



AI Tag Game

Project Engineering

Year 4

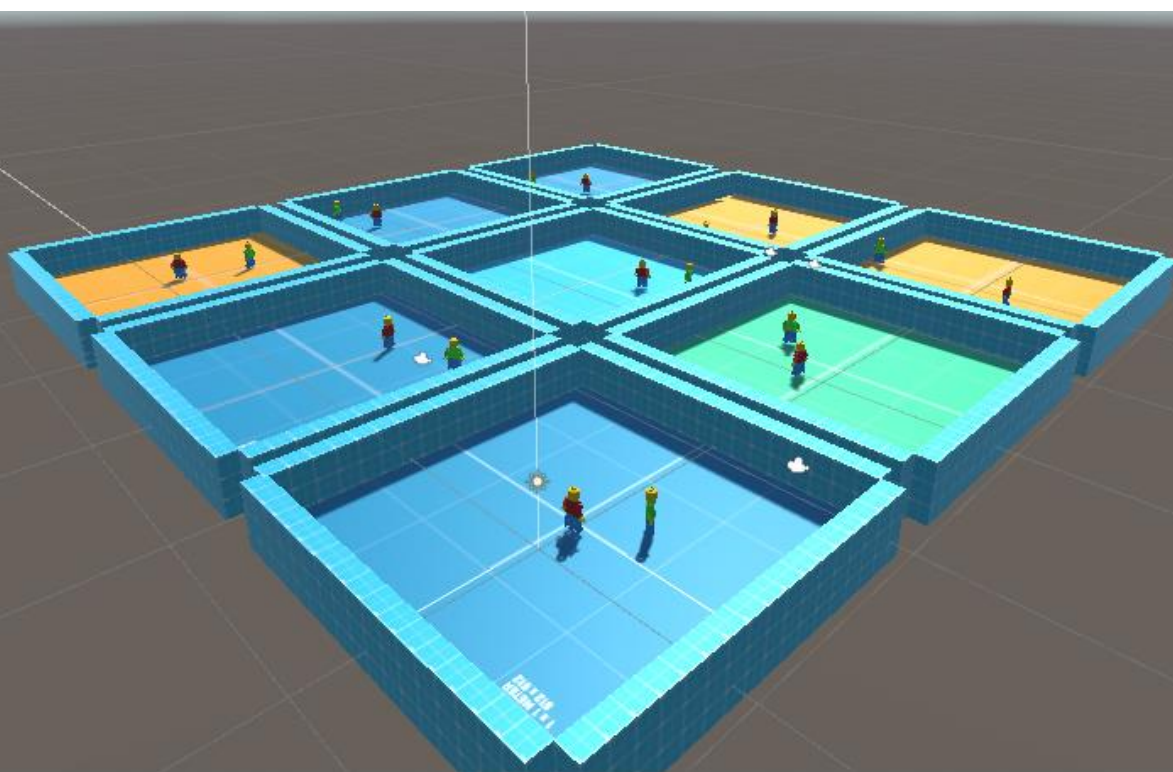
Matthew Waters

Bachelor of Engineering (Honours) in Software and
Electronic Engineering

Galway-Mayo Institute of Technology

2020/2021

Agents learning how to play tag



Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

Matthew Waters

G00357709

Acknowledgements

I would like to thank my supervisor Michelle Lynch for giving me valuable insight and support. I would also like to thank the other lecturers who held project classes, Niall O'Keefe, Paul Lennon and Brian O'Shea.

Table of Contents

1	Summary.....	6
2	Poster.....	7
3	Introduction.....	8
4	Beginning Information.....	9
4.1	Machine Learning.....	9
4.2	Unity.....	10
4.3	C#.....	11
4.4	Visual Studio.....	11
5	Project Architecture.....	12
6	Project Plan.....	13
7	Technologies.....	15
7.1	Unity.....	15
7.2	Reinforcement Learning.....	16
7.3	ML-Agents.....	17
7.4	Anaconda.....	20
8	Ethics.....	20
9	Conclusion.....	21
10	References.....	22

1 Summary

With my project I wanted to delve into something I had limited experience with. I was leaning over to artificial intelligence (AI). The idea of an artificial object using its own 'brain' to teach itself really interested me. Anytime I seen something related to AI I always read on. My goal for this project was to create two characters who would teach themselves how to play tag.

For this project I would need an engine to complete my project visually. This would allow a much easier experience with learning how things were going. I decided to go with Unity for this as I had a little bit of experience with it in the past.


I wanted to create a simple visual arena of two characters who would play tag with each other. With Unity I would be able to easily to achieve this. Unity is known to be easy to work with and a good start for beginners. Unity also has a plugin called ML-Agents which allows you to train AI using reinforced learning with Pycharm libraries.

The idea came when I saw a project done by OpenAi, who is a non-profit research organization for AI. They did a project with characters learning how to play AI when given different obstacles to use and hide with. The video I came across a few years ago showed how they learned to eventually exploit bugs in the game to gain an advantage. I wanted to replicate this and train AI characters to play tag as I found myself always going back and watching the same video and being fascinated.

I have managed to achieve this. I have two AI characters playing tag with each other. They constantly learn each other's weaknesses and try to use the environment to outsmart each other.

I enjoyed working on this project and will more than likely continue to try and develop this idea further and continue to work with trying to make the tag game more complicated for them.

2 Poster



GMIT
GALWAY MAYNO INSTITUTE OF TECHNOLOGY

AI Tag Game with Unity

Matthew Waters
BEng Software & Electronic Engineering

Summary

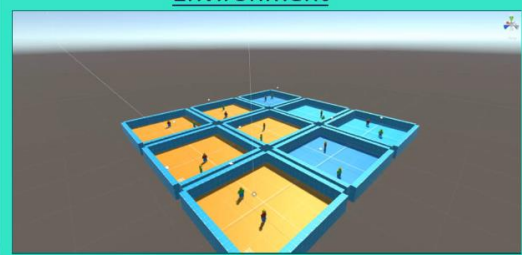
My plan for my project is to create a tag game with Unity which I will use as a base to train two AI agents to play tag with each other. I have a fascination with games, and I like the idea of AI characters playing against each other, so I want to bring the two ideas together.

I plan to use Unity's ML-Agents plugin which uses reinforcement learning to teach the AI agents. I think this is where the challenge will be. To successfully program the agents so they can learn easily and hopefully play tag with each other.

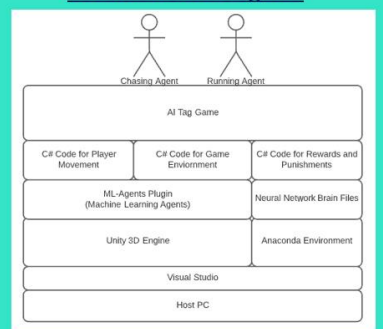
This will be challenging because I will be learning a new language C# and learning how reinforcement learning works along with teaching a neural network for the agents.

The basic game of tag starts off simple with two characters chasing each other.

Agents Learning Environment



Architectural Diagram



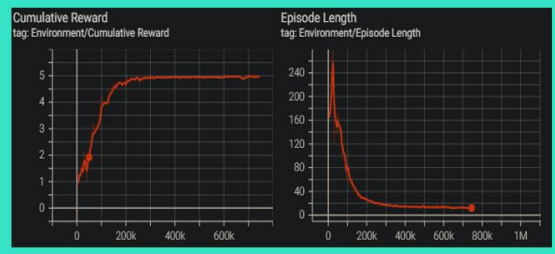
Unity

Unity is a cross platform game development engine which was developed with ease of use in mind which is why it was my chosen development platform.

Unity has a plugin called ML-Agents which allows you to train AI Agents within Unity

It is developed in a way where it can create extremely complex games. There is many corporations which use Unity as their game engine and use it to create industry standard games which do very well. Having this diversity and support in a game engine is great for modern game development.

Agent learning to how to get a reward faster



Reinforced Learning

Reinforced Learning works very similar to how humans and animals use their brain to learn new things.

Reinforced learning uses its own brain and given abilities to make observations about its surroundings and use these observations to make a decision.

The brain then gets a reward depending on its decision. If it is correct it gets a big reward and learns that they did something correct. If it is incorrect it gets a negative reward and then learns from this mistake for the next time it tries.

3 Introduction

This project was set to create an AI tag game with Unity. The characters teach themselves to play tag using reinforcement learning. I have a background and fascination with games. I have also always had an interest in AI and everything AI, but I never fully delved into the idea. I wanted to bring these two ideas together as it was a new area for me.

I decided to use Unity as I had some experience with the basics already. I also knew it had a plugin called ML-Agents which allowed you to train characters how you wish. The challenge was in this area for me as I had experience with creating arenas and characters already, but I have never used ML-Agents or touched AI before. The aim was to successfully program two agents to play tag together and learn how to outsmart each other. I learned how all this works as well as learning a new language, C# as this is the language Unity uses for scripts.

This project will receive no user input besides pressing go. The AI characters will learn to play tag together.

4 Beginning Information

4.1 Machine Learning

Machine learning is the learning of artificial intelligence through the given data where it learns to make decisions to reach an end goal. It is becoming very common in the modern world as a solution rather than hard coding software to make certain decisions. Some examples where machine learning is available in the everyday life is websites such as YouTube or Netflix which analyse your interaction with the app and suggest new relevant content to you. Self-driving cars is also a new and upcoming technology which uses machine learning algorithms to analyse the road ahead and use its observations to make certain reactions like stay in a lane or avoid a crash.

There are 3 main types of machine learning. **Supervised learning**, **unsupervised learning**, and **reinforced learning**.

Supervised learning is the easiest and most basic to understand. The algorithm is trained with data that is named and put in front of it to make it obvious what it has and what the exact end goal is. It takes the data it knows and creates the relations between the data and the solution to come up with the most efficient direction.

Unsupervised learning is given any data and an end goal. This data does not have to be specified how its used or what it is. The algorithm then learns how to use this data to get to the end goal efficiently. It creates its own path to get to the finish line rather than the supervised learning using a set path to get to the objective.

Reinforcement learning, which is my personal favourite and most interesting type. It works very similar to how humans and animals use their brain to learn new things. It uses its own brain to analyse the surroundings in the environment it has been placed in to gather observations about the surroundings. It then uses these observations to learn about how it can complete the task. These ideas created from the observations are reinforced by a reward system. The algorithm is rewarded for completing the task correctly. Usually there is multiple rewards, a big reward for completing the task extremely efficiently, a reward for going in the direction and a negative reward for a mistake or failure. This use of rewards is very similar to

how we learn. We learn that if we go near fire and touch the fire, that the fire is hot and to not touch it. In this case our reward for completing the task, which is not to touch the fire, is not getting burnt. Whereas if we touch the fire and fail the task, we get a negative reward as we get burnt. We will use this new information to make a better decision the next time we get around fire as we know it is dangerous. Reinforced learning works in this exact way. It uses its previous experiences to make decisions the next time it is faced with the problem.

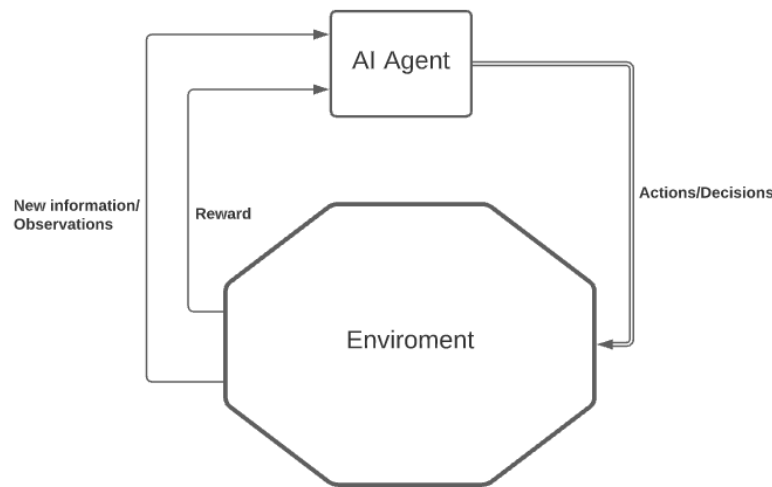


Figure 4-1 Architecture Diagram
Created with Lucid Charts

4.2 Unity

Unity is a cross platform game development engine which was developed with ease of use in mind. It is designed to be an easy access development engine which also has huge support and information about online. It is also developed in a way where it can create extremely complex games. There are many corporations which use Unity as their game engine and use it to create industry standard games which do very well. Having this diversity and support in a game engine is great for modern game development.

4.3 C#

C# is the language which Unity likes to work with when writing the scripts for your players/agents. It is intended to be multiuse and simple, which is why it is used in majority with Unity. Unity supports cross platform development so having a simple language which can be developed in many different areas and complexities is ideal for a platform like this.

I found this language relatively easy to pick up as it is easily describable as a mix between Java and C or C++. The syntax and design show traits from both sides. This allowed me to learn a new language in a developing environment where I could expand my knowledge of the language easily.

4.4 Visual Studio

The IDE I chose is Visual Studio as I had a lot of experience with this and it directly supports C# and Unity. I can use this to link with Unity to step through and debug my code if need be. I found Visual Studio to be easy to use with C# as it supported auto complete which helped me to find child objects and functions without knowing the exact name.

5 Project Architecture

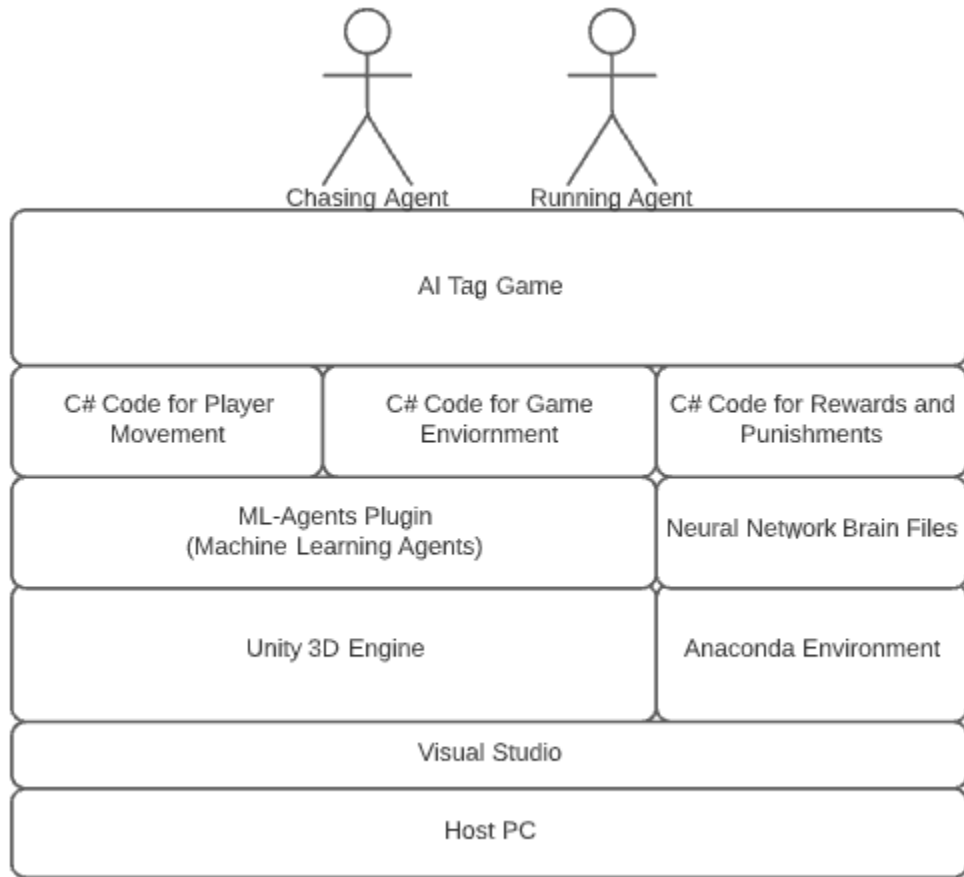
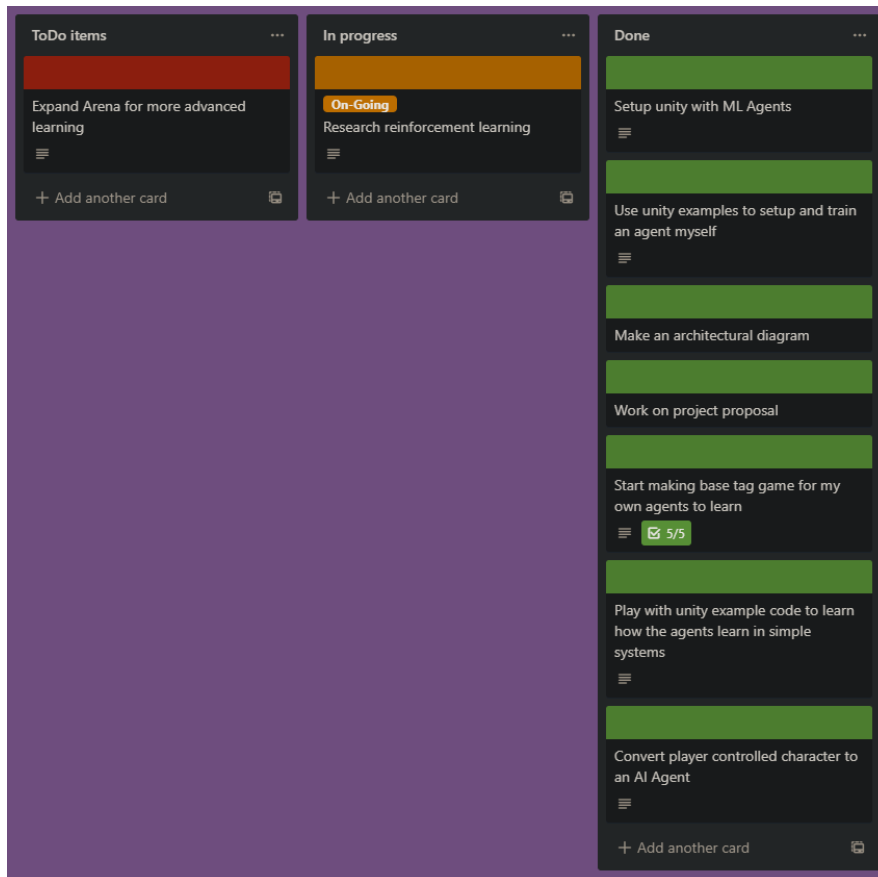


Figure 5-1 Architecture Diagram

6 Project Plan

With this project I wanted to develop something that would catch my interest. I wanted to create something to do with games which sparked the idea of combining AI. Below is what I used to plan and create ideas to progress the project. I went into this project with plans to continue develop the project. I did not have a set 'done' in terms of complexity of the game and what the agents are doing. I wanted to continue to develop this project over a long period of time and keep the complexity increasing. Thus because of this I did not want to set a fixed timeline, but to have set objectives that I wanted to complete. This allowed me to work on the objectives I have with head room to develop them further and get to know what I was doing to put my own twist on them.

Trello:



I found that Trello was very nice to visually layout and inspect what was needed to be done. I started by creating a list of what I thought was needed to be done. I then added on to these cards as I got more and more into the development of this project.

Journal and Github:

I was using GitHub to store my journal which I would update when I saw fit. I included things that I had completed within a timeframe which I outline and would also include what I wanted to get done the next time frame. I found this was useful when I was reflecting on what I had done as I would be able to compare Trello and the Journal and see how my progress was going.

7 Technologies

In this section I will go through some of the technologies associated with my project. This project is done solely through software. I will outline and explain the aspects of the technologies that made this project possible

7.1 Unity

For this project I decided to go with Unity. Unity is a popular video game engine. It is well known in video game development as many popular games today are made in Unity. I found that it had an intuitive user interface and had endless resources and support for development.

I chose Unity as it would allow me to give an easy visual representation of an AI character as I would be able to create my own virtual arena with a character who will be controlled by AI. Unity also comes with a plugin called ML-Agents which allows you to train AI to your customizable need.

Unity uses C# language to script its characters. You use scripts to control the behaviour of all objects within your environment. It works by constantly looping and reading data that you have set within your code.

```

12  void Start () {
13
14  }
15
16
17  //Update is called once per frame
18
19  void Update () {
20
21  }

```

Here is the basic starting script for Unity. The Start() function runs once and you usually use this to setup objects position or other variables that do not need to be changed.

The Update() constatly loops and this is where the majority of your code will sit such as checking for key presses etc.

7.2 Reinforcement Learning

The ML-Agents plugin uses a type of AI learning called Reinforcement Learning. Reinforcement Learning works by giving the AI Agent actions and allow them to use these given actions to maximize the efficiency of receiving a cumulative reward. For example, in this project, the character gets a bigger reward depending how quick it tags the other player, which I will outline below.

```
public override void OnActionReceived(ActionBuffers actionBuffers)
{
    //Debug.Log("OnActionReceived()");
    //what ai does with various actions
    Move(actionBuffers.DiscreteActions);

    AddReward(-1f / MaxStep);
    if(GetCumulativeReward() == -1f)
        StartCoroutine(ChangeMaterial(failureMaterial, 2));

    Debug.Log("Reward: " + GetCumulativeReward() + " Max Step: " + MaxStep);
}
```

Here you can see the AddReward() function is where the agent gets his reward. This OnActionReceived() function gets called every step and decides what to do with the observations the agent is constantly receiving from its environment. For every step (which is every Unity loop) the agent is getting a negative reward to encourage the agent to move quicker.

```
public void TaggedAPlayer()
{
    Debug.Log("TaggedAPlayer()");
    AddReward(5f);
    Debug.Log("Reward: " + GetCumulativeReward());
    StartCoroutine(ChangeMaterial(successMaterial, 2));
    EndEpisode();
}
```

This is where the player gets his big reward. This function gets called when the tags for each player get detected that they have collided. It is done with a function called OnCollisionEnter(), which is a built in Unity function which is constantly polled to check if the player tags have touched. Once it detects a collision, Unity knows that the player is tagged and it then calls the TaggedAPlayer() function and gives the agent a big reward to encourage the agent to go towards this reward.

Reinforcement learning is another reason why I chose ML-Agents. Reinforcement learning works close to how an animal or human learns in the world. We get placed into this world which is our environment. We know certain actions and we use these actions to learn. For example, we know that fire is hot from learning through our senses that it is hot and will hurt if we go near it. AI Agents use reinforcement learning to learn this way also. We give them their senses and they use these to learn about the environment they are placed in.

There are two main models within reinforcement learning, Markov Decision Process Learning and Q-Learning. This project uses Q-Learning, which technically is a type of Markov learning, but it does not know specific penalties and rewards. Q learning allows the agent to be fully dropped into an environment and must learn itself using only its senses and knowledge that the agent is getting a penalty the longer it spends not getting a reward.

I like this model-free learning as it allows the agent to learn itself which is how I wanted to go about this project. I wanted the agent to learn itself when given minimal information as that is what was most interesting for me.

7.3 ML-Agents

ML-Agents combines both Unity and reinforced learning. It is an open source Unity plugin which allows you to train neural network models for agents you have created within Unity. It allows an environment for your agents to learn continuously within Unity. You can visually see your agents learning which was hugely beneficial for me as I could see how well the agents were learning with the senses, I had given them.

ML-Agents uses Unity as its learning environment where it communicates with Python API's which contains the learning algorithms. The Python API's control training. It is used to start and control all aspects needed to run learning inside of the Unity environment. It takes the observations taken from the Unity environment as well as what it knows about rewards. The observations in this project mainly contain, the position of the agent as well as the boundaries and position of the character the agent is supposed to be tagging. It then uses these observations to make an action depending on the actions you have given it, i.e move forward, back, left or right.

ML-Agents outputs a Neural Network 'Brain' which you can then use to control your agent once it has learned. You can also use the same Brain to continue learning and allow it to increase its

Unity has functions inside it which automatically get called on each step(loop), like Start() and Update() mentioned before. Within the ML-Agent library there are functions which get called at certain stages also.

The first one is OnEpisodeBegin().

```
public override void OnEpisodeBegin()
{
    //reset
    Debug.Log("OnEpisodeBegin()");
    ResetPlayerPos();
    taggedPlayer.transform.localPosition = new Vector3(Random.Range(-7f, 7f), 1f, Random.Range(-9.3f, 5.3f));
}
```

This function acts like a reset for the environment. It gets called at the very beginning of the project, and when a new episode begins which is when the max step count is set. The max step count is set by the user and it counts every loop which Unity does. Once this max step count is hit it counts as an episode which then called this function and resets the players to their starting position.

The next important function is CollectObservations().

```
public override void CollectObservations(VectorSensor sensor)
{
    sensor.AddObservation(transform.localPosition);
    sensor.AddObservation(tagPosition.localPosition);
}
```

As the title suggests, this function passes necessary observations to the agent. In this case I am giving the agent its own position as well as the position of the other player as this is similar to using its eyes to see the other player.

OnActionReceived() is called every step and it analyses its observations and information to make a decision on what to do. The actions available are passed to the Move() function which I have written and controls the directions the player can move in. I am also doing a negative reward here to penalize the agent not moving, as discussed in the previous section. For ease of learning I am also changing the material/colour to red if the player fails to tag within the episode. This allow me to visually monitor how well the learning is taking place.

```
public override void OnActionReceived(ActionBuffers actionBuffers)
{
    //Debug.Log("OnActionReceived()");
    //what ai does with various actions
    Move(actionBuffers.DiscreteActions);

    AddReward(-1f / MaxStep);
    if(GetCumulativeReward() == -1f)
        StartCoroutine(ChangeMaterial(failureMaterial, 2));

    Debug.Log("Reward: " + GetCumulativeReward() + " Max Step: " + MaxStep);
}
```

This Heuristic() function is used for testing. It allows me to manually control the Agent to test that the abilities and directions the agent can move in is working correctly. For each if statement I am checking for a keypress which then sets the first index of the actions array to a number. This then gets used in the OnActionRecieved() function and passes this actions array to the move function which has a switch statement, checks the number that is set inside this array and moves in the relevant direction. For example, when I click the W key, the first index of the array is set to one, it gets passed to the Move() function in OnActionRecieved() which the switch statement uses to transform the agent forward. This same logic applies to each direction.

```
1 reference
public override void Heuristic(in ActionBuffers actionsOut)
{
    //player override for test
    //Debug.Log("Heuristic()");
    var discreteActionsOut = actionsOut.DiscreteActions;
    discreteActionsOut[0] = 0;
    discreteActionsOut[1] = 0;

    if(Input.GetKey(KeyCode.W))
    {
        discreteActionsOut[0] = 1;
    }
    else if (Input.GetKey(KeyCode.A))
    {
        discreteActionsOut[0] = 4;
    }
    else if (Input.GetKey(KeyCode.S))
    {
        discreteActionsOut[0] = 2;
    }
    else if (Input.GetKey(KeyCode.D))
    {
        discreteActionsOut[0] = 3;
    }

    if (Input.GetKey(KeyCode.LeftArrow))
    {
        discreteActionsOut[1] = 2;
    }
    else if (Input.GetKey(KeyCode.RightArrow))
    {
        discreteActionsOut[1] = 1;
    }
}
```

7.4 Anaconda

Anaconda Environment is a virtual Python environment. I used this to setup a virtual environment where I installed the ML-Agents Python libraries which would handle the learning. I used a virtual environment because I did not want these libraries to affect or be affected by the Python installation on my PC. This is new technology, so I did not want the ML-Agents libraries to possibly tamper with my Python installation.

8 Ethics

With this project specifically there is not any area where ethics would come into play as it is solely a character going from one point to another.

Artificial Intelligence does cause some ethical concerns, especially with the fear created around AI from movies etc. The concern is that AI does not have moral thinking that humans have. This can cause bias towards the 'easiest' way to complete the goal. Because of this the 'easiest' route to the goal can often result in a route which is morally incorrect according to human thinking.

We can battle this through giving the algorithm base specifications, similar to how we give it the goal and abilities for the choices it can make.

I can relate this same thinking into my project as in the future when more complexity is developed, there will be a need to state what the character can do when using the more complex arenas.

9 Conclusion

This project was extremely enjoyable for me as I finally got to delve into the world of Artificial Intelligence. It allowed me to combine my fascination for games with something I have been fascinated about but never thought I was able to research and develop. I do plan to further develop this project and try to increase the complexity with the game of tag that the agents are playing.

I learned a lot from this project as I was looking into a new area as well as learning a new language which was C#. Coming away from this project and looking back I feel as though I could improve on my organisation of work and working on different parts of the project individually instead of developing multiple parts of the project at once.

10 References

- [1] Unity Technologies, "ML-Agents Github", *GitHub*. [Online]. Available: <https://github.com/Unity-Technologies/ml-agents>. [Accessed: October 2020].
- [2] S. Schuchmann, "Unity ML-Agents 1.0+ - Self Play explained.", *YouTube* 2020. [Online]. Available: <https://www.youtube.com/watch?v=zAtcRbYdvuw>. [Accessed: October 2020]
- [3] T. Simonini, "An Introduction to Unity ML-Agents", *Towards Data Science*, 2020. [Online]. Available: <https://towardsdatascience.com/an-introduction-to-unity-ml-agents-6238452fcf4c>. [Accessed: November 2020].
- [4] Guru99, "Reinforcement Learning: What is, Algorithms, Applications, Example", *Guru99.com*. [Online]. Available: <https://www.guru99.com/reinforcement-learning-tutorial.html>. [Accessed: November 2020].
- [5] S. Schuchmann, "Ultimate Walkthrough for ML-Agents in Unity3D", *Towards Data Science*, 2020. [Online]. Available: <https://towardsdatascience.com/ultimate-walkthrough-for-ml-agents-in-unity3d-5603f76f68b>. [Accessed: December 2020].
- [6] K. Doshi, "Reinforcement Learning Explained Visually (Part 4): Q Learning", *Towards Data Science*, 2020. [Online]. Available: <https://towardsdatascience.com/reinforcement-learning-explained-visually-part-4-q-learning-step-by-step-b65efb731d3e>. [Accessed: February 2021].
- [7] "Machine Learning: What it is and why it matters", *Sas.com*, 2021. [Online]. Available: https://www.sas.com/en_ie/insights/analytics/machine-learning.html#:~:text=Machine%20learning%20is%20a%20method,decisions%20with%20minimal%20human%20intervention. [Accessed: February 2021].
- [8] "C Sharp (programming language)", *wikipedia.org*. [Online]. Available: [https://en.wikipedia.org/wiki/C_Sharp_\(programming_language\)](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)). [Accessed: Apr- 2021].
- [9] "Unity (game engine)", *wikipedia.org*. [Online]. Available: [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine)). [Accessed: April 2021].