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 goes 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](#_Toc103177957)

[Abstract 2](#_Toc103177958)

[Word Count 3](#_Toc103177959)

[Code 3](#_Toc103177960)

[Introduction 3](#_Toc103177961)

[Project Definition 3](#_Toc103177962)

[Purpose 4](#_Toc103177963)

[Objectives & Functional Requirements 4](#_Toc103177964)

[Minimum viable product 4](#_Toc103177965)

[Minimum awesome product 4](#_Toc103177966)

[Method of Approach 5](#_Toc103177967)

[Project Management 6](#_Toc103177968)

[Microsoft Teams 6](#_Toc103177969)

[Testing 6](#_Toc103177970)

[Version Control 7](#_Toc103177971)

[Technologies 7](#_Toc103177972)

[Visual Studio, Unity and C# 7](#_Toc103177973)

[Legal, Social, Ethical and Professional 8](#_Toc103177974)

[Implementation 8](#_Toc103177975)

[Sprints 0 - 12 8](#_Toc103177976)

[End of Project Report 8](#_Toc103177977)

[Project Post-Mortem 8](#_Toc103177978)

[Conclusion 8](#_Toc103177979)

[Bibliography 9](#_Toc103177980)

[Appendices 9](#_Toc103177981)

# Word Count

# Code

(*GitHub link*)

# 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 amalgamate 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]

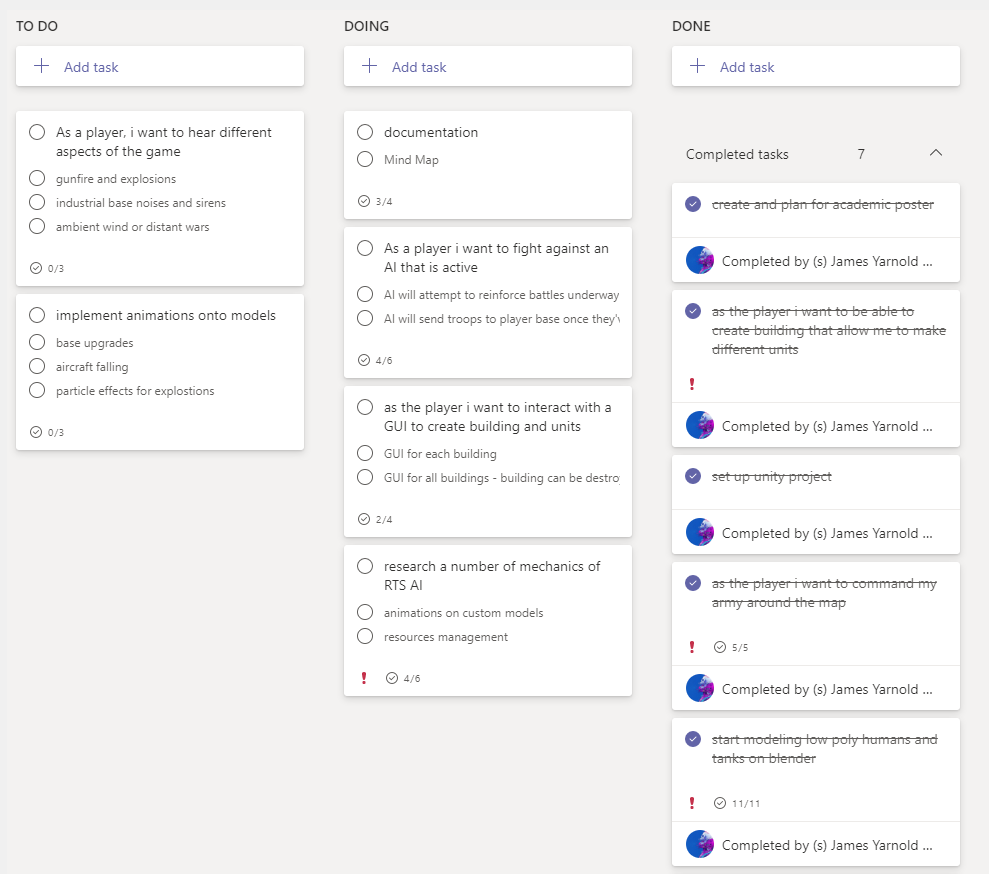
# Method of Approach

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.

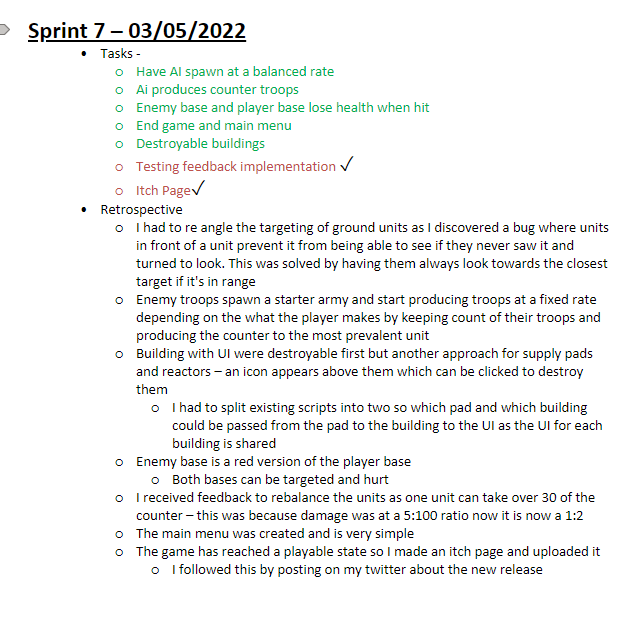
To help with the flow of tasks, a Kanban style board was used to keep track of large “parent” tasks which were defined in user stories. By splitting these down into smaller tasks that could be picked and chosen as and when they were necessary for development, they formed the basis of checklists that 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 plans for the next sprint could be assessed with the information of the last sprint in plain text and new tasks could be followed more accurately with the qualitative information presented in them that judged, 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 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 retrospective these over arcing tasks from the board are in green and errors or added tasks during the sprint are in red, these tasks were ticked if they were completed when the sprint ends, if they weren’t the task was moved into the next task list. 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.

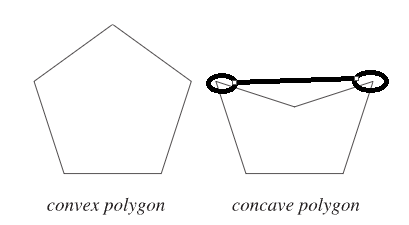
## Technologies

### Visual Studio, Unity and C#

Unity is the game engine the project was built on and it utilises two intrinsic systems built into Unity to achieve mechanics around movement and around the AI of player-controlled troops and AI controlled troops.

#### 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 attached to the agent such as its navigation abilities, steering and obstacle avoidance. The project uses “.SetDestination()”, and “.stoppingDistance()” 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.

#### 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

Credited assets

Appropriate licences

PEGI 16

# Implementation

## Sprints 0 - 12

# End of Project Report

# Project Post-Mortem

# Conclusion

# Bibliography

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

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