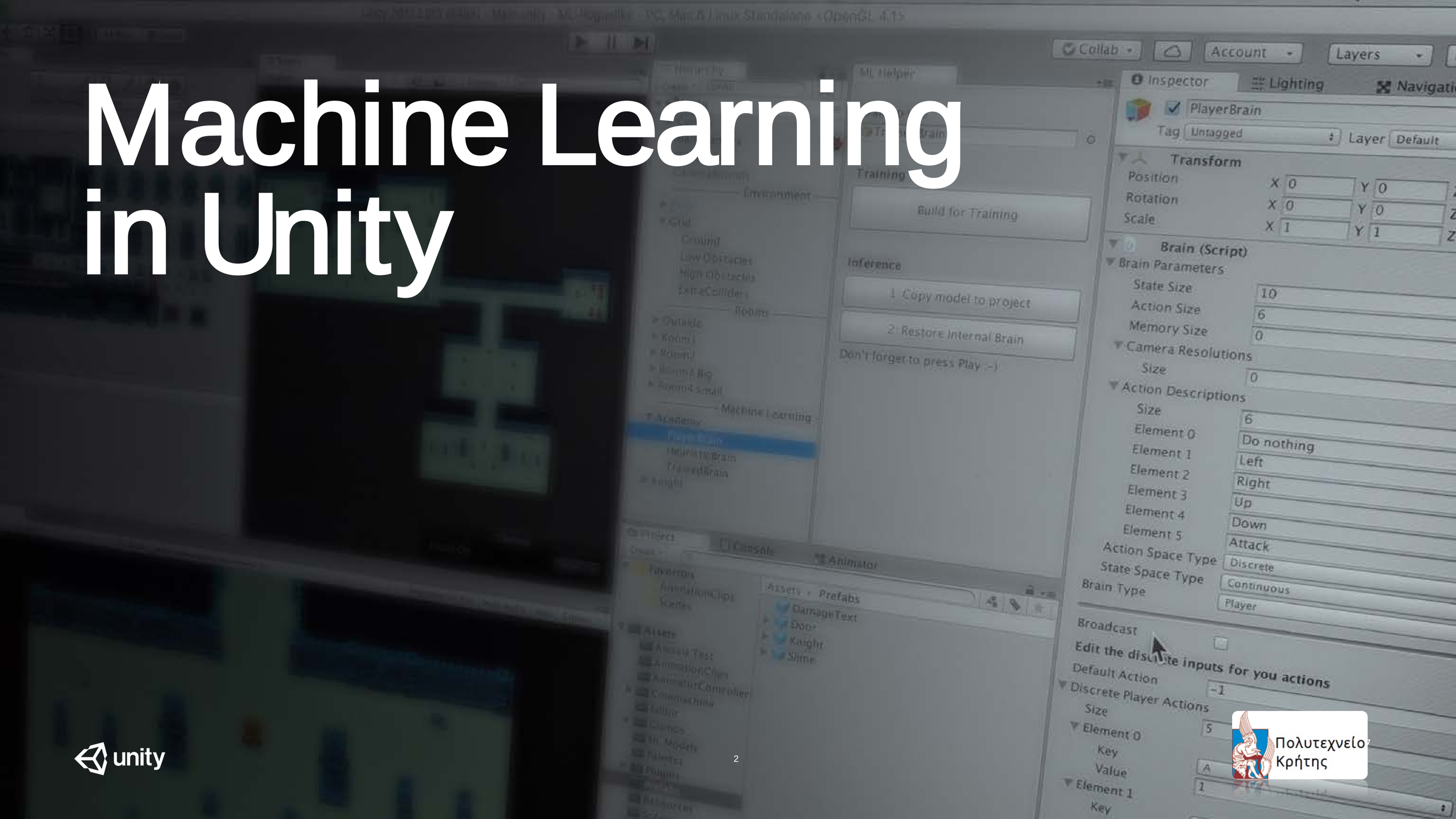
- Alexandros Michael | Αλέξανδρος Μιχαήλ
- 2014030077
- mmichail@isc.tuc.gr

Machine Learning in Unity

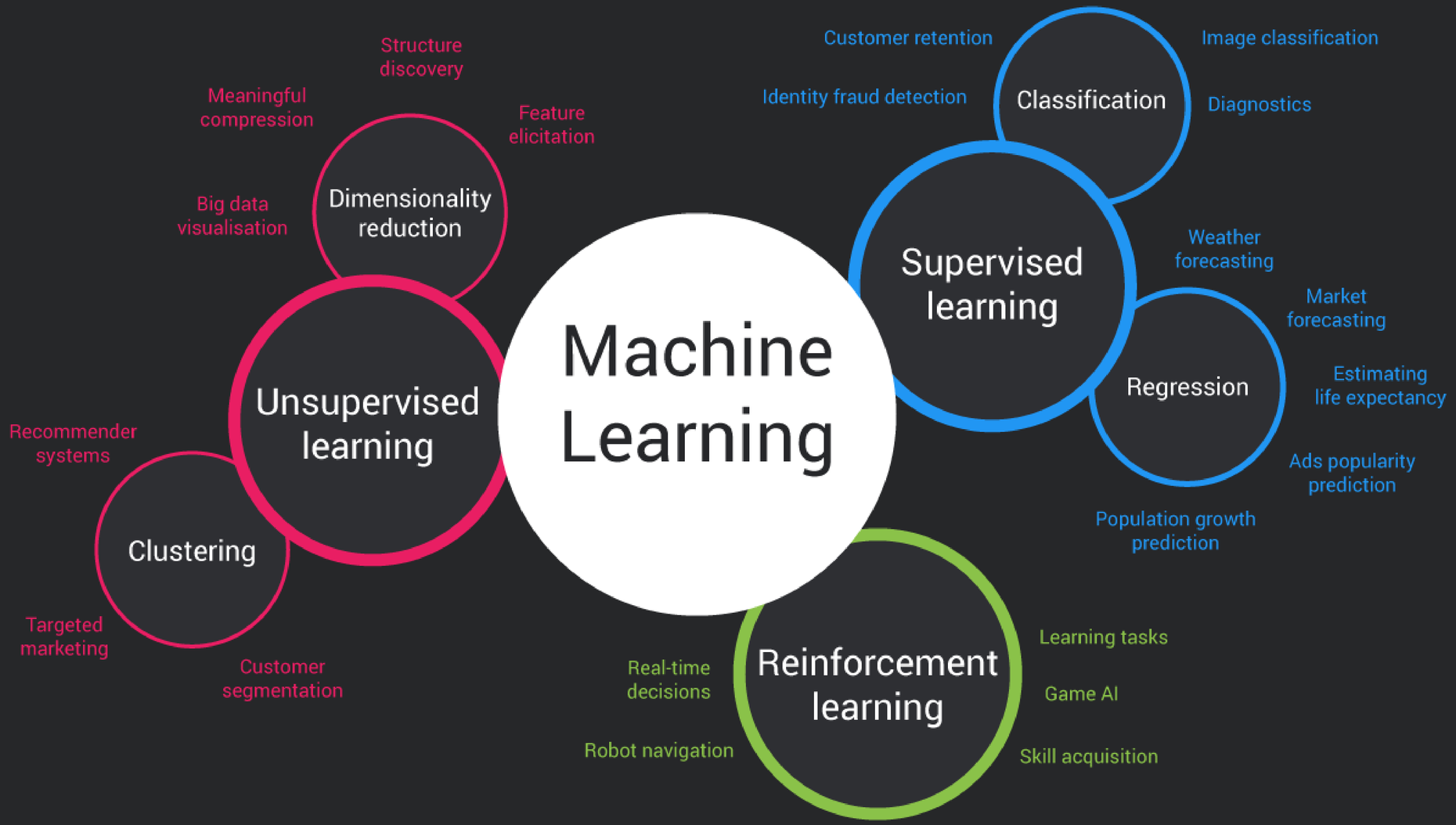


unity

Πολυτεχνείο Κρήτης

What is Machine Learning?

What is Machine Learning



Machine learning is an application of artificial intelligence that provides systems with the ability to automatically learn from experience without being explicitly programmed.

The process of learning begins with observations, in order to look for patterns and make better decisions in the future based on the examples that we provide.

There are different machine learning algorithms that can be adopted, and the one we will look at today is Reinforcement Learning (which is down there, in green).

unity

Πολυτεχνείο Κρήτης

Reinforcement Learning

Reinforcement Learning is a learning method where agents interact with their environment by producing actions, and getting punishments or rewards in exchange.

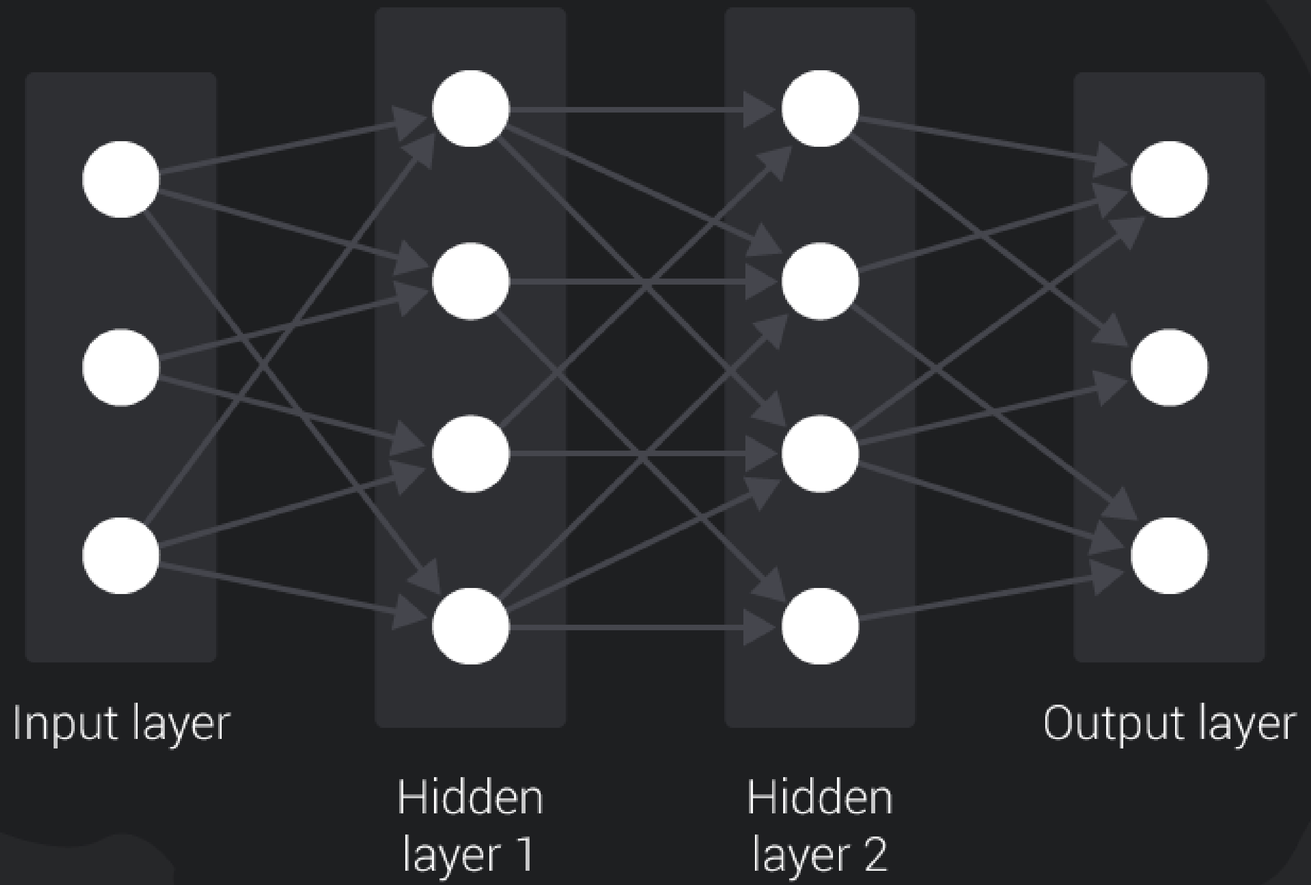
Trial and error search is the most relevant characteristic of reinforcement learning. This method allows agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. The feedback received is called reinforcement signal.

It’s like we’re teaching our dog to sit down when we ask, and we give a treat if they actually do it. They might not understand our language, but they link an action they perform with some form of reward that follows it directly.

unity

Πολυτεχνείο Κρήτης

A computer system
modelled on
the human brain and
nervous system



Neural Networks

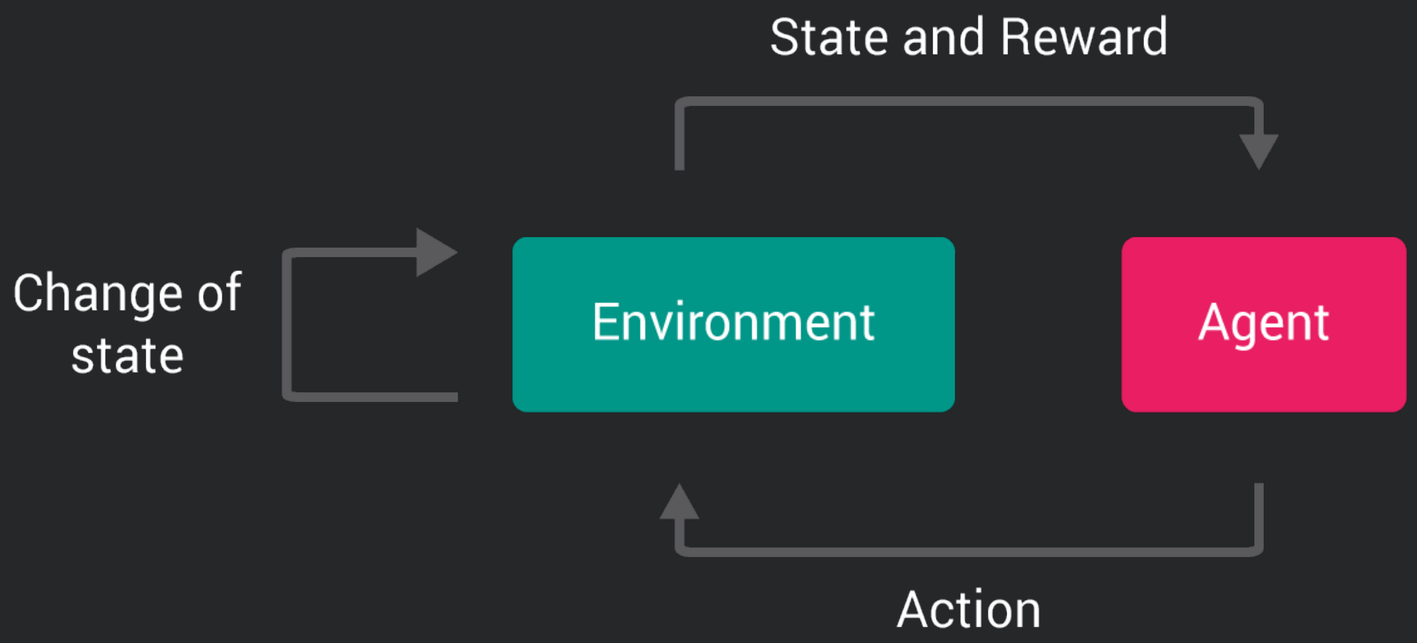
Neural Networks

To perform learning we use computer systems modelled on the human brain, called Neural Networks,

A typical neural network has anything from a few dozen to millions of artificial neurons called “units” arranged in layers. Some of them, known as “input units”, are designed to receive various forms of information from the outside world. Other units sit on the opposite side and provide a result, and they are known as “output units”. In between are one or more layers of hidden units, which is where all the calculations happen.

Most neural networks are fully connected, where every unit in one layer connects with all units on another layer. These connections are represented by a “weight”, which can be either positive (if one unit excites another) or negative (if one unit suppresses or inhibits another).

Reinforcement Learning



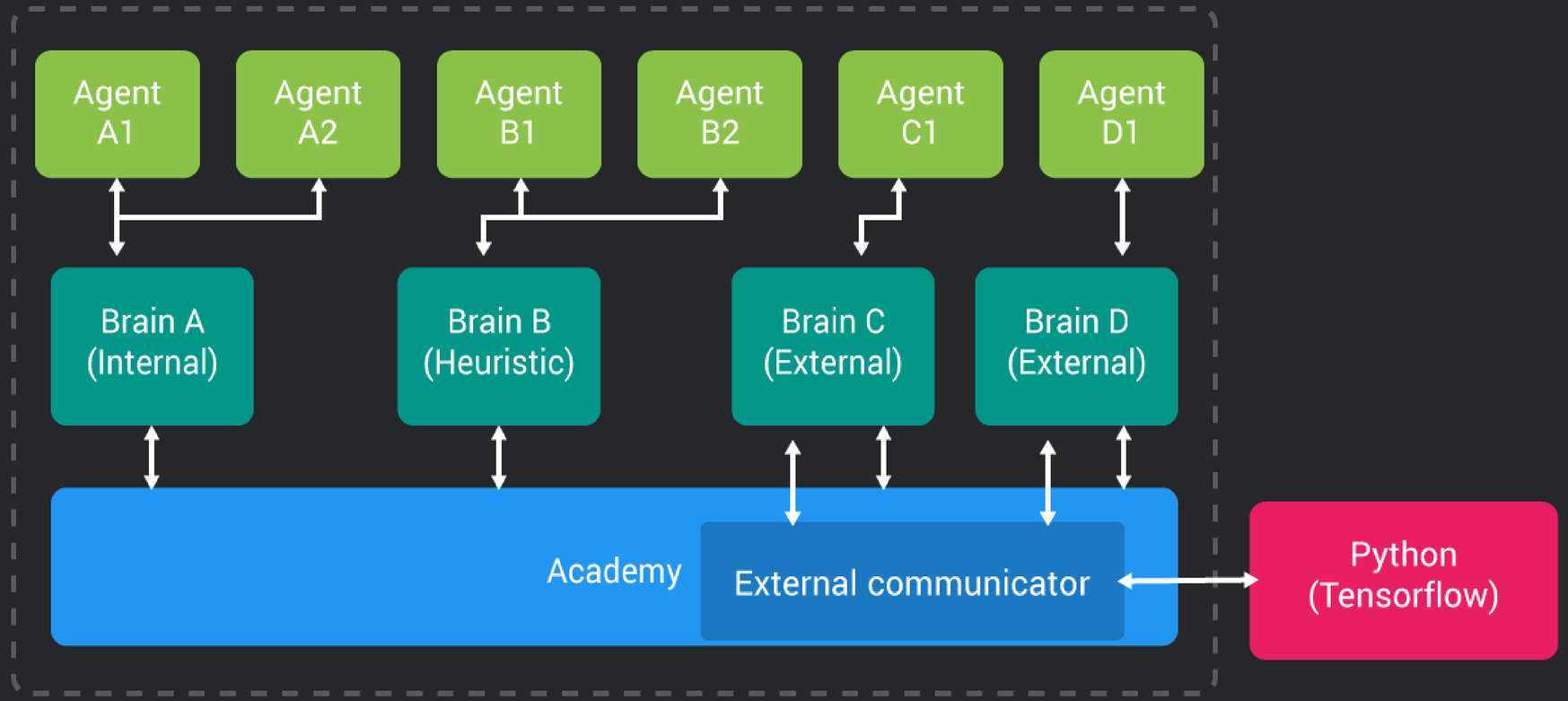
- ML Agents use the Reinforcement Learning model. At its center there are the Agents who perform actions on an Environment. The actions provoke a change of state in the environment, and that in turn is fed back to the Agent, together with some reward.

Learning Environments

And this is the grand picture. The action happens in what we call the Learning Environment - here marked by the dotted line - which is basically a normal Unity scene.
At the bottom we have the Academy (in blue) which is a script that defines the properties of the training: framerate, timescale, and other useful things.
The Academy passes these parameters to the Brains (in the central row, in green), who are the entities that contain the model to train. As you can see there are various types of Brains.

At the top of the diagram we have the Agents which, as you can see, can be more than one per scene, and more than one per Brain.
Each Agent refers to a single Brain, from which it gets the actions and feeds back information so that the Brain can learn.

When we export a Unity build to train the Agents, there's an extra module that allows the Brains marked as External (Brains C and D) to communicate with an external environment coded in Python using a library called Tensorflow.
Once training is over, this environment will distill the learning process into a "model", which then gets reimported in Unity to be used by Internal Brains (like Brain A).



unity Unity Learning Environment

Πολυτεχνείο Κρήτης

Examples

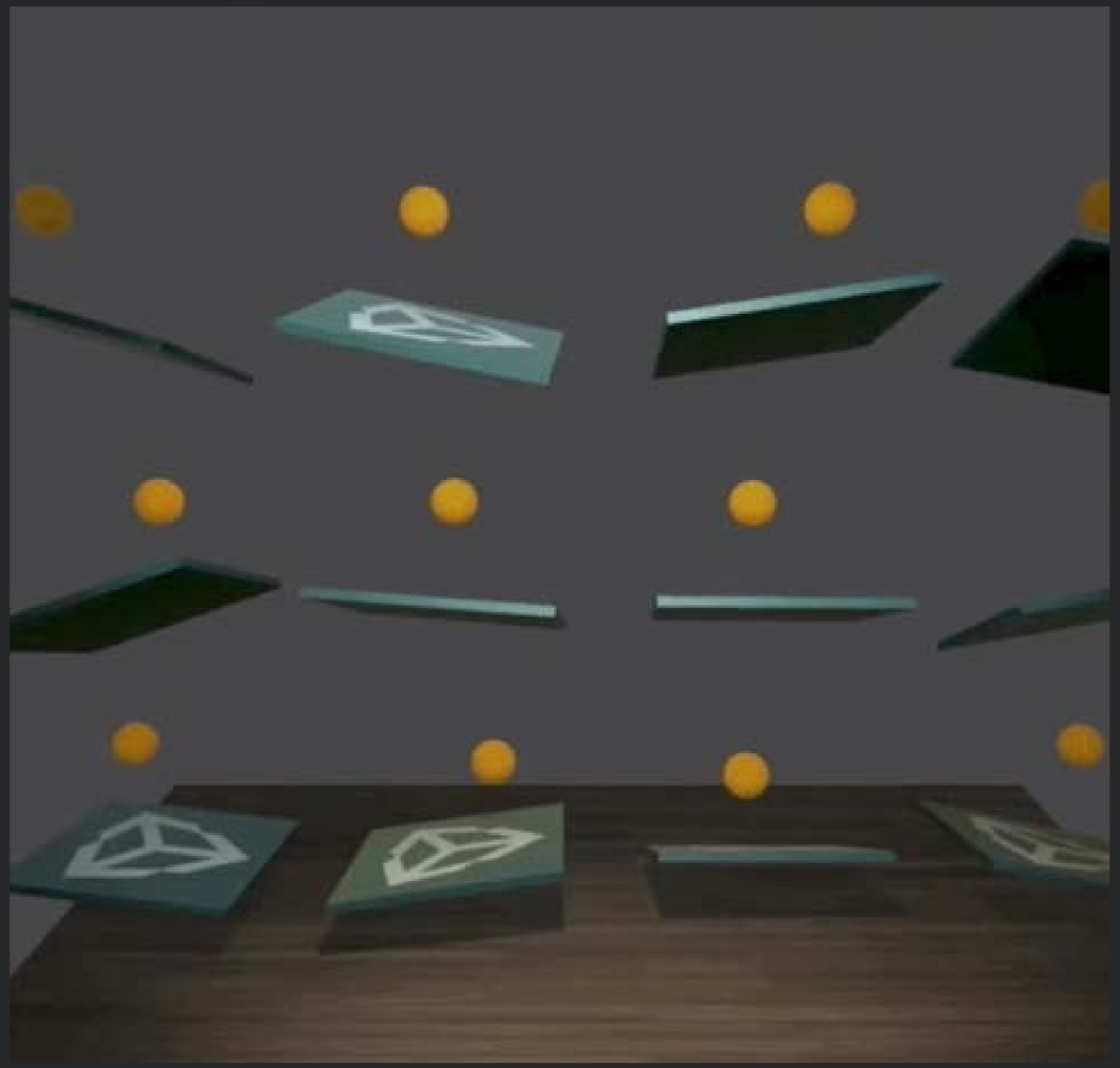
3D Ball

Goal:

Balance the ball on the platform

Reward:

- +0.1 for every frame the ball remains on the platform
- -1.0 if the ball falls from the platform



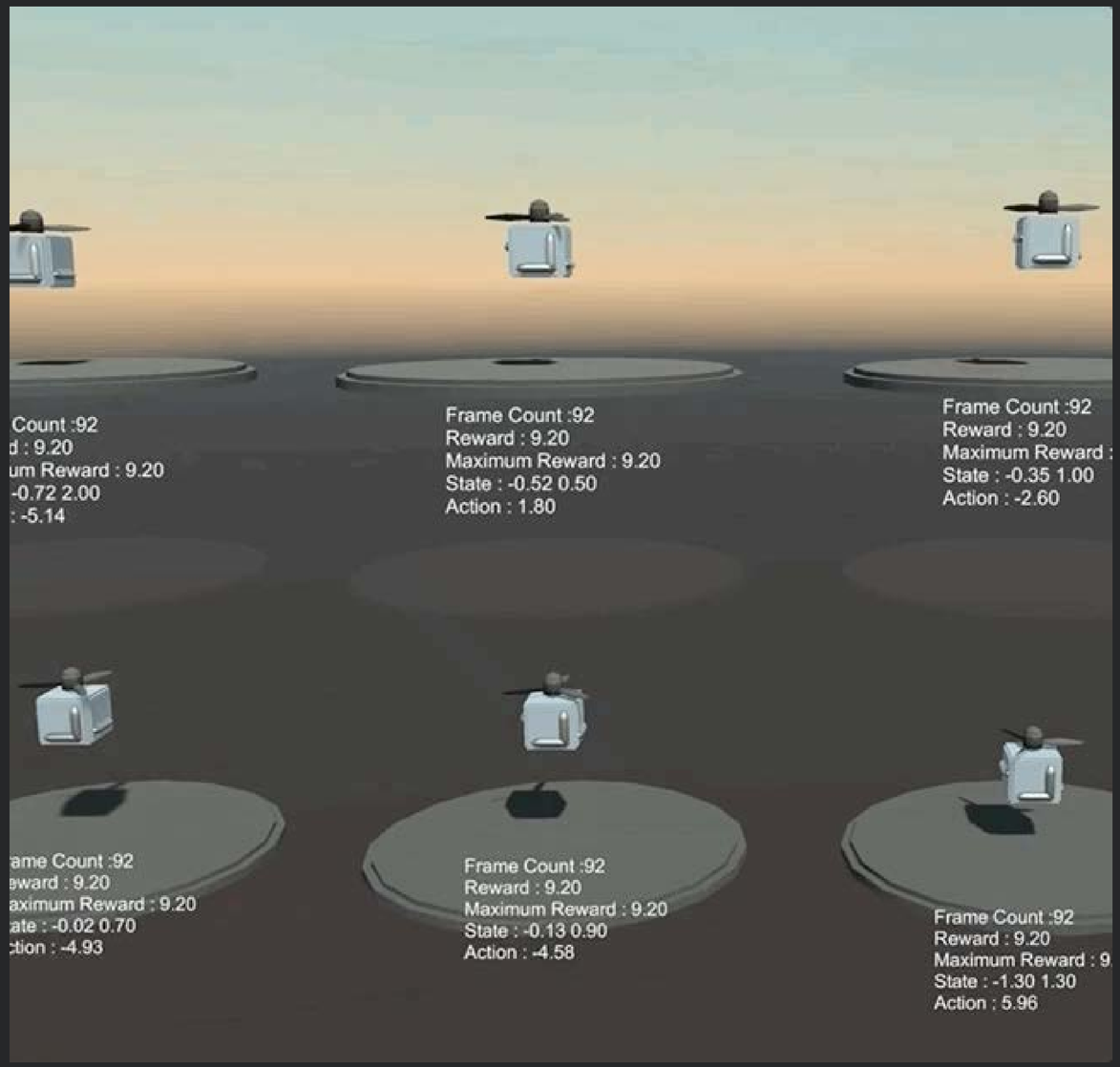
Propellers

Goal:

Have the cubes learn to float

Reward:

- +0.1 for each frame the cube floats
- -1.0 for each collision with the floor



Arena

Goal:

Push the crate out of the arena

Rewards:

- +0.2 for if closing on the crate
- +0.5 when crates gets further from the center
- Neg. rewards for delaying, or falling

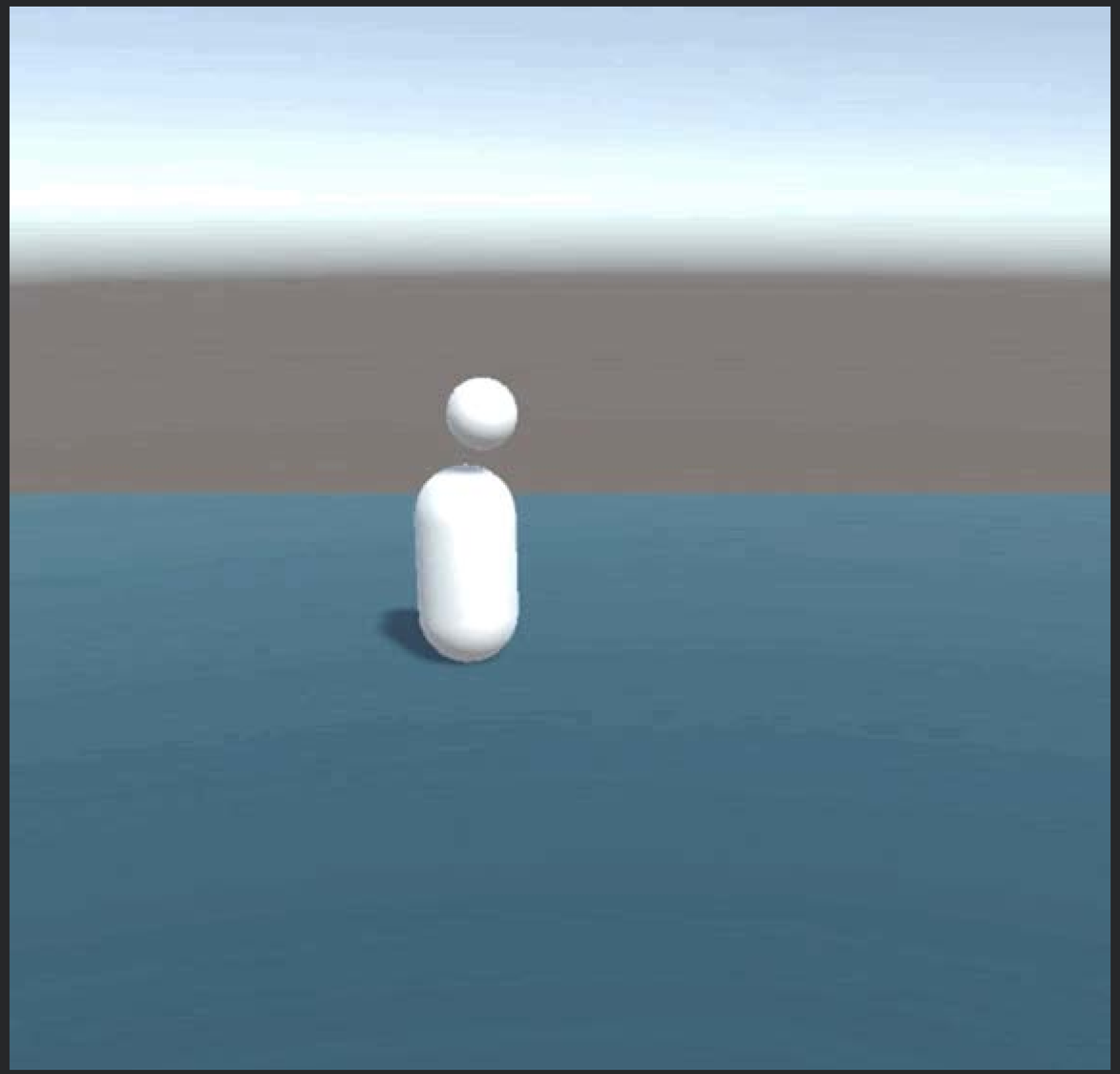
Bounce Ball

Goal:

Bounce ball on top of agent’s head

Reward:

- +0.1 for each frame the ball is closer to the agent
- -0.1 for each frame the ball is further away from the agent



Can you use Machine Learning in a real game?

Score: 20
Steps: 908
Hazards: 5
Counts: 7

Space Shooter

Unity Classic Game To Test On!

unity

Πολυτεχνείο Κρήτης

Ingredients

- A simple action game
- All entities are Agents, both the player and the enemies
- Establish a common “interaction language”
- The goal is survival, while attacking other entities

Design and ideas

- What are the game actions
- What you want the Agent to learn
- What’s right or wrong (what to reward)

Discrete vs. Continuous

Discrete means that the States/Actions can only have one value at a time. Like an Enum. It's either 0, or 1, or 2, or 3, etc.

- Easier: Agents associate actions with rewards more easily

In Space Shooter, we use Discrete for Actions. It can have 6 values:

0: Stay still / **1-4:** Move in one direction / **5:** Attack

Discrete vs. Continuous

Continuous means you can have multiple States (or Actions) and they all have float values.

- They require more memory for training (hyperparameters)
- Hard to use: they can confuse the Agent

In Space Shooter, we use Continuous for States:
Move,Fire at ship,Fire at asteroid,none…

The pseudo-algorithm (AgentStep)

If Shooter is not moving towards the center
Then Punish
Else Reward

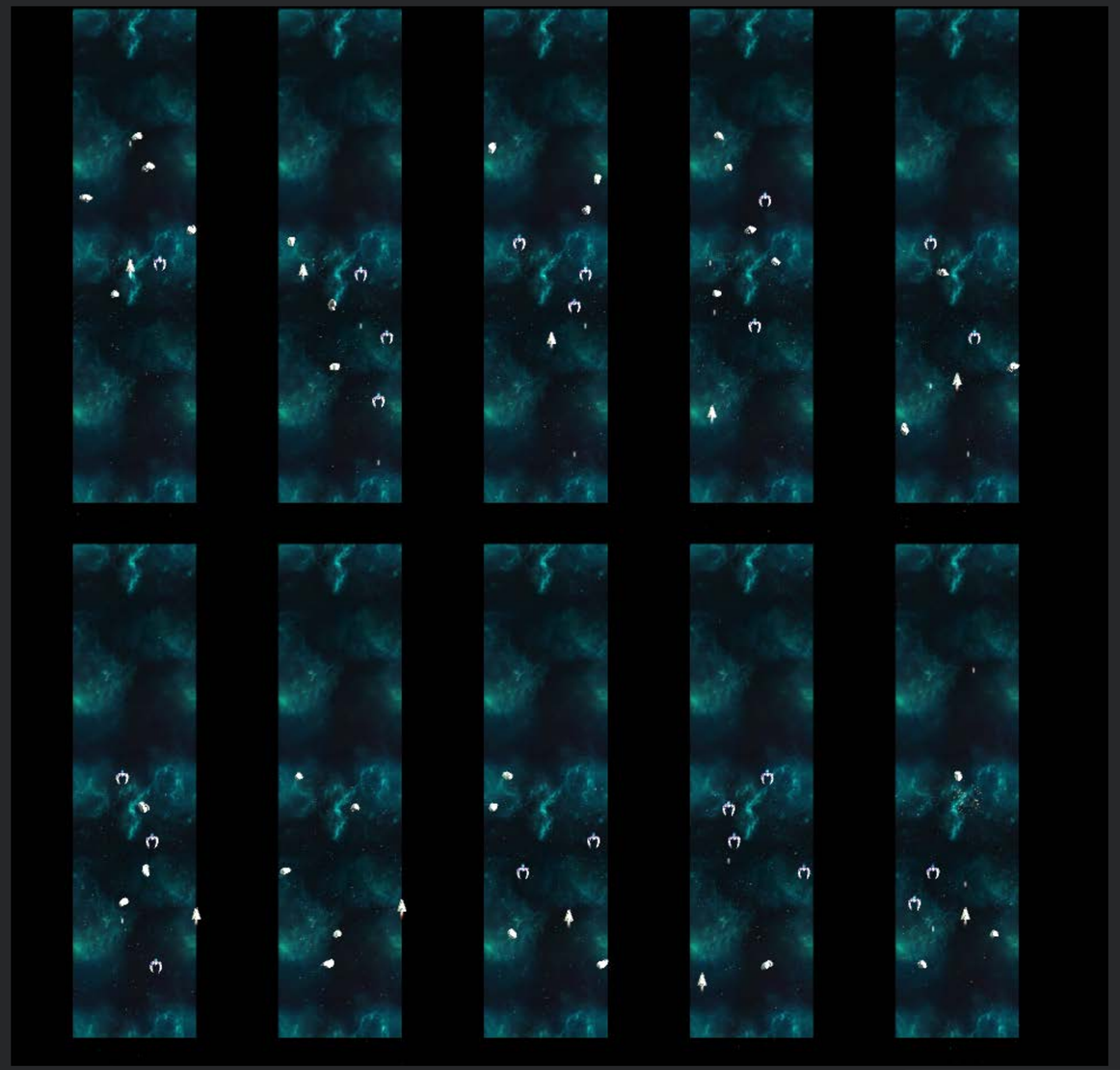
Movement

If IncomingThreat coming towards shooter
then Start attack
Reward
Else if shoot at nothing then
Punish

Attack

Training scene

- Position and configure the agent(s)
- Connect them to the relevant Brains
- Configure the Academy



Final Product

When we apply this model to our Space Shooter game, we can see that it works the same way. Our shooter has learnt how to act intelligently and to adapt to scenarios using Machine Learning !