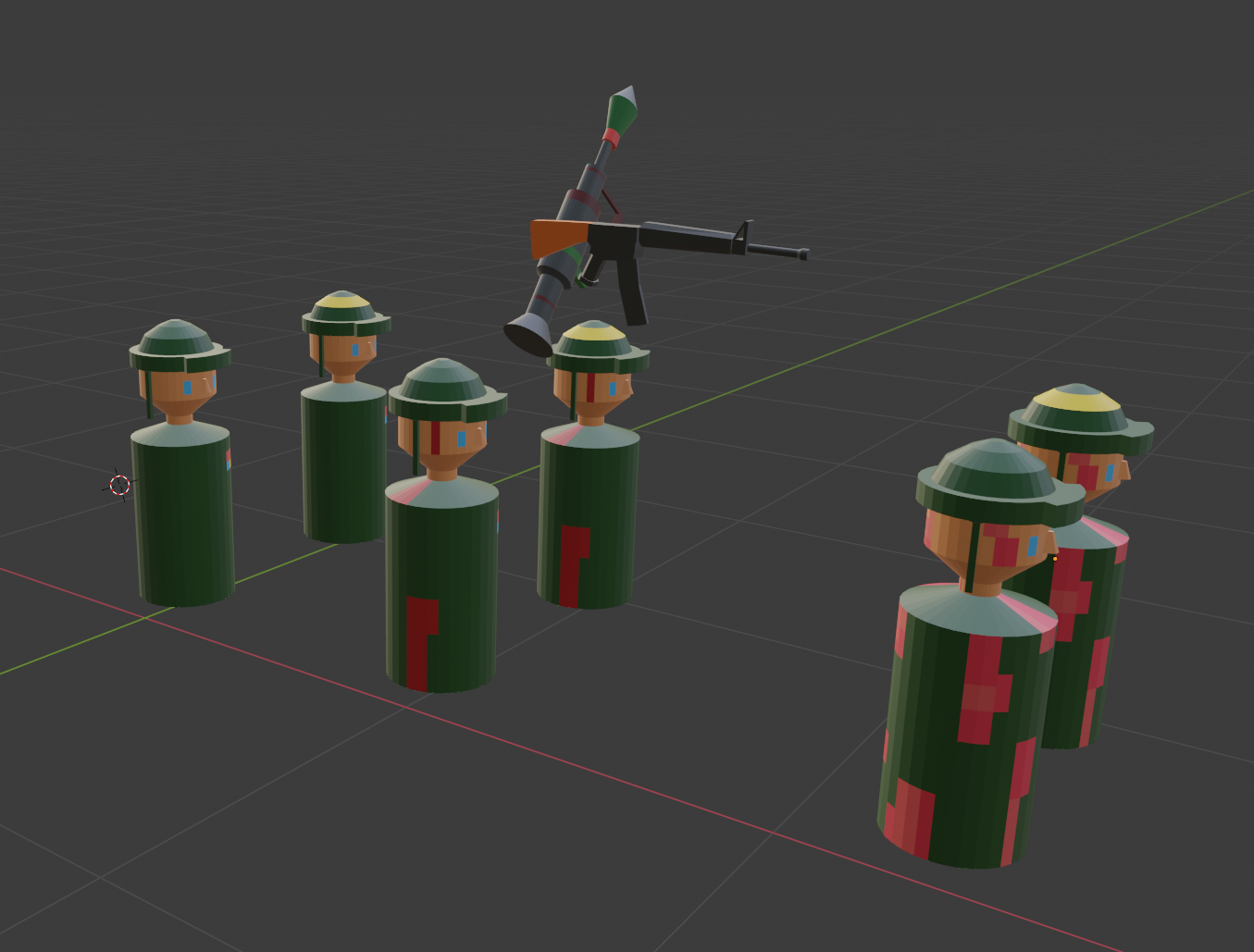
*Figure [4] documentation created to plan the system of building and upgrades*

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 a strategy 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 solider was completed (final versions shown in figure five and six). 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]

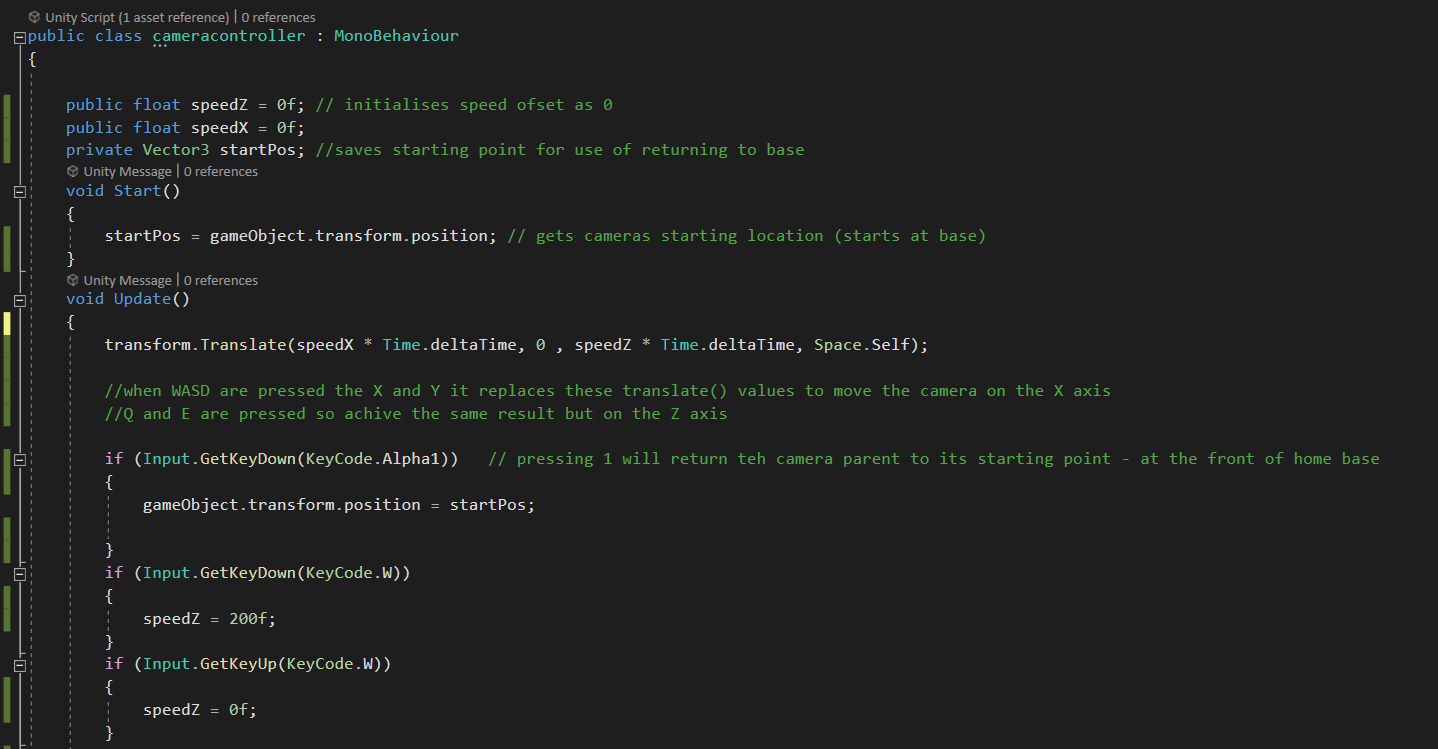
#### Tasks

* 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 x) 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 that was selected. 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 include of weather or not is 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

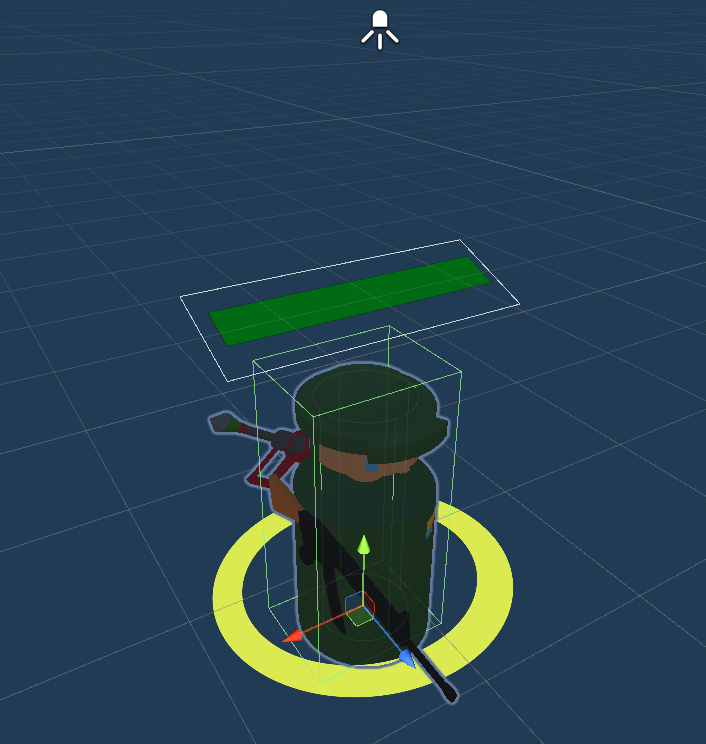
* Create navigation mesh controls for units
* Selected units are highlighted and can move to points

#### Implementation

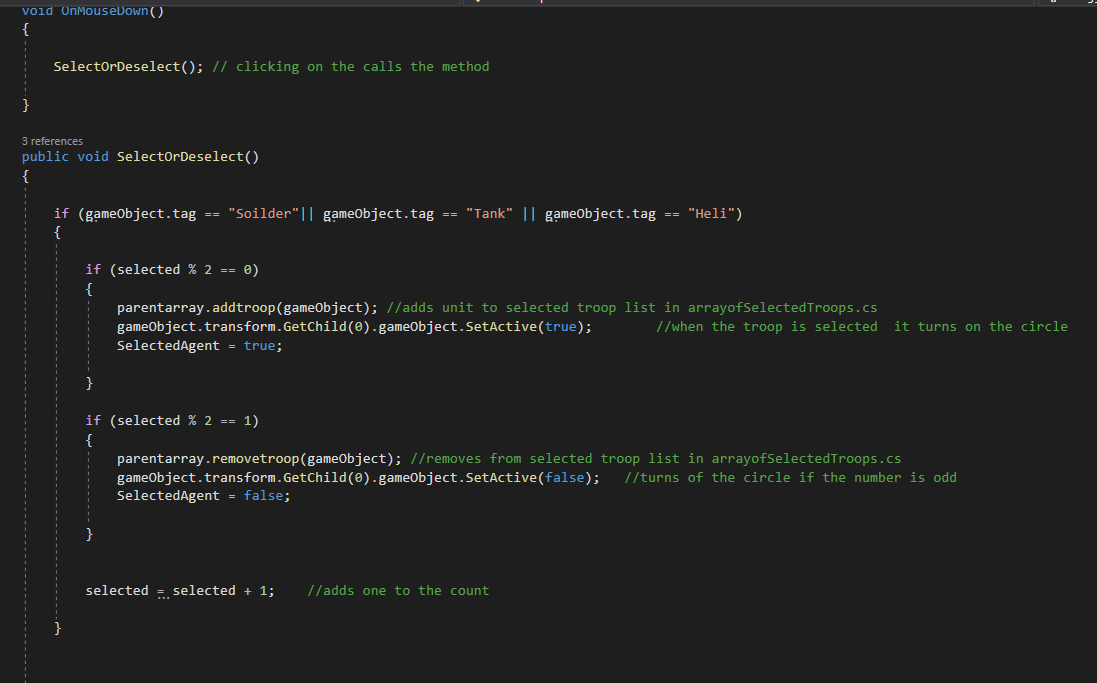
The troop navigation was implemented via the navmesh agent component functionality that unity has embedded into it. The nav mesh agent component needs something to detect walkable surfaces, and while the option to bake surfaces into these walkable areas using object meshes is available on the standard 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 uses 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 script*

Once the troop movement function was successfully added, a crucial bug developed when testing with more than a few units, and even less that 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: 26/01/2022 – 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

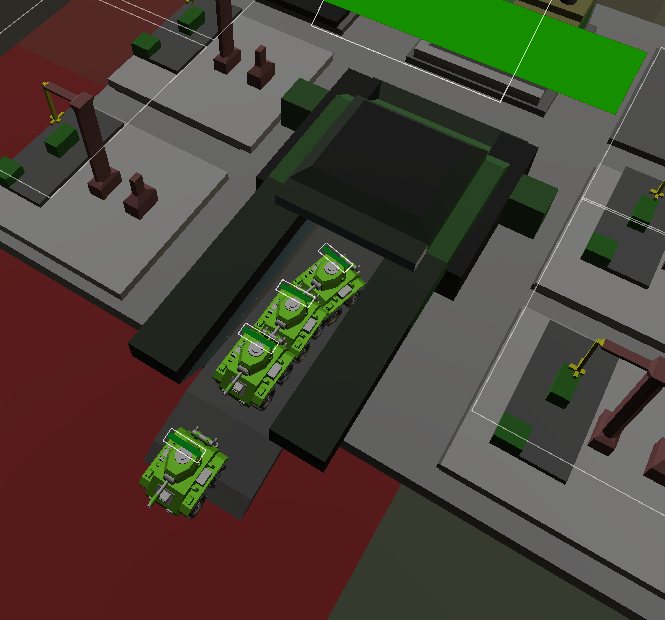
#### Tasks

* 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 this sense that 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 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 the troops to maintain the fast-paced environment that was being moulded. 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 with the mouse cursor. Using a similar raycast system to the moving of troops, a navmesh agent with a collider was introduced that would be updated to the mouse position when it is active, it has the tag “Selector” and units would register this tag on a collision to activate its methods to select or deselect it.

# End of Project Report

# Project Post-Mortem

# The Future

# Conclusion

# 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