



Project Title

Project Engineering

Year 4

Tosin Biala

Bachelor of Engineering (Honours) in Software and
Electronic Engineering

Galway-Mayo Institute of Technology

2020/2021

Project Graphic (Optional)

Declaration

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

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

Acknowledgements

Use this section to acknowledge anyone who might have helped during your project if you wish to.

Table of Contents

1	Summary.....	6
2	Poster.....	8
3	Introduction	9
4	Background	10
5	Project Architecture	11
6	Project Plan.....	Error! Bookmark not defined.
7	Gamification	12
7.1	Unity	13
7.2	Vector.....	15
8	Git	16
9	GitHub.....	17
10	Machine Learning	18
11	Neural Networks	20
12	HyperParameter	22
13	ML Agents	23
14	PPO	26
15	Virtual Enviroment	28
16	Pytorch	29
17	ConClusion	32
18	Reference	32

1 Summary

The project I decided to work on throughout the final semester was a game called Sanio. My project, to its core, was a 2D unity-based game that implemented intelligence to the player using reinforcement learning. The objective of the game was to allow the player to finish the game without any external help.

I decided to do a project based on game development was because I was always intrigued by game development. I thought this project would be most enjoyable to complete. I also thought this project was unique compared to my other classmate's projects.

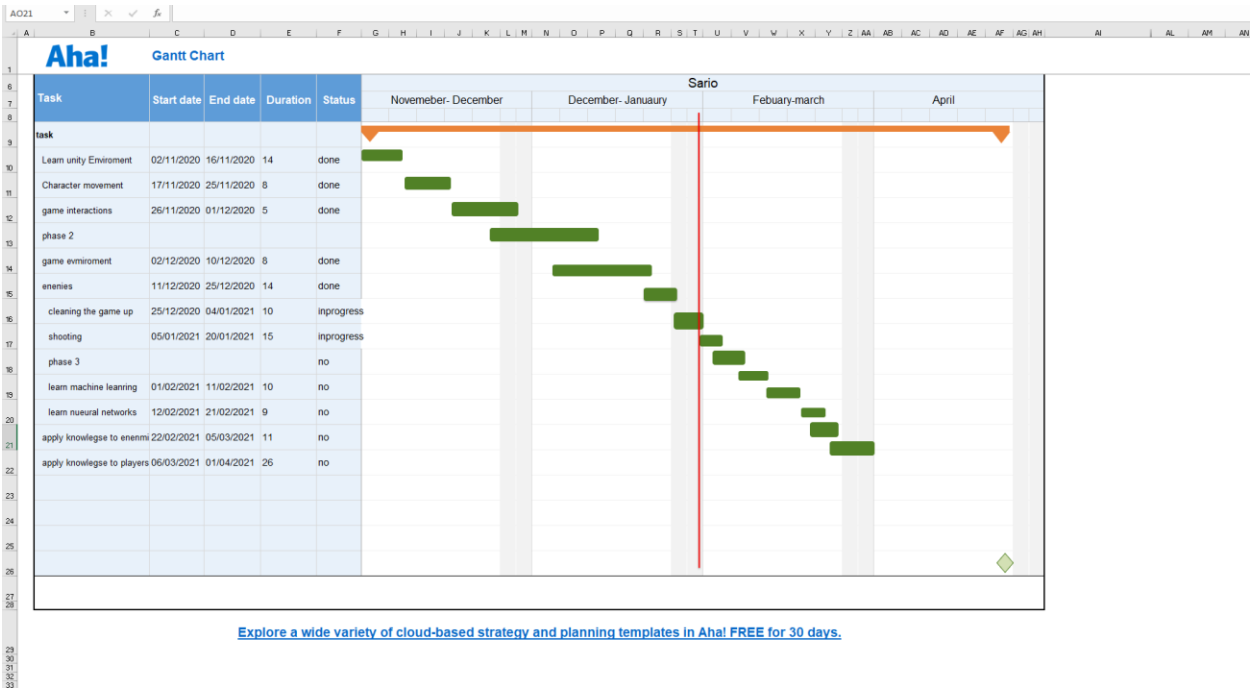
The essential skills I would like to gain when completing my project are to be a competent c# programmer are to gain skills in Unity and get familiar with machine learning to a stage that can create more complex machine learning-based projects.

The approach of my project was to train my player to play the game and go up every level. He would also try to aim to hit the highest score at each level. I then set a schedule for myself. If I finished my project a month earlier, I would implement reinforcement-learning to my players to learn from them and perform different activities to prevent themselves from dying.

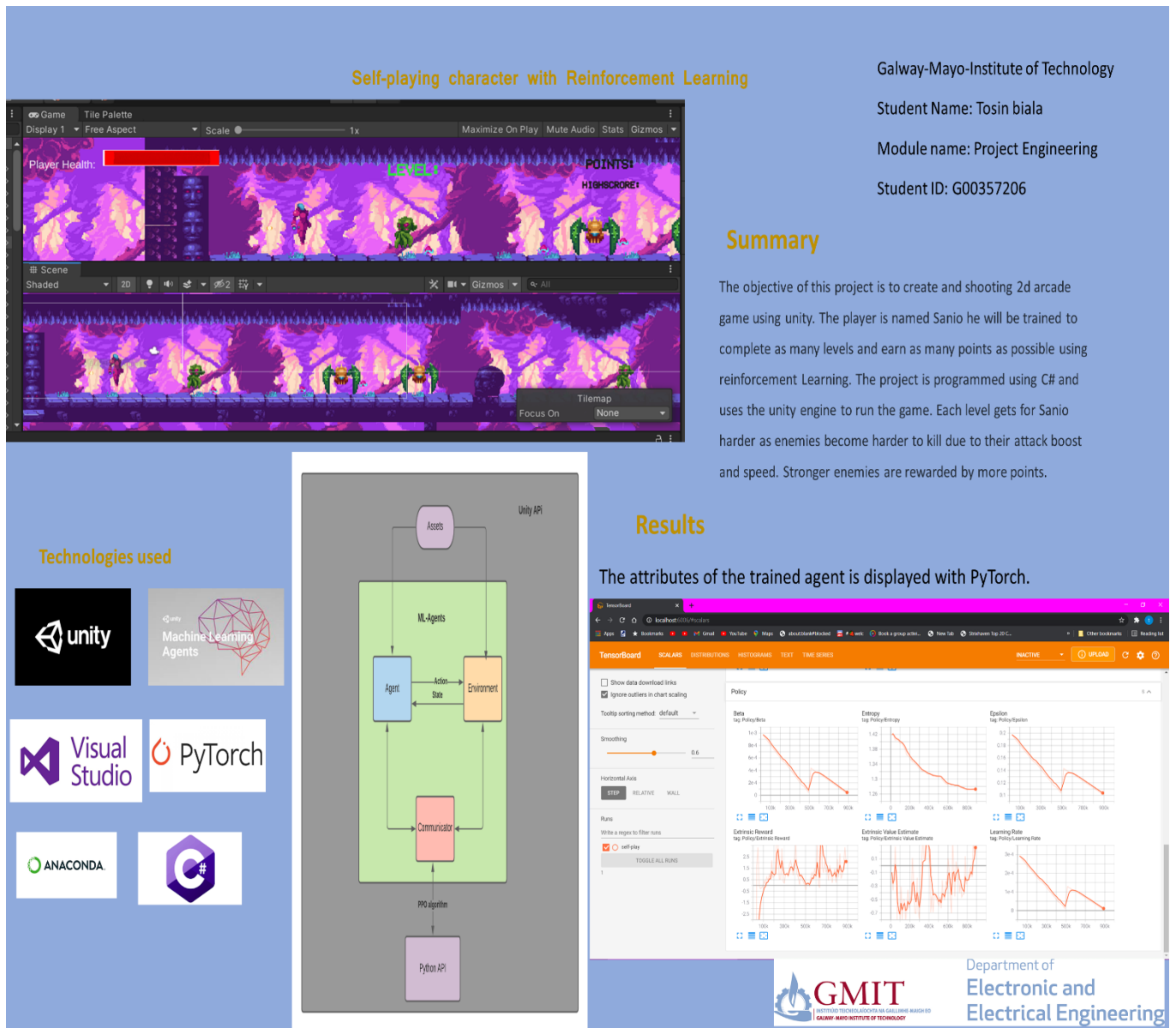
Unfortunately, due to time restraints, I was unable to complete the model.

This project is mainly focused on machine learning since I have no prior experience with it. To train my player, I used an open-source tool kit named ML-agents.

Project Plan



2 Poster



3 Introduction

This report will reveal the research I have studied throughout the final college year. It will also explain the technologies I've to use to complete my project. My goals for this would be to train my player to finish a game by himself and simultaneously to get a high score each time the player completes a game session.

My next objective is to add levels to the game. During each game level, the player's enemies will become faster and deal more damage. That is an extra feature I decided to implement in my project. Although the model might not be fully complete, I will demonstrate my understanding of the model. This part will be not so easy to train because the learning environment is constantly changing. The Agent will try to find the best policy, thus taking it longer to program and train.

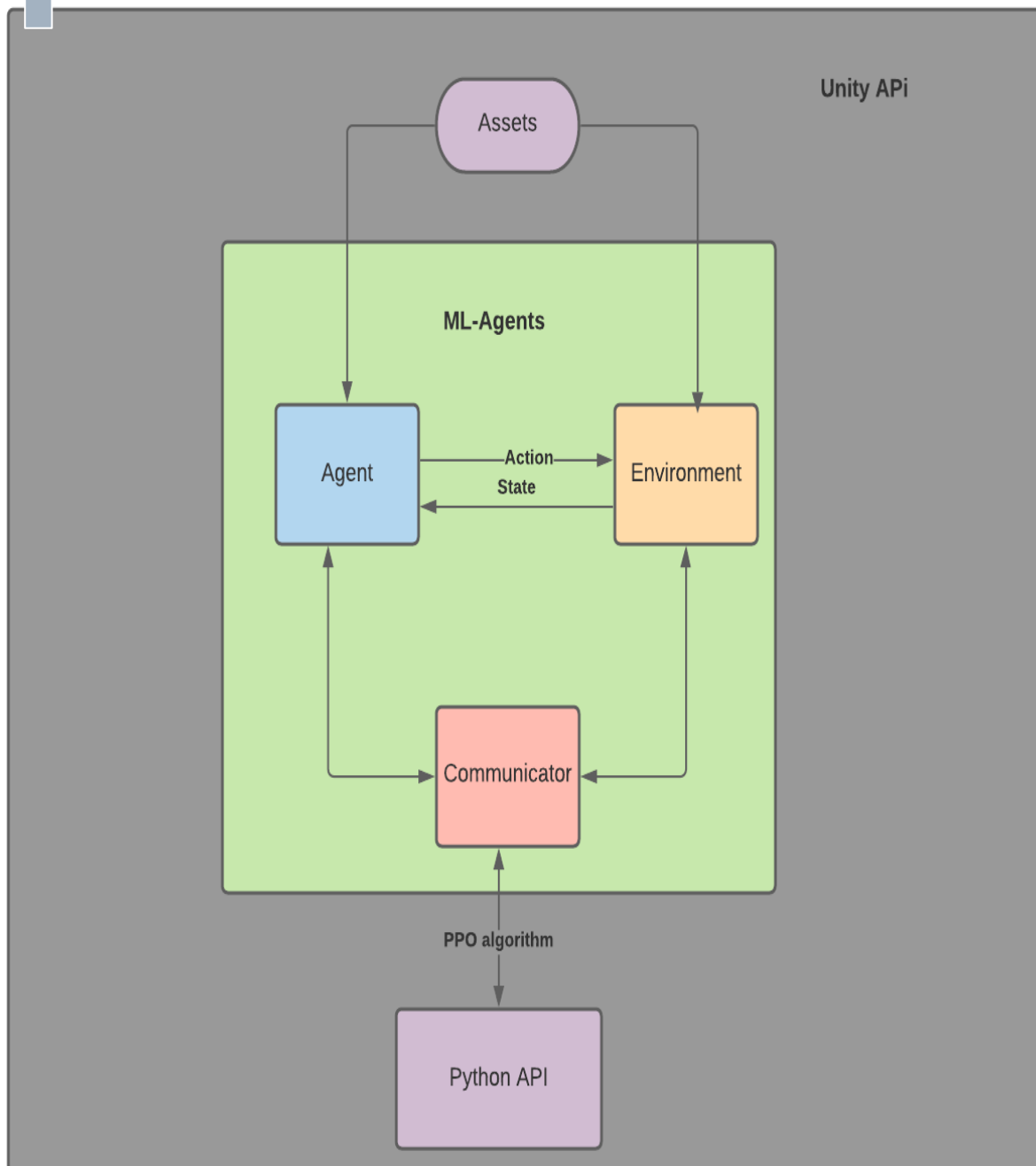
The next objective I would like to implement in my project would be the Ai enemy that trains from the players perspective and prevent them from being defeated. Although this isn't the most crucial feature, I will only try to implement it if I finish my project early.

4 Background

The Background technologies I would need to know for my project are

- C#
- Unity
- Machine Learning
- Reinforcement Learning
- Mlagents
- Anaconda
- PyTorch
- Git
- GitHub
- Neural Networks
- Hyper parameters

5 Project Architecture



6 Gamification

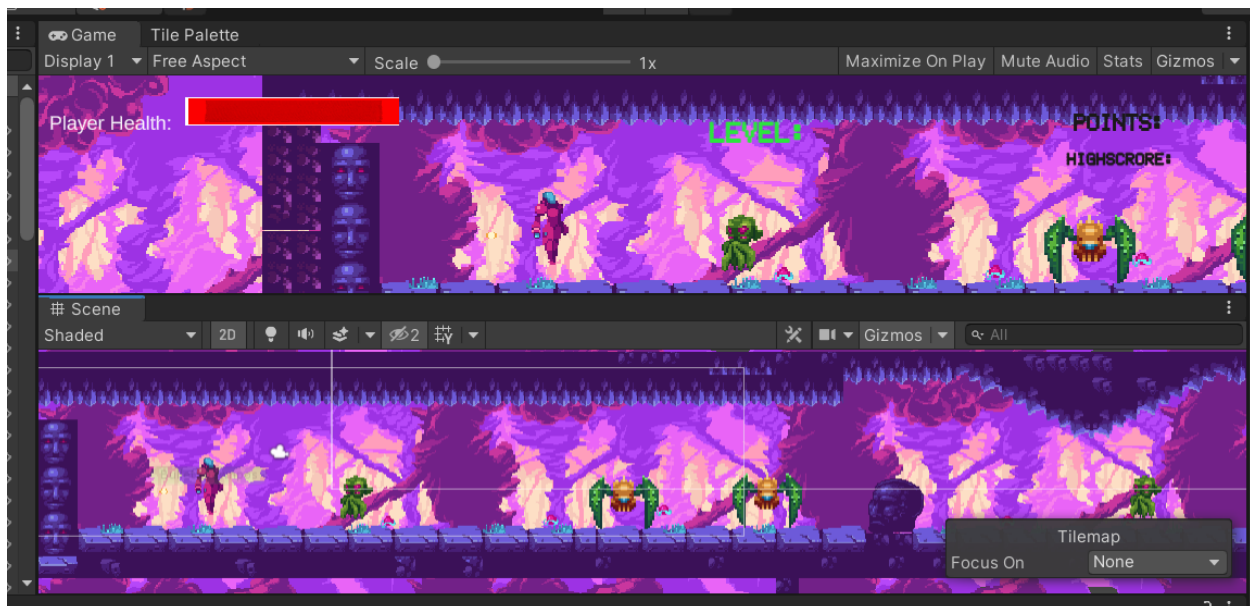
Gamification is the process of applying mechanic or game dynamic in a non-game context. The most prominent non-game context out there is that uses gamification are enterprises.

Gamification techniques are used in enterprises because they engage people in their products, whether by incentives, reward programs, badges, leader board and sometimes free stuff. These rewards motivate people to complete actions.

Another Successful implementation of gamification is Education. Math teachers have been applying some particular way to their classes to make their class environment more enthusiastic. The most popular way they implement gamification would be to add to give a reward at the end of the day based on who got the highest amount of question or create point anarchy based on how the most well-done points.

Trying to apply gamification to my project was difficult because my original idea was to create a 2D shooting game. But back then, early in development, I was not thinking about the gamification techniques; I was more concentrated on the complexity of my project. After some advice from classmates and supervisors, I came to a reionization on how vital gamification is. But as my project was changing throughout the college year, my previous gamification idea did not match my existing game idea at the time.

I decided on a game implantation design that would work for any configuration of my game. The Scoring system is implemented in the game. The player will earn points by destroying an enemy and goal of the game to achieve a high score at every level.



7 Unity

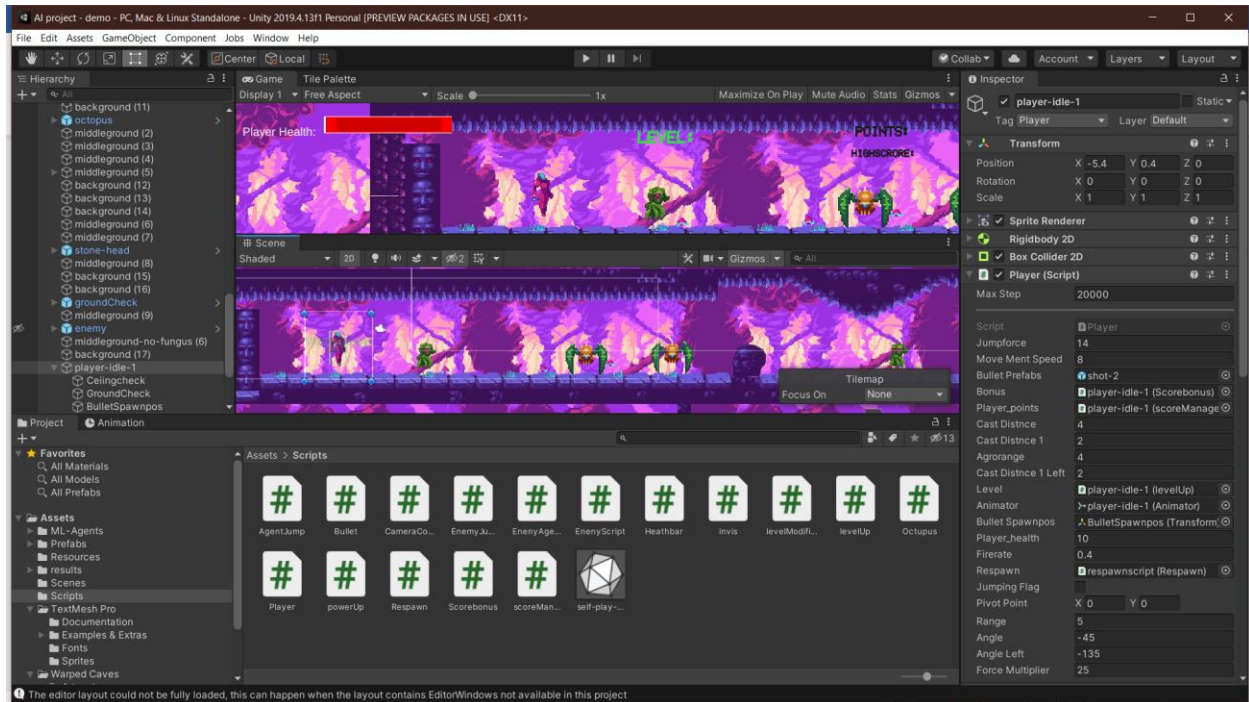
Unity is an American software development company based in San Francisco. Unity is an engine that enables users to design 2D/3D games or app scenes. Unity is not just for game design. It is a use or simulation design for businesses. Unity allows users to interact with assets with code and imported animation. The programming language that I needed to learn to use Unity was C#.

Another game engine I was planning of using was the Unreal engine. The unreal engine was a more powerful game engine than Unity and programmed in a language I had experience in, which was C++. I decided to go with Unity because I found out Unity is more user-friendly than the Unreal engine after some research. Also, because I was new to game development, I would want memory crashes due to insufficient memory management to disturb my progress during the project if I used the Unreal engine.

Instead of using language-based libraries that made it possible to create games, e.g. PyGame for Python, game engines are much faster efficient when creating games. One major factor that differs from those two would be Unity has a built-in physics engine, making it easier to program complicated game projects.

An example of a method that access physics in my project is Rigid body. It allows that game object assigned to it to give the game object access to gravity forces mass and Velocity. I also implanted some other physics such as colliders to check if a game object has collected with it and Raycast that shoots out a ray and from the action if I hit the specific thing.

In Unity, there are three fields Inspector, Hierarchy and Scene. The Hierarchy displays all the game object that is in your Scene. The Scene views show what the user can see when they start the game. The inspector shows the properties of a specific game object.



Vectors

In-game development, a vector is used to meshes calculate distance. A vector represents two points from origin to and 2D plane. The line that represents the space from the two points is called the magnitude. Unity can create vectors at any dimension. Since my project is a 2D game, I was mainly using 2D vectors. I use vectors in my project to get the position and my player. I also use it for distance calculation and to add force to my game object.

```

if(vectorAction[1] == 1 && Mathf.Abs(rigidbody.velocity.y) < 0.001f)
{
    vectorAction[1] = 0;
    animator.SetTrigger("isjumping");
    rigidbody.AddForce(new Vector2(0, Jumpforce), ForceMode2D.Impulse);
    rigidbody.AddForce(new Vector2(8f, 0), ForceMode2D.Impulse);
    //Debug.Log("Rigth jump is working fine");
}
else if (vectorAction[1] == 2 && Mathf.Abs(rigidbody.velocity.y) < 0.001f && jumpleft == true)
{
    animator.SetTrigger("isjumping");
    rigidbody.AddForce(new Vector2(0, Jumpforce), ForceMode2D.Impulse);
    rigidbody.AddForce(new Vector2(-8f, 0), ForceMode2D.Impulse);
    jumpleft = false;
    vectorAction[1] = 0;
    //Debug.Log("left jumping is workig fine toon");
}
}

```

Git

Git is a version control system. A version control system allows users to save the code that we are working on. Git uses features to revert subtle changes to their previous state or revert the entire project and compare changes over time. Git also has a part if you lose files or can quickly recover them. In git, there are three types of version control local version control, centralized version control and distributed version control.

Local version control is stored locally in a database and is the most common type of version control. An individual usually uses this type of version control. For my project, this is the method I used for my project

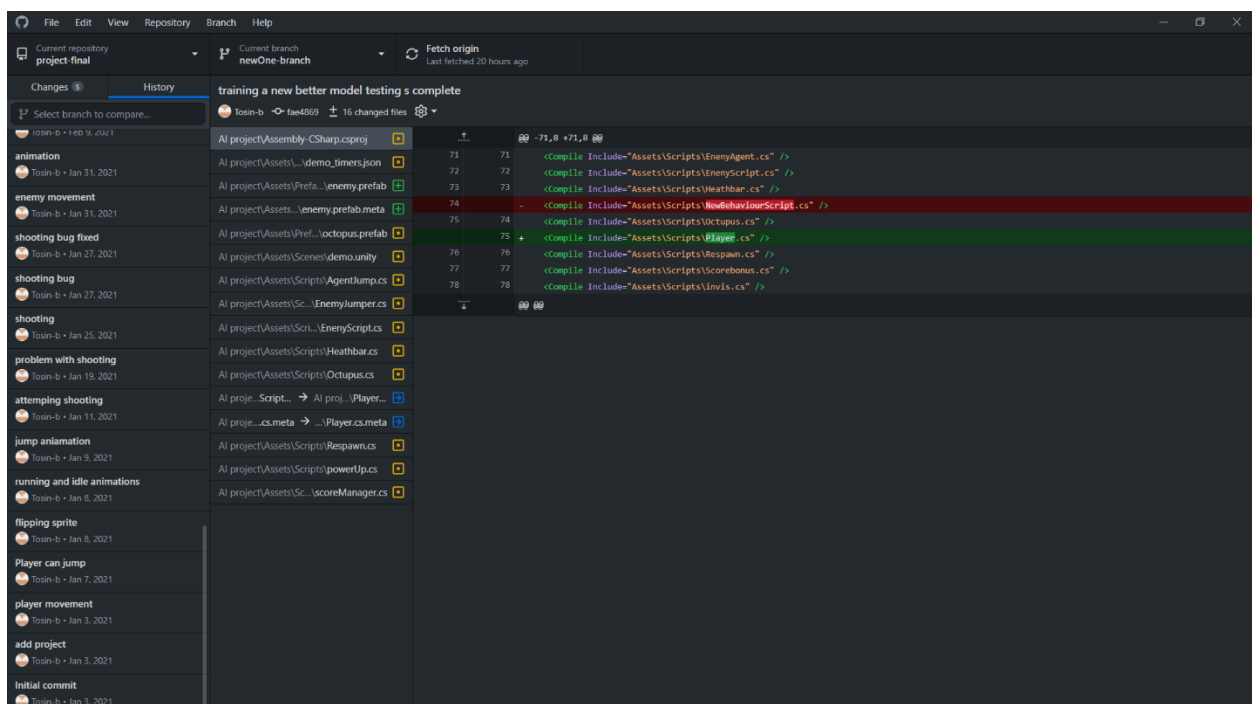
Centralized version control a version-controlled that you would usually use in a team. Code is then managed in the central VSC server. This type of control system gives the administrator more control of the system.

Commands I used during my project were git add, git init , git status, git commit. Git add, adds all untracked and new files and preparing for the next stage to commit. Git commit commits your project to a repository. Git init, to create a new repository. Git status gets the status of a file.

GitHub

GitHub is a platform that helps programmers work together. GitHub has a massive number of features, but because my project was an individual project, I was limited to the number of features I could use. GitHub allows programmers to share their projects that get stored in a repository to the world. One issue I've come across during my project was a merge conflict. Merge conflicts generate when the previous code has not pushed the local repository and attempted to commit more existing code. This issue was complex, too, because when starting my project, I was working on the master branch with I found out later this was not the most efficient way to use GitHub. I solved this problem was to create a new branch and clone my previous repository, save my clones repository into a new one.

Another problem was I had no gitignores in my repository, meaning GitHub read every single file in my unity project, making the size of the project bigger. So big that it surpassed the max size you can send, which was 100MB.



Machine learning

Machine learning is algorithms that give computers the ability to learn from existing data.

Machine learning is significant in the world we live in and constantly evolving. We use machine learning from self-driving cars to Amazon, Netflix recommendations, to fraud detection. The learning system is broken into three steps decision process, error function, a model optimizer process.

The decision process finds a pattern in the data then attempts to produce predictions.

The error function is responsible for comparison to validate if the generated model is accurate is fine.

The model Optimization process optimizes the model to fit data points in a training set by adjusting the weights. The algorithm repeats until the threshold has been met.

Machine learning Methods

There are different types of machine learning algorithm and methods in my projects, but the kind of method I decided to learn was reinforcement learning.

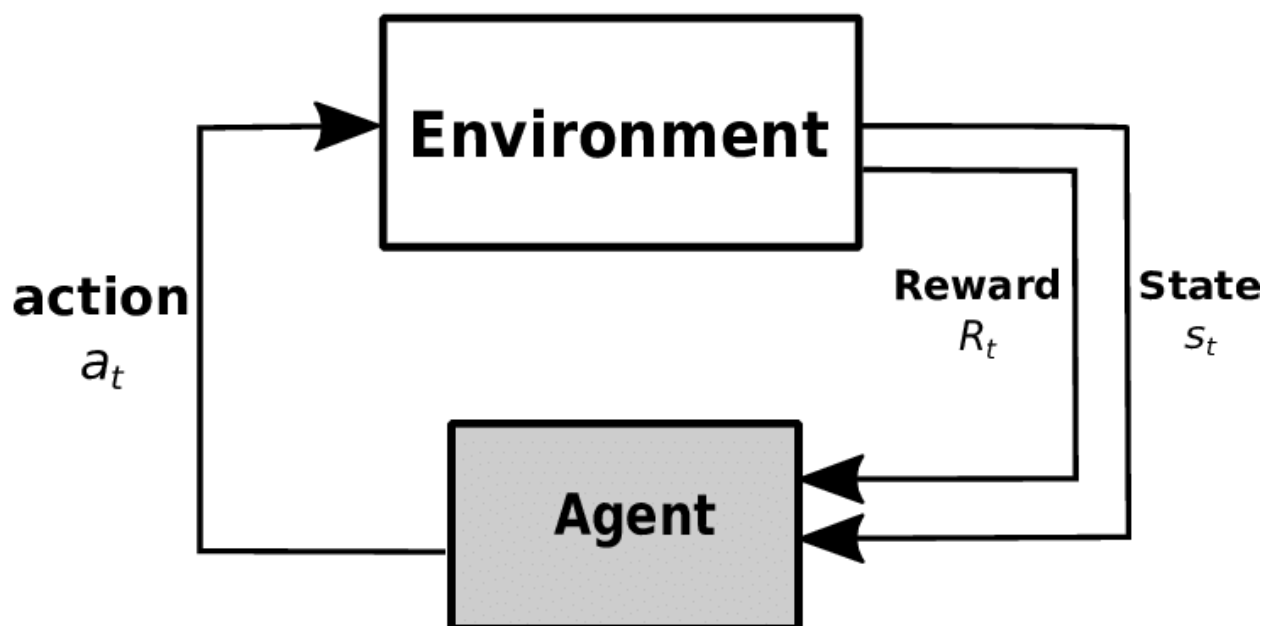
Reinforcement learning is the process of an agent receiving a reward based on the action it took. That reward is saved, and the Agent then tries to earn as many rewards as possible by taking action. The goal of reinforcement learning is to increase your cultivated reward steadily. Thus your Agent gets smarter. I decided to use this machine learning method because I wanted my player to learn from the environment and past experiences with no Prior knowledge.

In reinforcement learning, there is an environment. An environment is where the Agent is. The environment is then given a state to action.

A state is when an environment provides a situation in the environment to the Agent.

The Agent then determines an action to take based on the state that was given to the Agent.

During each state, a policy is generated. The policy calculates the best action to take that will earn you the most amount of rewards. The longer you train an agent, the better your policy is by calculating the best cumulative reward for future rewards



Q function can find the optimal action selection policy. It takes any state-action pair and gives you a real value and find the best action.

There are two other types of Reinforcement learning Supervised learning, unsupervised learning.

Supervised Learning is when a computer learns how to map an input to an output based on the input output pairs.

Unsupervised learning learns by observing patterns from untagged data.

Neural network

The human brain inspires neural networks. Neural networks can take in data train themselves by identifying a pattern and then predicting the outputs. A neural network is made up of neurons. The neurons job is to compute the weighted average of its input.

A neural network has three layers input layer, an output layer, and a hidden layer. When observation is fed into the neural network, they go through the input layer first. The neurons are then connected through channels. Each channel is given a weight. In the input layer, the inputs are multiplied to each neuron corresponding weight. The sum is then set to the hidden layer. These individual sums are called a bias. Then the bias value is then sent to an activation function. The role of an activation function is to determine a neuron is activated or not. An activated neuron transmits data to the next neuron layer through channels. This process is called forward propagation. At the output layer, the neurons with the highest probability will be the predicted ones.

Backpropagation is when the neural network compared the predicted result to the input and generates the error magnitude. Then transfer backed into the neural network.

For my project, I had to tune my hyperparameters to change the neural network. Num epoch hidden units num layers

Num epoch is how many times the neural network runs to generate a typical prediction range from three to ten. I implemented. The bigger the number, the longer it will take to train.

The hidden unit determines how many units are in each neuron layer typical range was from 32 to 512. I chose sixty-four. I tried it with thirty -two and my Agent didn't seem to learn very well. This was because the about of observation I was feeding into the neural network was not efficient at 32.

Num layers are how many hidden layers are represented in the neural network I decided to implement three. From using trial and error, I found three to be the most efficient one based on my Agent.

Hyperparameters

Hyperparameters interprets data. ML agents generate you a default one but the more observation you are feeding into your neural network, the poorer your neural network will perform. I will explain below the hyperparameter turning I've implemented into my project.

Learning rate speed up the neural network update weights depending on the number you insert typical range is $1e-5$ to $1e-3$. I chose my learning rate to be $1e-3$ because I wanted to update many times during training, thus slowly improves the Agent. I have a maximum rate of 0.3 will result in the loss of previous past experiences.

Learning_rate_scheduled determines your environment. If your environment does not change it should be linear but if your environment is constantly changing, it should be constant. I previously set this parameter to linear. Later on, I found out that I should have set it to constant because my environment was changing.

The **batch size** determines how many of the agents' experiences are fed into the neural network. To produce a neural network update. I set it to 64.

Buffer size determines the policy update

Beta determines how random the policy is at the beginning of training. if this value is high, the agent explores more of the environment.

Mlagents

Mlagents is a open source toolkit that I used to train my Agent. It provides simple to use python api to train your Agent. The python api contains machine learning algorithms.

ML agents provide three function OnActionRecieved, CollectObservations, OnEpisodeBegin, heuristic.

OnEpisodeBegin is when a episode end it calls this function and depending on what you want your program to do you would call it in this function. Usually you would just reset the scene but i needed to implement to reset the score and to bring you back to level 1.

```
10 references
public override void OnEpisodeBegin()
{
    Debug.Log(endgame);
    if (endgame)
    {
        Debug.Log("endgame has ben entered");
        endgame = false;
        respawn.Startagain();
        level.LevelingUp();

        Debug.Log(levelModifierScript.moveSpeedModifier + " moveSpeedModifier");
        Debug.Log(levelModifierScript.enemyDamageModifier + " enemyDamageModifier");
    }

    else if (Player_health <= 0)
    {
        respawn.again();
        levelModifierScript.DecreaseModifier();
        level.LevelDecrement();
        //bonus.resetPoints();
        Debug.Log(levelModifierScript.moveSpeedModifier + " moveSpeedModifier");
        Debug.Log(levelModifierScript.enemyDamageModifier + " enemyDamageModifier");
    }
}
```

OnActionReceived tells the Agent what the agent can control. In this function I would add rewards depending on the action that was taken. Since I want my player to walk back and forward, I added forces to the player's x and y axis so it can jump.

```

public override void OnActionReceived(float[] vectorAction)
{
    bool jumpLeft = false;
    Vector3 controlSignal = Vector3.zero;

    controlSignal.x = vectorAction[0];
    controlSignal.y = vectorAction[1];
    bulletDirection();
    GroundcheckLeft();
    Groundcheck();
    Falling();
    LeftRight();
    autoShoot();
    autoShootLeft();
    healthReset();
    jumpingLeft();
    jumpRight();

    if(vectorAction[1] == 1 && Mathf.Abs(rigidbody.velocity.y) < 0.001f)
    {
        vectorAction[1] = 0;
        animator.SetTrigger("isjumping");
        rigidbody.AddForce(new Vector2(0, Jumpforce), ForceMode2D.Impulse);
        rigidbody.AddForce(new Vector2(8f, 0), ForceMode2D.Impulse);
        //Debug.Log("Right jump is working fine");
    }

    else if (vectorAction[1] == -1 && Mathf.Abs(rigidbody.velocity.x) < 0.001f && jumpLeft == true)
    {
        vectorAction[1] = 0;
        animator.SetTrigger("isjumping");
        rigidbody.AddForce(new Vector2(-8f, 0), ForceMode2D.Impulse);
        rigidbody.AddForce(new Vector2(0, Jumpforce), ForceMode2D.Impulse);
        //Debug.Log("Left jump is working fine");
    }
}

```

Collects observations collected the observations that I was going to use to feed into my neural network. These were the player's x and y position so the player could learn to move and jump. Raycast to check if an enemy is near the player. The player's health and points to allow the Agent to keep aware of the health.

```

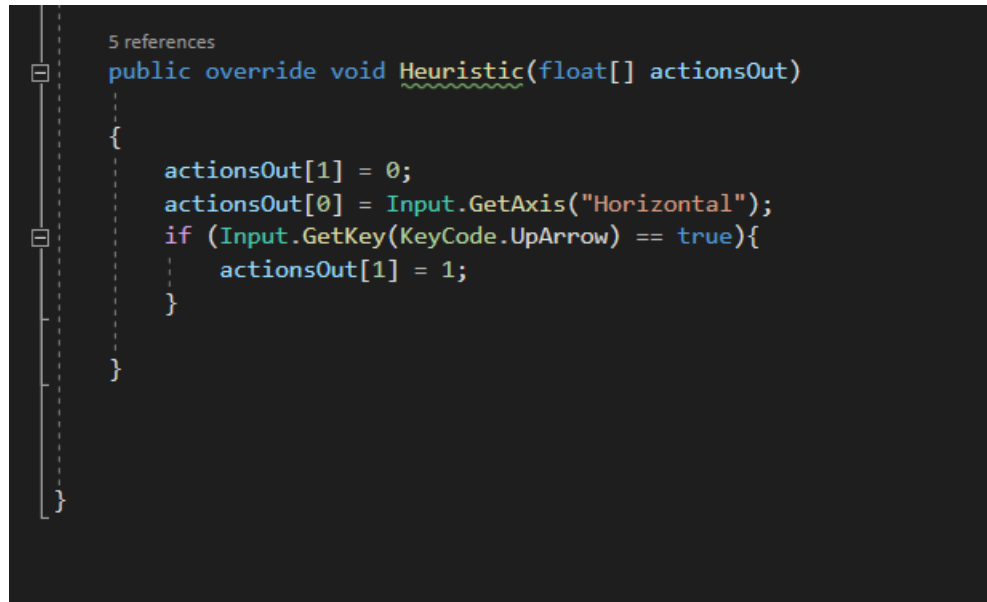
public override void CollectObservations(VectorSensor sensor)
{
    int observ_player_score = Player_points.score;
    Vector2 endpos = transform.position + Vector3.right * castDistance;
    RaycastHit2D hit = Physics2D.Linecast(transform.position, endpos, 1 << LayerMask.NameToLayer("Action"));
    Vector2 health = new Vector2(Player_health, 12);

    //Debug.Log("Scaled health: " + health.magnitude);

    sensor.AddObservation(transform.localPosition);
    sensor.AddObservation(hit);
    sensor.AddObservation(transform.position.x);
    sensor.AddObservation(transform.position.y);
    sensor.AddObservation(Player_health);
    sensor.AddObservation(observ_player_score);
}

```


With the heuristic feature I was able to test my environment for bugs before I went to train my Agent. I would do this by taking control of the jumping and the movement that the Agent has control.

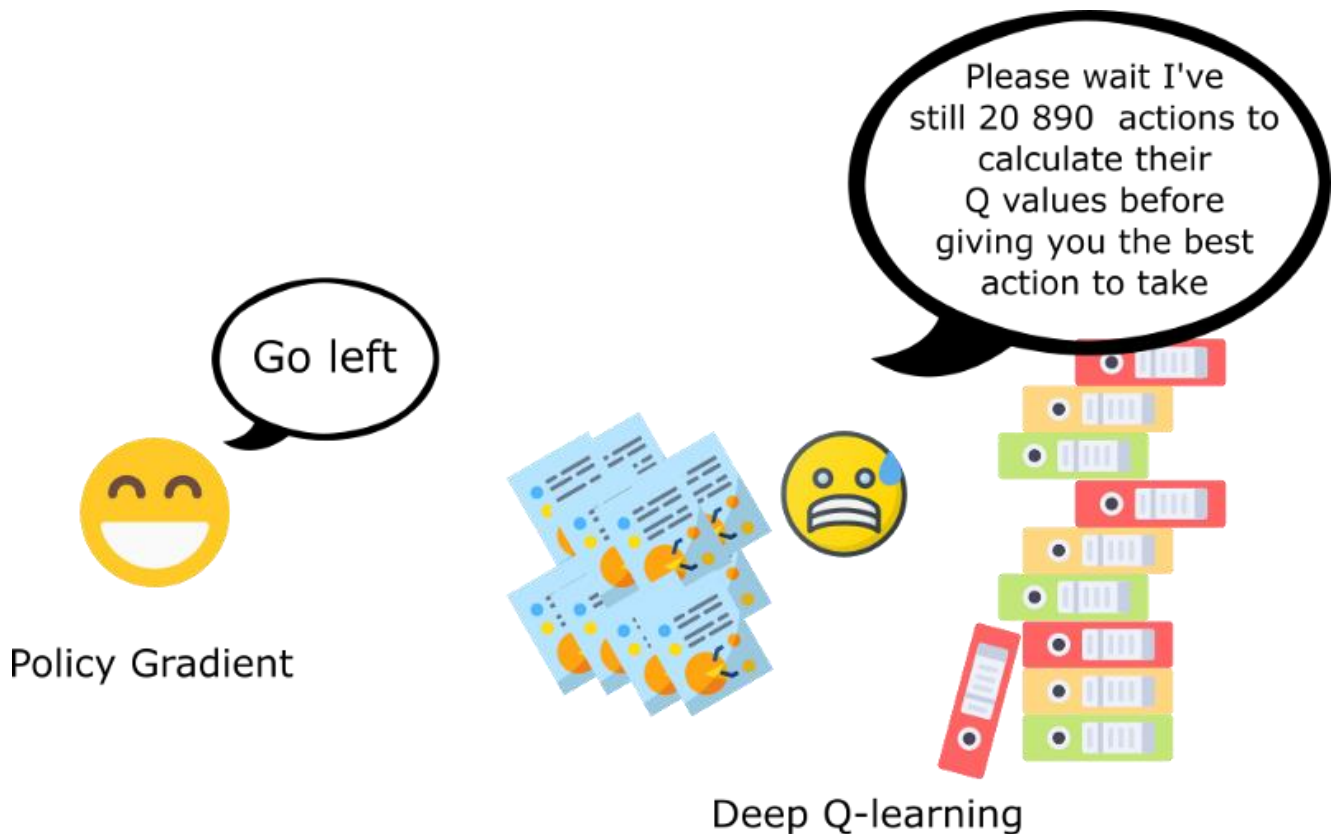
A screenshot of a code editor with a dark background. On the left, there is a vertical scrollbar and a line of code with a small square icon. The code is in C# and defines a method named `Heuristic`. The method signature is `public override void Heuristic(float[] actionsOut)`. The body of the method is enclosed in curly braces. Inside, the first line is `actionsOut[1] = 0;`. The second line is `actionsOut[0] = Input.GetAxis("Horizontal");`. The third line is an `if` statement: `if (Input.GetKey(KeyCode.UpArrow) == true){`. Inside the `if` block, the line is `actionsOut[1] = 1;`. The `if` block is closed with a closing brace. The method body is closed with a closing brace. The code is color-coded: `public` is blue, `override` is blue, `void` is blue, `Heuristic` is green and underlined, `float[]` is blue, `actionsOut` is blue, `actionsOut[1]` is blue, `0` is blue, `actionsOut[0]` is blue, `Input` is blue, `GetAxis` is blue, `"Horizontal"` is red, `if` is blue, `Input` is blue, `GetKey` is blue, `KeyCode` is blue, `UpArrow` is blue, `==` is blue, `true` is blue, `actionsOut[1]` is blue, `1` is blue. At the top left of the code area, it says "5 references".

```
5 references
public override void Heuristic(float[] actionsOut)
{
    actionsOut[1] = 0;
    actionsOut[0] = Input.GetAxis("Horizontal");
    if (Input.GetKey(KeyCode.UpArrow) == true){
        actionsOut[1] = 1;
    }
}
```

Proximal Policy Optimization

PPO is a policy that was created for reinforcement Learning. The goal of this concept is to have an algorithm that has great data efficiency and performance when training an agent.

In PPO, the concept advantage means how good an action is compared to another action at a specific state. The Advantage is = to the q value minus that state value. Because PPO is a policy gradient method this means it doesn't use a replay buffer to track past experiences but uses whatever the agent encounters in the environment.



Policy Gradient loss

$$L^{PG}(\theta) = \mathbb{E}_t \left[\log \pi_{\theta}(a_t | s_t) \hat{A}_t \right].$$

The π_{θ} of the equation is the probability outcome from the policy neural network.

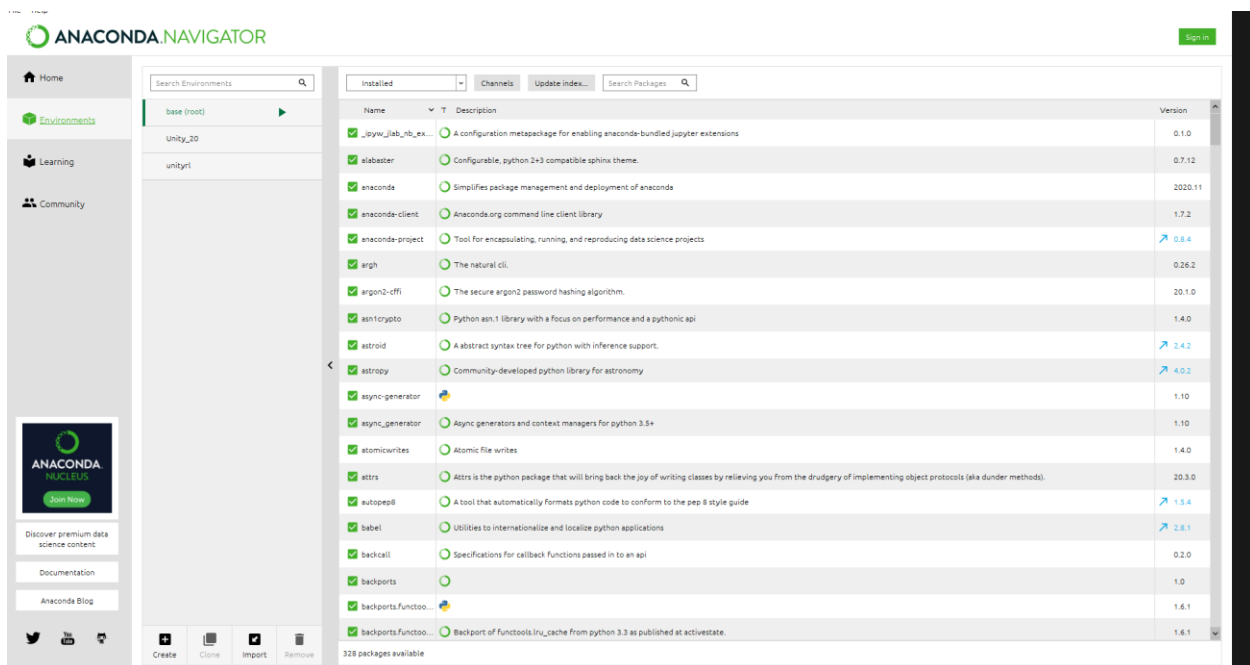
\hat{A} in the equation accounts to the advantage estimate which is the discounted reward – baseline estimate.

By multiplying the log probability of your policy action and the advantages estimate you get, an optimization is created for policy gradient.

I chose a PPO algorithm because I wanted my agent to collect values by observing the environment.

Virtual environments

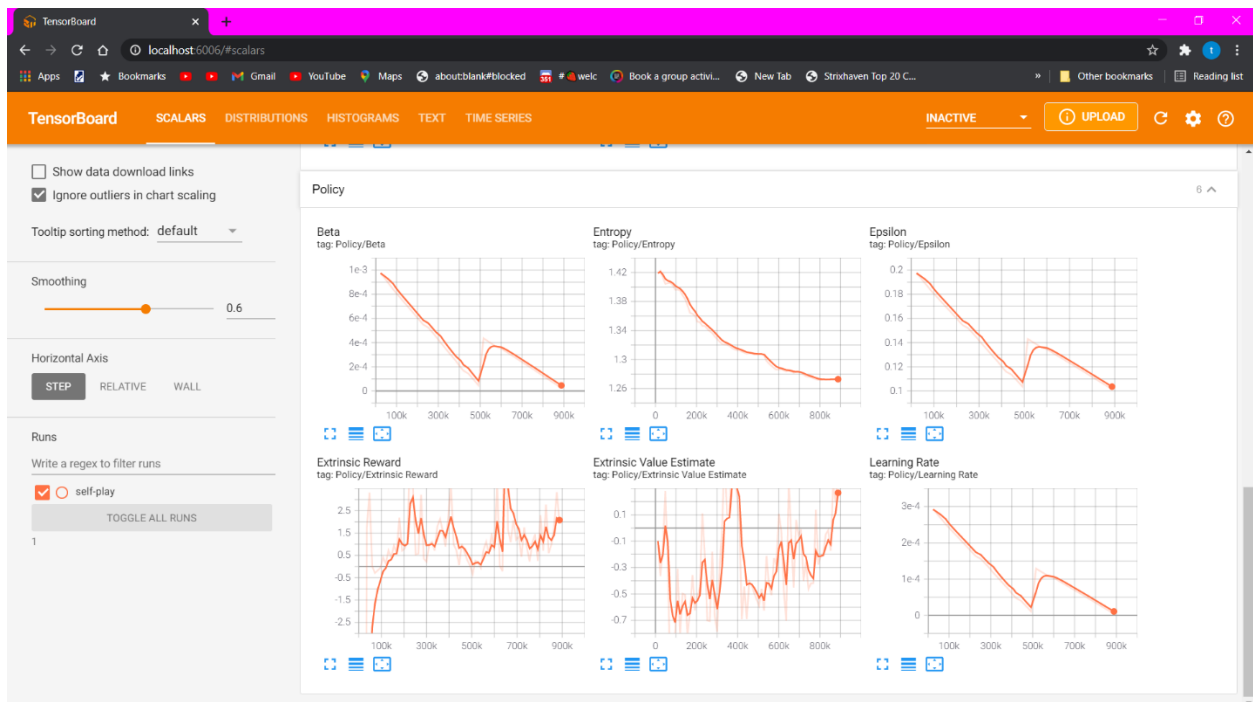
A virtual environments role is to create an isolated environment that has same dependencies as the other projects. For my training process for it was best for me to create a virtual environment in the case of my corrupting the project due to inexperience, my independent project was still to be fine.



During the project I came across some issues due in my virtual environment. The most common error was mlagents package not found. Meaning that I did not install it in the associated virtual environment. Without the mlagents package I was not able to train my agent, with further research and help from other that had the same problem we concluded that I had a virtual environment inside another virtual environment. This was possible because I was using anaconda Navigator to have access to have access to their environment. When I got access to the environments, within the anaconda terminal I then started to create a virtual environment.

PyTorch

PyTorch is a deep learning framework that I was used to represent my training process into an analytical format.



8 Conclusion

Overall, I got happy with my project outcome and have met all the requirement I set for myself at the start of the year plus extra additional features. I also gained some skills technical skill such as C# using a game engine and Reinforcement learning. I learned project management and improved my problems solving skills. Completing this project was very rewarding and hope to create more Machine learning projects in the future.

If I were to tweak my project, I would implement machine learning towards then enemies. I would optimize my agent to be able to learn in different environments.

As I improve in Reinforcement learning I will implement my knowledge and make my player smarter.

9 Appendix

10 References

- [1] What is Gamification, TechnologyAdvice, Youtube,[Online],Available:
<https://www.youtube.com/watch?v=Xa-rvSbT0rQ>
- [2] What is gamification? ,How does gamification work?, BI WORLDWIDE[Online], Available:
<https://www.biworldwide.com/gamification/what-is-gamification/>
- [3]] What is gamification, Gamifly,[Online],Available:
<https://www.gamify.com/what-is-gamification>
- [4] imphenzip, LEARN UNITY – The most Basic Tutorial I’ll Ever Make, YouTube [Online],
 Available: <https://www.youtube.com/watch?v=pwZpJzpE2lQ&t=4058s>
- [5] Alvin Roe, Complete* Unity Beginner Platformer series, Youtube [Online],Available:
https://www.youtube.com/watch?v=rMr1sHQ0_bc&list=PLpj8TZGNIBNy51EtRuyix-NYGmcfkNAuH
- [6] Lost Relic Games, Unity Tutorials, YouTube [Online],Available:
https://www.youtube.com/watch?v=44djgUTg2Sg&list=PLB6BAQR-fTkKdSm_0EzEpRliosI_yw2ni
- [7] Coco Code , Point counter High score and display UI in your game, YouTube
 [Online],Available: <https://www.youtube.com/watch?v=YUcvy9PHeXs&list=WL&index=48>
- [8] Chris’ Tutorial, Testing for Raycast Collison | “d Game Development, YouTube [Online],
 Available:
<https://www.youtube.com/watch?v=epZuI-YgXYo&list=WL&index=63>
- [9] Alexander Ztov, Unity 2D Tutorial About How To Make Enemy To Jump Over Obstacle
 Creating Very Simple AI Gor Game, YouTube [Online],Available:
<https://www.youtube.com/watch?v=Rmuya0hLdIA&list=WL&index=68>

[10] Immerse Limit, Hyperparameters tuning for Unity ML-Agents, YouTube [Online],Available:

[11] Blackhornprod, Making RUN, IDLE & Jump 2D GAME ANIMATIONS, YouTube
[Online],Available: <https://www.youtube.com/watch?v=FTxQKHG5WCA&t=315s>

[12] Is it possible to add a angle to a Raycast 2D, Unity, answers Unity[Online],Available:

<https://answers.unity.com/questions/1240427/is-it-possible-to-add-angle-to-raycast2d.html>

[13] What is Git, Atlassian Bitbucket, [Online],Available:

<https://www.atlassian.com/git/tutorials/what-is-git>

[[14] Github, What is GitHub, Youtube [Online],Available:

<https://www.youtube.com/watch?v=w3jLJU7DT5E>

[15] IBM Cloud Educaation, Machine Learning , IBM [Online],Available:

<https://www.ibm.com/cloud/learn/machine-learning>

[16] Oxford Speaks, What is Machine Learning, Youtube [Online],Available:

https://www.youtube.com/watch?v=f_uwKZIAeM0

[17] Thomas Simonini,An Introduction to Unity ML-Agents, towards data science
[Online],Available:

<https://towardsdatascience.com/an-introduction-to-unity-ml-agents-6238452fcf4c>

[18] 3Blue1Brown,But what is a neural network Chatper1, Deep learning , Youtube
[Online],Available:

<https://www.youtube.com/watch?v=aircAruvnKk&t=51s>

[19] Bot Academy , Improve you're a>l> | Hyperparameters , YouTube [Online],Available:

<https://www.youtube.com/watch?v=6sNIDqgICLY&t=171s>

[20] Proximal Policy Optimization, policy gradient Methods, [Online],Available:

<https://paperswithcode.com/method/ppo>

[21] Arxiv insights, An introduction to policy Gradient methods – Deep Reinforcement Learning , Youtube [Online],Available: <https://www.youtube.com/watch?v=5P7I-xPq8u8&t=582s>

[22] Code Monkey , How to use Machine learning AI in Unity, Youtube [Online],Available: <https://www.youtube.com/watch?v=zPFU30tbyKs&t=2241s>

[23] Real Python , Python Virtual Environment : A Primer, Real Python [Online],Available: <https://realpython.com/python-virtual-environments-a-primer/>

[24] An introduction to policy Gradients with cartpole and Doom, FreeCodeCamp, [Online],Available: <https://www.freecodecamp.org/news/an-introduction-to-policy-gradients-with-cartpole-and-doom-495b5ef2207f/>