

Computer Games Development SE607

Technical Design Document

Year IV

|  |
| --- |
| [Leo Chen]  [C00250593]  [26/04/2023] |

**DECLARATION**

﻿

**Work submitted for assessment which does not include this declaration will not be assessed.**

* I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.
* I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.
* I have provided a complete bibliography of all works and sources used in the preparation of this submission.
* I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: (Printed) Leo Chen

Student Number(s): C00250593

Signature(s): Leo Chen

Date: 26/04/2023

--------------------------------------------------------------------------------------------------------

**Please note:**

1. \*Individual declaration is required by each student for joint projects.
2. Where projects are submitted electronically, students are required to submit their student number.
3. The Institute regulations on plagiarism are set out in Section 10 of Examination and Assessment Regulations published each year in the Student Handbook.

Contents

[Packages and Libraries used in Unity 3](#_Toc133427295)

[Artificial Intelligence 4](#_Toc133427296)

[Neural networks 4](#_Toc133427297)

[Unity’s Machine Learning Agents 5](#_Toc133427298)

[Finite State Machines 6](#_Toc133427299)

[Behaviour Trees 7](#_Toc133427300)

[Formations and Grouping 7](#_Toc133427301)

[Base Units 8](#_Toc133427302)

[Navigation 8](#_Toc133427303)

[Field of View 8](#_Toc133427304)

[Shooting 8](#_Toc133427305)

[Control Points and Resource Points 9](#_Toc133427306)

[User Interface 9](#_Toc133427307)

[RTS Camera 10](#_Toc133427308)

[Unit Spawning 10](#_Toc133427309)

[References 11](#_Toc133427310)

Technical Design

# Packages and Libraries used in Unity

|  |  |
| --- | --- |
|  |  |
| Navmesh plus | Custom pathfinding package for 2D  https://github.com/h8man/NavMeshPlus |
| ML Agents | Unity’s Machine Learning Agents toolkit  (Unity ML-Agents Toolkit 2023) |
|  |  |
|  |  |
|  |  |

# Artificial Intelligence

## Neural networks

A neural network, also known as artificial neural networks are a subset of machine learning which attempts to train itself through the input of data. With said data being used to make decisions on the type of actions it can conduct.(What are Neural Networks? | IBM n.d.)

Our neural network is meant to take a set amount of inputs in order to use one of three actions as its output:

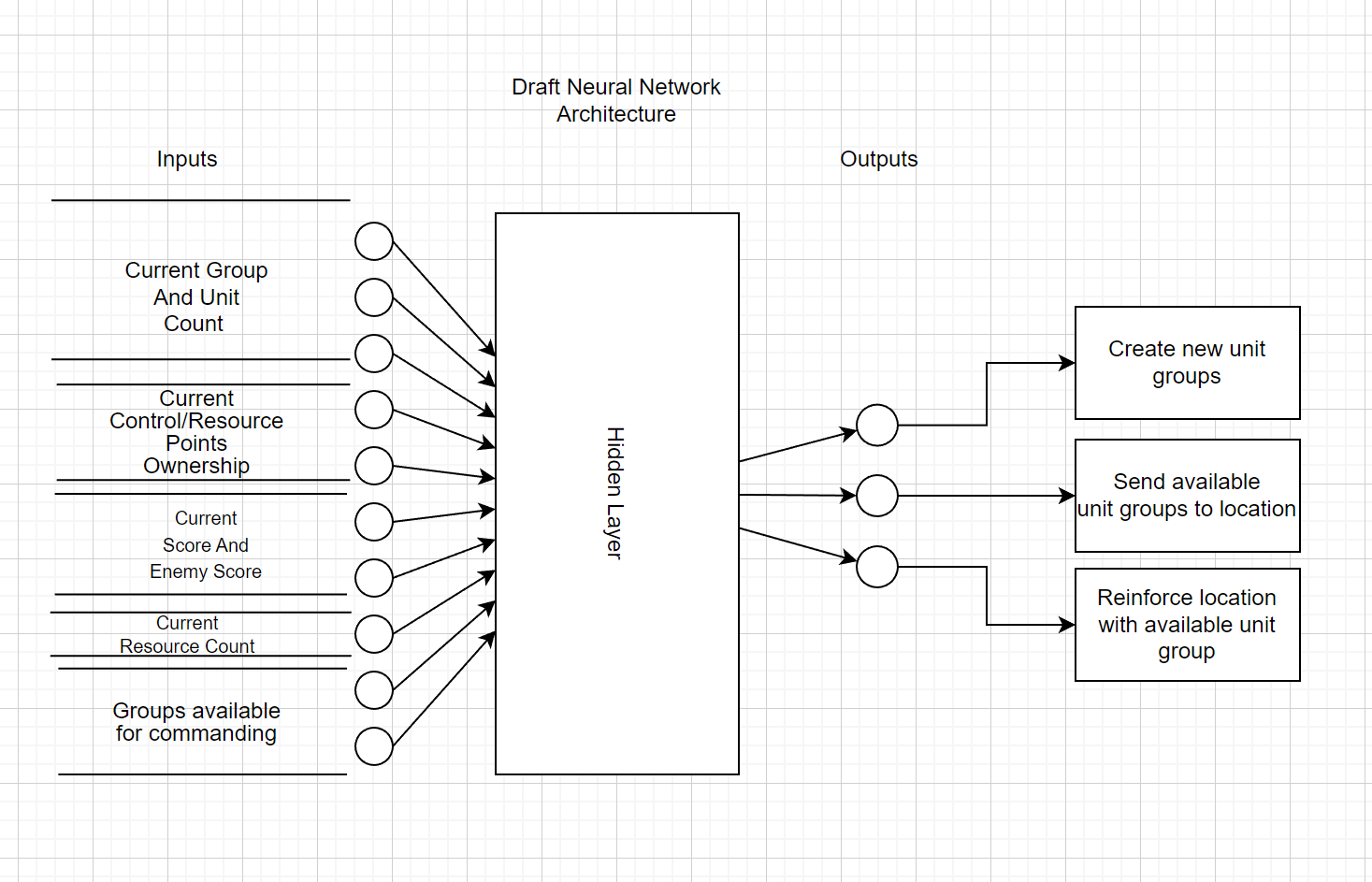


Fig. Proposed neural network structure

In this diagram, the inputs for the RTS consists of what the AI needs to know in order to produce an output. The AI requires the need to know what is available to it at all times when there is a need for deciding on what actions to take.

**Inputs:**

* Current group and unit count
  + The main purpose of this set of inputs is to tell the AI how many units and grouped units it has on the field.
* Current Control/Resource Points Ownership
  + Tracking the control/resource points of both itself and the opponent is needed for the tactical choice of moving groups to take or hold those objectives.
* Current score and current enemy score
  + Having both the scores tracked and fed into the neural network weighs on their decision to affect the former point. Should the AI priorities the resource points for more units to produce, or should it prefer control points that generate score?
* Current resource count
  + Similar to the above point, the resource count is meant to keep track of how much funding the player or AI has to produce units.

**Outputs**:

* Create new unit group
  + This action has the AI create a new set of units to immediately group up. The groups themselves are what the AI commands so it doesn’t need to manage individual units.
* Send available unit groups to a location
  + For this action as it implies, send an available group of units to a destination of the AI’s choosing. Usually towards a strategic objective.
* Reinforce location with available unit group
  + The action is meant for reinforcing a position with additional troops for the AI to cement its control over it.

## Unity’s Machine Learning Agents

The machine learning-agents toolkit (Unity ML-Agents Toolkit 2023) designed for both research and game development is one of the options for having it as an opponent for player in the game.

Similar to the actions listed in the neural network diagram, the AI requires several inputs in order to execute actions that can be utilised in game. For their observations they required a specific dataset to be utilised.

* Current friendly unit count.
* Current friendly group count.
* Location of the selector.
* AI Camera Location.
* Location of all control/resource points
* Team affiliation of said control/resource points
* Current resource count
* Current Score count

Their actions on the other hand will depend on the training data fed to it. Using imitation learning, I would need to modify the selection method on selecting units to be using a different control scheme to the one a player would use. Instead of the typical drag selection, they would get a circular area selection that they would move to a unit and select them for use.

## Finite State Machines

Though the main focus of the game were on neural networks and machine learning for the AI. A simple very basic finite state machine was required to test the game’s features. In the end I had wanted the machine learning agent to have faced a slightly more advanced version of the finite state machine I had created during tested.

For the loop of the FSM, it would, in similar fashion to the player, select all units within a predefined area and move them towards a rallying point. After a certain amount of time has passed, they would send the selected group of units towards one of the capture points and repeat the process.

The implementation of the Finite State Machine is very basic. Having only three states in total to act on was serviceable for testing alone. Though I had wished to have had a more expanded to AI, the main focus was on the neural network and machine learning algorithms.



Fig. Basic loop of a Finite State Machine

## Behaviour Trees

Another artificial intelligence I would have liked to implement would have been a behaviour tree for an opponent. The architecture for one is there but the implementation of it never came to pass. Following another simple implementation of it, it would have been a more advanced version of the finite state machine’s actions. Instead of three simple actions it would have, the expanded AI with the behaviour tree would decide in real time what choices to make.

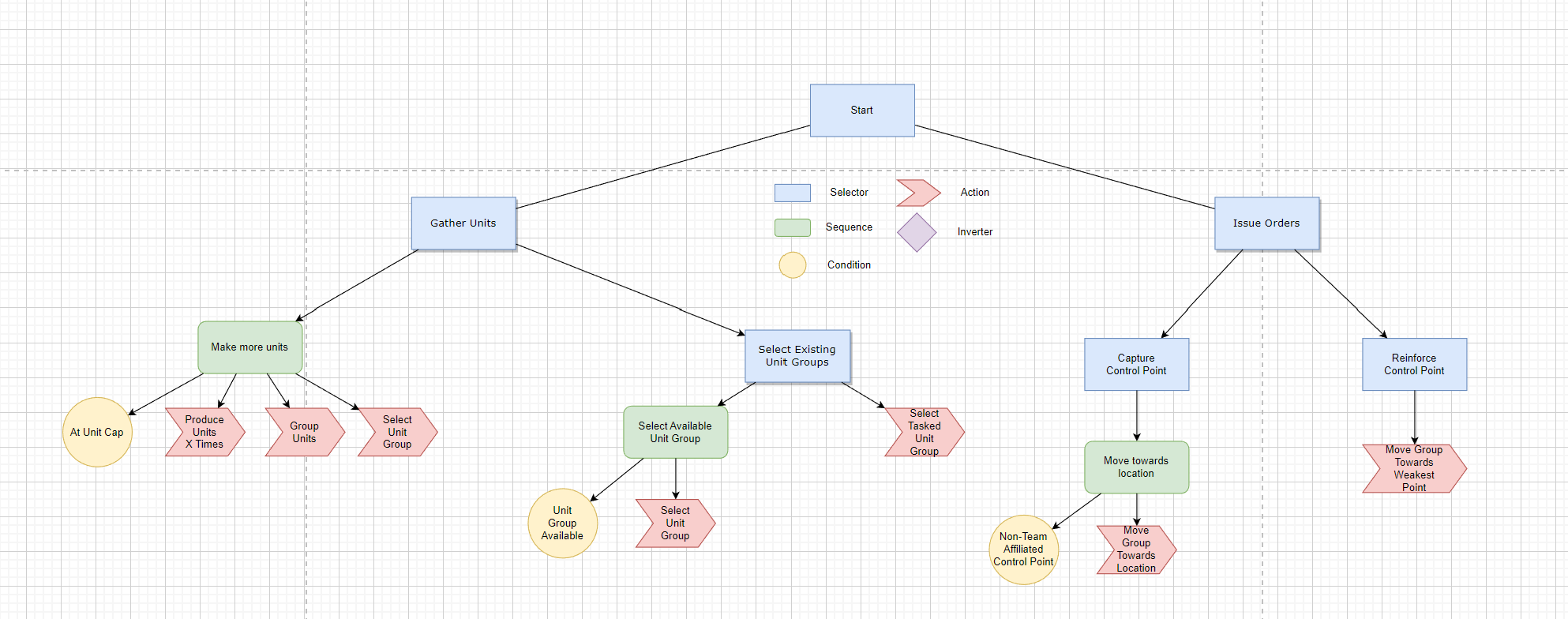


Fig. Proposed Behaviour Tree Structure

The idea for the behaviour tree is meant to use the data it has to act on. Going through the nodes, there are two main choices it has. It can either produce more units or send a group of units towards a location. Similar to the other AIs in terms of actions but with the behaviour tree, it can be expanded upon. The current architecture used for the behaviour tree was based on this tutorial (Stacey 2020).

# Formations and Grouping

Based on EezehDev’s Co-ordinated Formation movement code (Moor 2023), upon a grouping of a selection of units. The game spawns a group leader prefab that creates formation points for the units to go towards. It is dynamically adjusted based on the number of units in the selection. If a unit were to die then the formation shrinks and expands if more units are added to it.

Currently there is only the line formation available to the player. Though more types of formations were planned, due to time constraints, only one was available for use.

# Base Units

The base unit is the basis of all derivatives of units. Currently there is the trooper class which inherits from it. Due to the need for customizing the unit’s navigation, most of the code is overridden to make it work the way I needed it to. It is broken into three main components that make up the game object.

## Navigation

Rotation, movement, and speed is augmented in a way that overrides most of the original navigation mesh’s work. It’s replaced with custom code that is tailored towards moving the unit towards a location. Once it takes a destination it simply moves in the direction and uses nav-mesh’s A star pathfinding to get towards its location.

## Field of View

Its field of view is taken from a tutorial (Field of view visualisation (E01) 2015) that showcases what it can hit and what it can’t. Though the tutorial is made for a 3D game, I had to adapt some of the code to work in a 2D environment. It would scan for targets and shoot at the closest one. For the visual aspects of the field of view, it uses several ray casts that shoot off in all directions. Between two rays will have a mesh that dynamically shapes with obstacles around it.

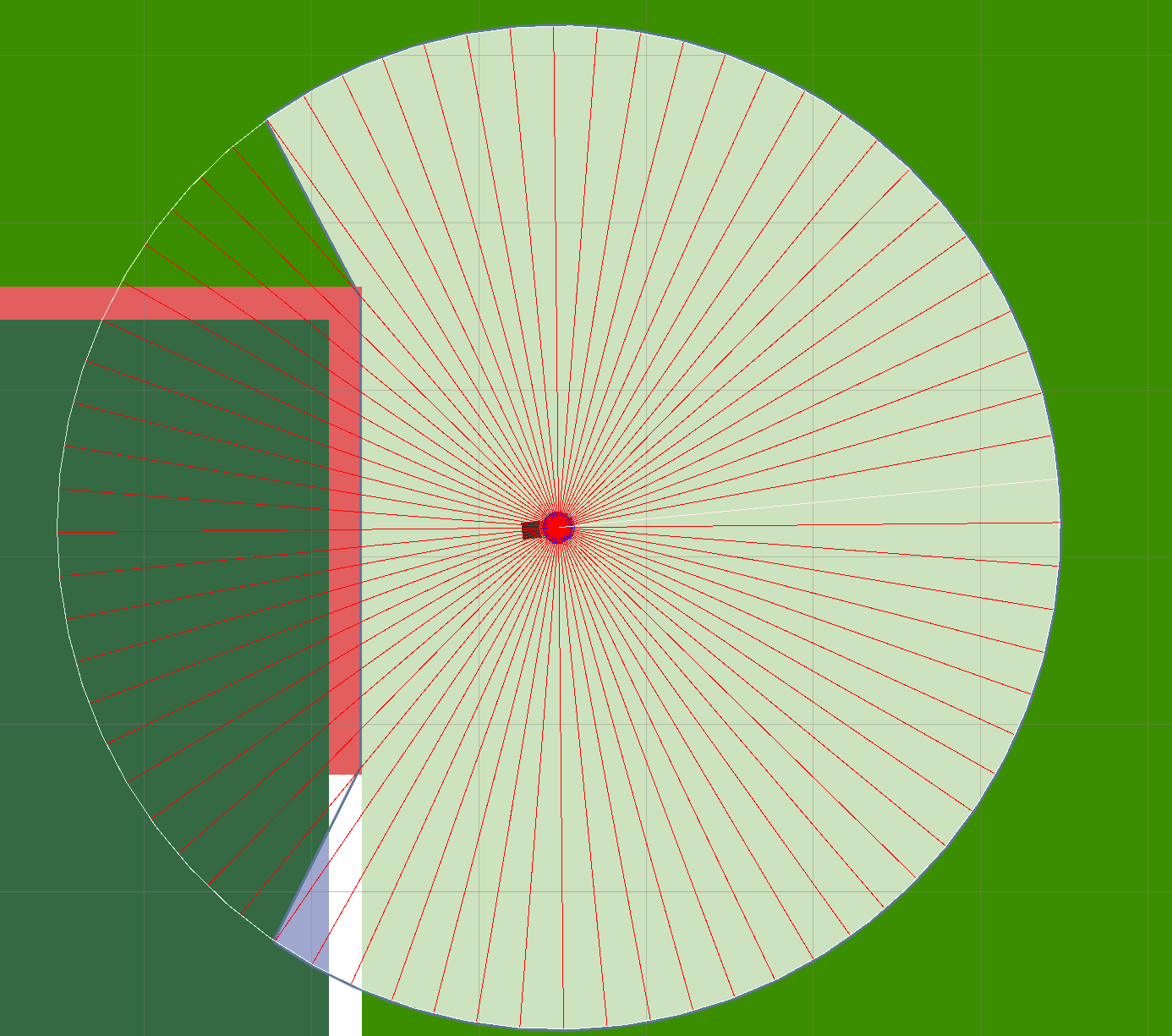


Fig. Filter mask and field of view

## Shooting

The unit’s shooting code is tailored for the trooper unit where it fires a simple hit-scan and draws a line from the shooter towards the target. Hitting another unit even if it’s your own unit will damage it. Neither an oversight nor a bug.

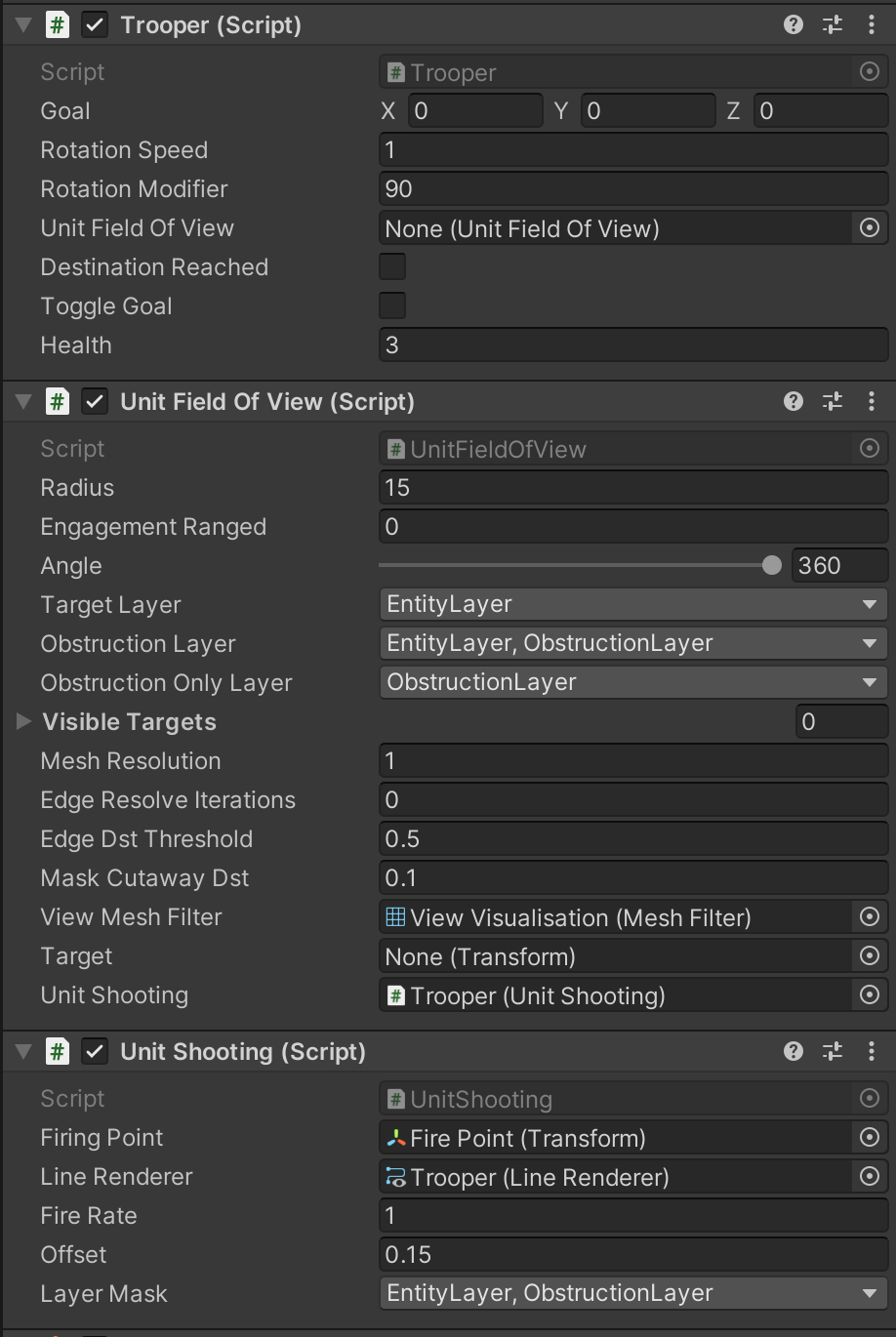


Fig. Trooper's Scripts

# Control Points and Resource Points

The main objective of both teams is to capture and hold these objectives to generate either score or resources. Again, taking elements of it from a tutorial, I had to modify it so that the code would work the way I needed it to. Having the points be either circles or squares to distinguish their role they had.

Logic for the capture points required some thought put into it. Various scenarios would have to be consider for it to be fully functional. Although as it currently stands, they fulfill their function, there’s still more polish I would have liked to have done.

The scenarios in which the capture points had to be considered. Is the player currently capturing the point, have they left the point and is now being taken over by an enemy unit, are both teams contesting the capture point, etc. These had to be considered in order for it to work more naturally, akin to other games which have control points.

# User Interface

For the user interface, the information that a player would need is as follows:

* Current team scores
* Resource count
* Capture point status
* Game time

The interface itself wasn’t all too impressive. Simple images on the top of the screen gave way the current status of the various capture points, text that indicated the resource/score points of either team among other things listed above.

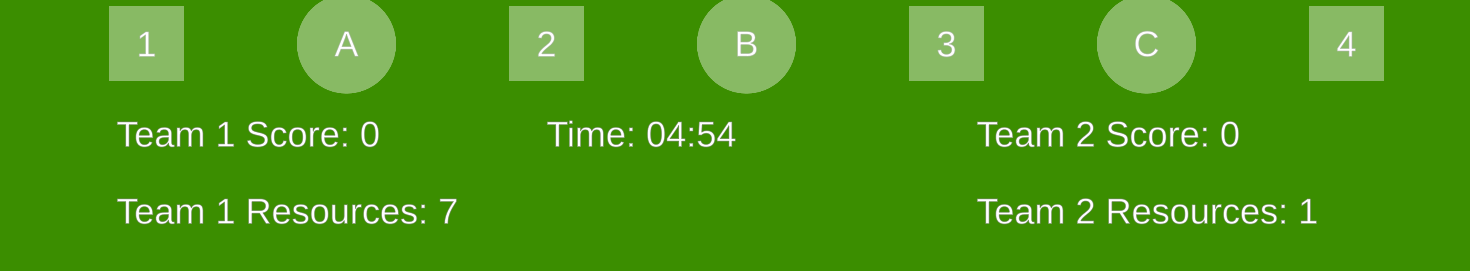


Fig. Current in game UI

# RTS Camera

The Real Time Strategy camera was what the player and by extension the machine learning agent could see. A simple script used for the movement scheme and controls were adopted. The more pressing part was unit selection and grouping. I would be using unity’s overlap function to get all object within the selection area’s transform. For the player’s selection, it would take all colliders into consideration and group them up.

The camera also allows the users to zoom in and out of the map, allowing a wider awareness of the game.

# Unit Spawning

There are two ways for the units to be spawned into the game. For the player they will be pressing a button that produces the unit after a timed delay, while the AI at current has their units spawned for them after a timed delay as well. They’re all spawned within a designated corner for each player.

# References

*Field of view visualisation (E01)*. (2015). Available from: https://www.youtube.com/watch?v=rQG9aUWarwE [accessed 26 April 2023].

Moor, W.D. (2023). Coordinated Group Movement (AI-Formations). , 10 April 2023. Available from: https://github.com/EezehDev/AI-Formations [accessed 26 April 2023].

Stacey. (2020). Get started with Behavior Trees. *GameDev Resources* [online], 16 November 2020. Available from: https://gamedev-resources.com/get-started-with-behavior-trees/ [accessed 26 April 2023].

*Unity ML-Agents Toolkit*. (2023). , 25 April 2023. Available from: https://github.com/Unity-Technologies/ml-agents [accessed 25 April 2023].

*What are Neural Networks? | IBM*. Available from: https://www.ibm.com/topics/neural-networks [accessed 25 April 2023].