Understanding what is machine learning, and what we have learned, and our recommendations

Reference website Justin from Team #3 recommend to use:
https://unity.com/products/machine-learning-agents

- Machine learning is training the machine to learn as it tries to aim for a reward. You give it observation, like human eyes for example and actions it can take. Which then uses these resources you give to go towards the reward or given goal.
- Our team learned that the training takes time and setting up. Think of the AI as it is starting as a baby that knows nothing around the world it is living in - in the unity world space. So you need to define what it can or cannot do based on rewards, and give key observations to look at to process information. Another fun fact would be machine learning only uses 0s and 1s, It doesn't understand anything but that
- It is recommended to learn and go through the tutorial from Code Monkey: https://www.youtube.com/playlist?list=PLzDRvYVwl53vehwiN_odYJkPBzcqFw110
- The playlist above showcases a good introduction to machine learning and how to set it up.
- Another one is taking a look at Unity Learn, as this website provides great knowledge that helps produce this prototype and is useful to get a foundation knowledge of AR in general.
- Unity Learn AR: https://learn.unity.com/pathway/mobile-ar-development
- Tips on setting up Machine Learning on your Windows PC or laptop
    - Aim towards a working Python version, this project is under this version: https://www.python.org/downloads/release/python-3913/
        - You will get into trouble once installing python 3.9.13 to install mlagents aka machine learning agents.
    - One of the troubles can be the right Python version you use to install mlagents, this is for (first thing to set up) mlagents-envs and (second thing to set up after) mlagents folder that will ask to be installed in the command console or powershell console.

        - Open up the release version (above 21, since our project use release 21) that you get from the github (link: https://github.com/Unity-Technologies/ml-agents/releases/tag/release_21)
        - In the folder you have downloaded the release version of mlagents, open up by following this path based on where you set it up or extract it ml-agents-release_21\ml-agents-release_21\ml-agents-envs and ml-agents-release_21\ml-agents-release_21\ml-agents
    - For both folders open up and locate the setup.py and open it in Notepad++ or any other type of software to edit the text inside (recommend Notepad++ or Visual Studio code for easy to read).
    - Inside location this code: `python_requires=">=3.10.1,<=3.10.12",`
    - Add the # part into it, which becomes this: `#python_requires=">=3.10.1,<=3.10.12",`
    - Save the file and you have now forced your way through the problem of installing the wrong version of Python for setup compared to what's needed.
        - As a tip:
            - Don't need to truly worry about which version of Python, but once you pick one, almost everyone who is working on the prototype or game must have the same run to not run into any issues. For this project I recommend Python 3.9.13, the link is above.
            - inside the setup.py, this is used to control what needs to be installed to allow mlagents (machine learning agents) to run in Python with Unity. You can change part of the setup if any version is not being accepted, just double-check online or with Professors with knowledge of setup in Python.

How does the AI train, learn, and the commands for it

- The AI trains by observing based on the tools you give, which can be raycasting. For example, the human body uses its eyes to process information and send it to the brain, the brain will produce that information and produce an action. So think of the AI doing that, however we need to give these ways or tools to observe. In the project there are key

points it looks at, it looks if it can find the new reward target, looks at itself and its velocity so it gains access to this information.

    - You can add more like maybe a ray perception 3D (it's like raycasting rays) to look for the reward basketball.

- The AI will intake the information you give to learn from it. How it learns is by taking any of the actions you give, which is another part of writing in the AI script. This is where you include such as switch statements or other factors such as distance to reward, if it falls out, and other stuff. You include a reward based on its action, or decrease the reward if the AI should do it, like colliding another wall which you create in the collision part of the script. The code to write reward is just AddReward(Add a number float value reward it gets (positive or negative) or SetReward(set the reward value)
    - Aim to set a reward based on actions, so the AI knows what it CAN and CANNOT do. This is important to teach the AI as it progresses in its steps.
    - Another thing to consider is understanding what is EndEpisode(), this defines when it should end what it is doing, and start redoing everything again. It is important that when a reward is given, you can end the episode and keep on going from there. There is a function for OnEpisodeBegin, which is where you can write what it does after the episode begins. (it's like an OnEnable Function)
    - The "public override void CollectObservations(VectorSensor sensor)" is like an update function, important to know that.

How does our Unity Project work for future teams - a basic rundown of a readme text file style

- Inside the Unity Project, you have to run this project in 2022.3.12f1 download from: https://unity.com/releases/editor/whats-new/2022.3.12
- Once installed, open up the unity project if you did download it. Inside there are two main folders to look for.
    - Team #3 Folder
        - This will have the full prototype project including scenes, scripts, materials and prefabs and everything else used by Team #3 2024
    - Machine_Learning and Machine Learning Examples
        - Machine_Learning has the AI brain and yaml file

        - Machine Learning Examples is from mlagents that can run in the project, to run them remember to install ml_agents in Window->Package Manager->ML Agents
    - To run the project, just open either the MainMenu Scene known as MetaMenu (test UI) or BasketballScene, or BasicImageTracking (Test the prototype)
    - Inside the BasketballScene, there are a couple key points to look at and remember:
        - MainBasketballScene is where the scene models and objects are, including training, controlling the basketball, controls and other stuff.
        - UI controls the UI batch of stuff inside this scene.
        - AR Scene is very important and should not be changed or touched unless you need to adjust the AR settings, such as XR Origin and camera or other bits
    - Inside the BasicImageTracking, it just spawns a prefab that you create in the BasketballScene which is where you do your training and testing, and then create a prefab of it into Image Tracking to spawn once you track the image.
        - Once open, look for XR Origin, that location is where you include the prefab, and inside is also how to set up the Image Tracking and what image to follow. Remember to look at the prefab to see the type of prefab being used to create the whole scene for the prefab.
        - Reference which also appears in that same location is the image it will reference to use as image tracking.
        - Another thing to look at is the XR Environment that appears in the scene at the top right under the scene tab. That is the prefab of the whole environment where you can set up where the image will be located for the user to aim at, for example, if aiming forward the image would be flat in this case.

Recommendation on using animation movement instead of physics that our team has used

- To avoid having issues in regards to the agent's movement using physics, it is recommended to use the movement done in animations instead as that'll ensure smooth movements without any heavy mathematical calculations being done

- Aim towards rewarding the AI for any smooth movement, as this will help the AI learn to time each movement, or if you are aiming to keep up with the physics to add a reward when moving smoothly.
- Always start it slow to see how the AI adjusts to the animation, you can then add more complex selections in the switch statement (or another statement of your choice but switch is truly recommended which is in the Unity Project to be seen)
- Use the Gamelab to create motion capture animations, which is known as Mocap.

How to run the machine learning itself, using Python code, how to set up machine learning and or Step-by-step process on running the machine learning and setting up the yaml file.

- I recommend following this website to install machine learning: https://unity.com/products/machine-learning-agents
- You may have questions so here are some answers to help with installing:
    - First, as said above, remember to install the right Python version, however, if using the version this project is using, follow the steps above to brute force to allow installing mlagents folder and mlagents-envs folder. If not installed correctly double check if done correctly, as a way to check is when doing the command like mlagents-learn in the correct folder path
        - If you don't know how to change the folder path, use cd "then write the folder path here after cd" like - cd C:\Users\"UserName"\Downloads\ml-agents-release_21\ml-agents-release_21 and this is for installing, if to train the AI it would or be close to this: cd C:\Users\"UserName"\Documents\GitHub\AR-Mobile-Basketball-Unity-Project-Team--3\Assets\Machine_Learning
    - How do we change the settings on the yaml file or what is it and how to use it for training AI?
        - Currently the file is named in the project as configuration.yaml which is located in Assets\Machine_Learning\AI_Learning_Player. When opening up you can open it with notepad, notepad++ or visual studio code (you can use other softwares). Inside there are a couple main factors to look at, checkpoint_interval, max_steps and summary_freg.

        - checkpoint_interval: is mainly used to tell the AI to save data at that step or close to it, and start brand new but gain a bit of knowledge from previous generations (which the first batch is gen 0, the next is 1 and so on base on the number you include compare to max steps in checkpoint_interval)
        - max_steps: is used as the total amount of steps that can be done. For example, once it reaches the current state which is 5000000, once it reaches that point it will end and you must start brand new.
        - Summary_freg: this is important to understand what the AI has learned, its mean reward (- value) or the other value which is the positive reward (+ value), you want to reduce the negative value based on the positive value. This is important because you can check when to see the progress of the AI
- To run the training, as said above, the command is mlagents-learn, if you want to resume it can be mlagents-learn --resume however to connect to yaml file, as you want to use the yaml you change in the settings you must do, for this case: PS C:\Users\"UserName"\Documents\GitHub\AR-Mobile-Basketball-Unity-Project-Team--3\Assets\Machine_Learning> mlagents-learn .\AI_Learning_Player\configuration.yaml
- Then you hit enter and it will run based on the settings, if you want to resume, also do the same and write --resume right beside it, or the location of the yaml file.
- If you need help, write mlagents-learn --help to find the commands. Remember to hit play once you see the Unity Logo that appears after writing the command
- Reference using this youtube video, it's super good to get a good idea on mlagents and the commands or understanding examples and how it all works: https://www.youtube.com/watch?v=GhS6-vvhOy8
- Last thing to learn and to understand is the scripts to run for mlagents, here are some to remember or take a look at:
    - Blocker Agent (Script)

- This is the AI script, which you can create your own, but must include these part to run as an AI script:
    - Blocker Agent (Script)

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Unity.MLAgents;
using Unity.MLAgents.Sensors;
using Unity.MLAgents.Actuators;

Unity Script (3 asset references) | 2 references
public class BlockerAgent : Agent
```

```
public override void OnEpisodeBegin()
{
    // If the Agent fell, zero its momentum
    if (this.transform.localPosition.y < -1.852f)
    {
        this.rBody.angularVelocity = Vector3.zero;
        this.rBody.velocity = Vector3.zero;
        this.transform.localPosition = new Vector3(2.23861051f, -1.62899995f, 1.52878571f);
    }
}
```

```
0 references
public override void CollectObservations(VectorSensor sensor)
{
    // Target and Agent positions
    Target = GameObject.FindWithTag("Basketball");

    Debug.Log("Target Local Position z");
    Debug.Log(Target.transform.localPosition.z);

    if (rBody.velocity.y < 0)
    {
        rBody.velocity += Vector3.up * Physics.gravity.y * (fallMutiplier - 1) * Time.deltaTime;
    }

   // Debug.Log("Distance To Target");
   // Debug.Log(distanceToTarget);
   // Debug.Log("localPosition of this AI");
   // Debug.Log(this.transform.localPosition);
   // Debug.Log("localPosition of Target");
   // Debug.Log(Target.transform.localPosition);

    sensor.AddObservation(Target.transform.localPosition);
    sensor.AddObservation(this.transform.localPosition);

    //Agent velocity
    sensor.AddObservation(rBody.velocity.x);
    sensor.AddObservation(rBody.velocity.y);
    sensor.AddObservation(rBody.velocity.z);
}
```

```
public override void OnActionReceived(ActionBuffers actionBuffers)
{
    // Actions, size = 2
    //Vector3 controlSignal = Vector3.zero;
    float moveX = actionBuffers.DiscreteActions[0];
    //controlSignal.x = actionBuffers.ContinuousActions[0]; // 0 = don't move; 1 = left; 2 = right
    float moveZ = actionBuffers.DiscreteActions[1];
    //controlSignal.y = actionBuffers.ContinuousActions[1]; // 0 = dont move; 1 = back; 2 = forward
    float moveY = actionBuffers.DiscreteActions[2];
    //controlSignal.z = actionBuffers.ContinuousActions[2] still defining - 0 = do nothing; 1 = jump once;
    //rBody.AddForce(controlSignal * forceMultiplier);
    Vector3 addForce = new Vector3(0, 0, 0);
```

```
0 references
public override void Heuristic(in ActionBuffers actionsOut)
{
    var continuousActionsOut = actionsOut.ContinuousActions;
    continuousActionsOut[0] = Input.GetAxis("Horizontal");
    continuousActionsOut[1] = Input.GetAxis("Vertical");

}
```

Explaining: These are important in every AI script, as they are part of the brain to allow the AI figure out what to do based on tools you give it.

- Behavior Parameters

        - Is part of the AI script you create, as that is also part of the brain, it defines what is happening during the AI script when it is running, such as if it should look at continuous action or discrete branches, currently this project uses discrete branches. You can hover to get an idea what each part does.
    - Decision Requester
        - How many times it can decide a decision per frame, the more the fast the AI will decide, due to the amount of decisions it can do, and if you allow the AI to take actions between decision

Recommendation on how to train the AI in the future

- We think it would be better to first train the AI using imitation learning to give it a basic idea of what it should do
- Once it has the basics down, it can start using reinforcement learning to improve what it learned with imitation

Basic written notes on what each script does

Old Scripts

- Just old scripts are no longer being used

AutoShoot

- A Script to automatically shoot the basketball, the location is in the Training
- Go to BaseCannon(ForMachineLearning) to test out the auto shooter

BlockerAgent

- AI script to be used and can be used in future of this project

Controller

- Basic control to jump - can be used in the future of this project for the player to play as the blocker

IgnoreWall

- A script that ignores walls, these walls can be the ones that are keeping the AI inside, but you want the basketball to go through.

MenuScript

- For the UI, like switching scenes, enable or disable GameObject UI

PanelScreenHow (funny name)

- For the main menu UI scene in MetaMenu

PauseMenu

- Functions for Pause Menu Buttons

ScoreColliderScript

- This is for the Main use of the game when the player is the one shooting, this is if the ball has collider and score, if manage to collider, spawn a new basketball, if not they don't get a point

ScoreForReward

- This is used for AI training, to learn if AI has score or not, used if training, not the ScoreColliderScript

SpawnBall

- This is also connected to allow and learn what happened to the basketball and respawn it after a certain amount of time.

ThrowBall

- Main script to throw the basketball, follow the AR camera, allow to shoot and other stuff

Timer

- This is used for the timer, and to end the game if the timer has finished, no loading scene, or game over scene, or final score scene.