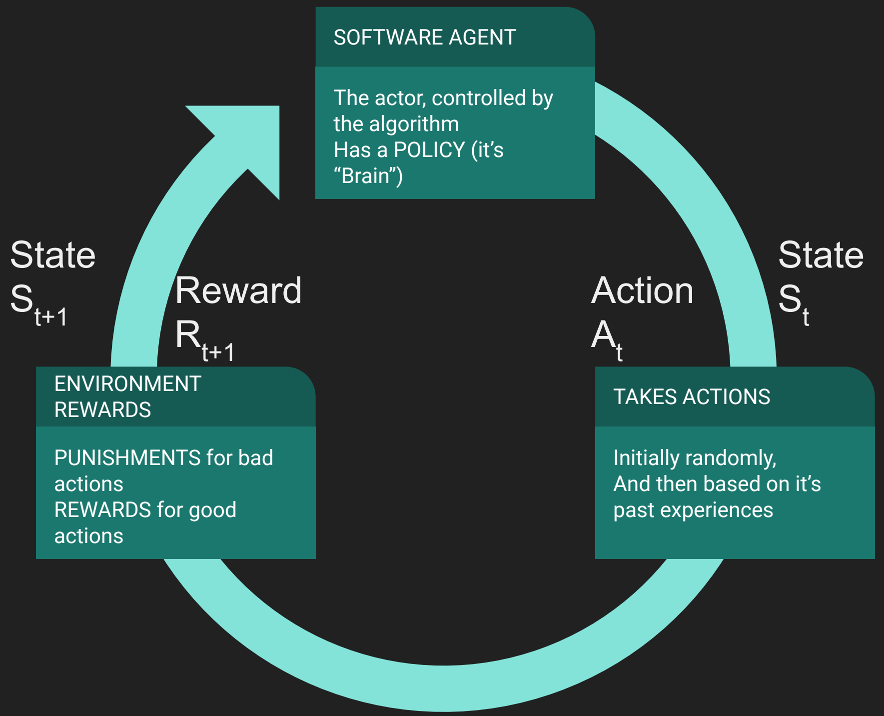
Reinforcement Learning Using Unity ML-Agents

Group 12

Xinyue Sui
Parthiv Shah
Pranav Mujumdar

Reinforcement Learning?

- Area of machine learning
- One of the basic three branches of ML, alongside Supervised and Unsupervised learning
- Inspired by behaviourist psychology
- Conceptually probably the easiest method of ML to grasp
- Deals with decision making, in order to get the most rewards



Unity Engine

- Gaming engine, popular with indie game developers
- C# based construction
- Convenient tool for programmers
- Helps develop and simulate 2D and 3D "Environments"

ML Agents

- Python based Toolkit for RL
- Provides connection between data collected in Unity and ML algorithms
- Helpful in building a Neural Network using Tensorflow
- Real-time Visualization using Tensorboard

Proximal Policy Optimisation (PPO)

- “On-Policy” Algorithm developed by OpenAi, released in 2017
- Their Dota 2 bot defeated the best human players
- PPO strikes a balance between supervised learning and reinforcement learning
- Easy to implement, compatible with gradient descent
- Easy to tune the hyperparameters

OpenAI

Soft Actor Critic

- Developed and Released by BAIR lab in 2018
- Central feature is Entropy Regularization
- Off-Policy Algorithm
- No Sensitive Hyperparameters

BAIR

BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Key Concepts

- Episode
  - Instance of training
  - Each episode must be similar
  - Example, the crashing bird in “Flappy Bird”
  - Each episode starts again from “zero”
  - Based on the problem the episode length changes
  - We can limit each episode for certain number of steps
- Rewards
  - Positive, if the action is GOOD
  - Negative, if the action is BAD
- GAIL (Generative Adversarial Imitation Learning)
  - Provide demonstration to the AI
  - So that it can learn faster

Flappy Bird Implementation

- Simplest environment to create and understand
- Environment
  - Agent has to stay alive for as long as possible
- Bird Agent: has two “discrete” actions
  - Jump or Flap its wings
  - Don’t Jump
- Rewards:
  - Punishment for crashing on the obstacles, and end of the episode (-1f)
  - Reward for staying alive and making decisions (+0.1f)

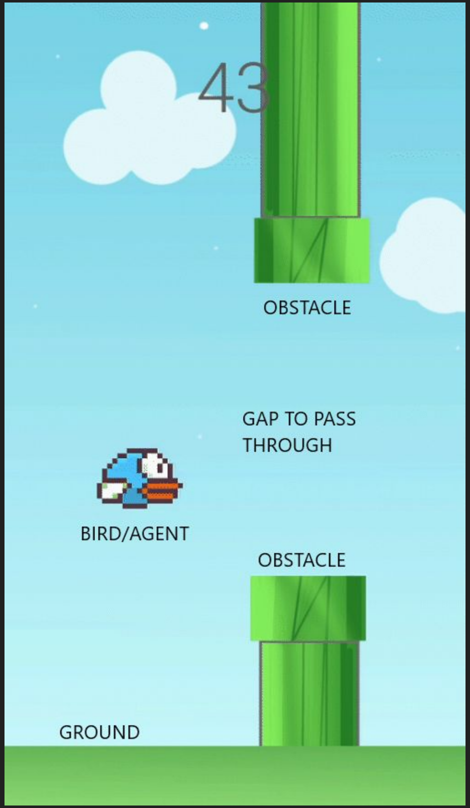
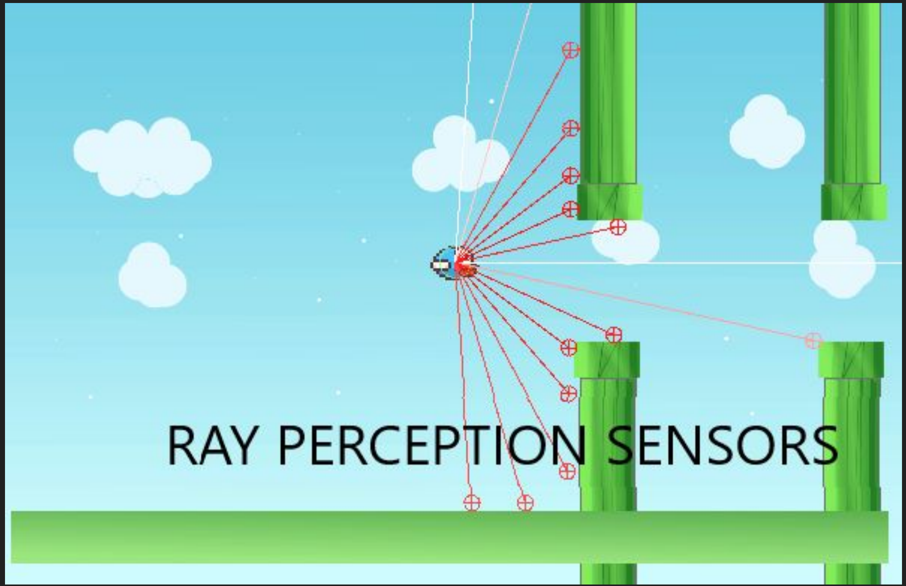


Image.1 Screen capture from our project

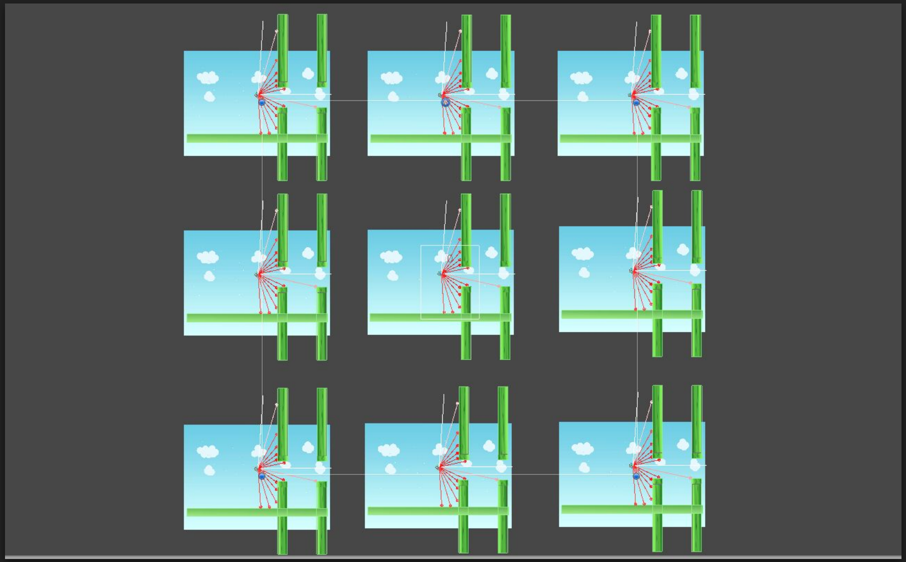
How does the agent understand the environment?

- Using Ray Perception Sensors
- Collect data about the distance from the obstacle
- No information about the nature of the object in the proximity, only the tag can be identified by the agent
- The agent learns whether this object is good or bad by trial and error

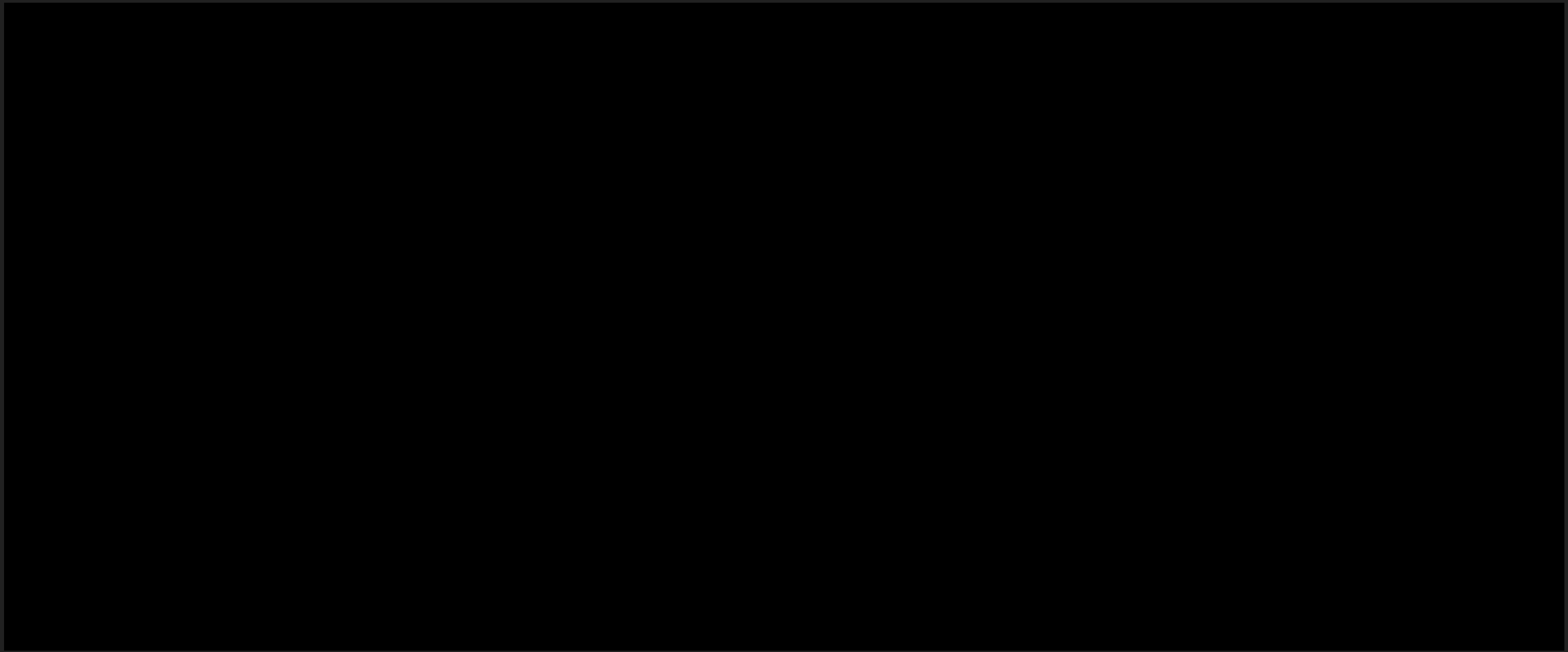


Time to train!

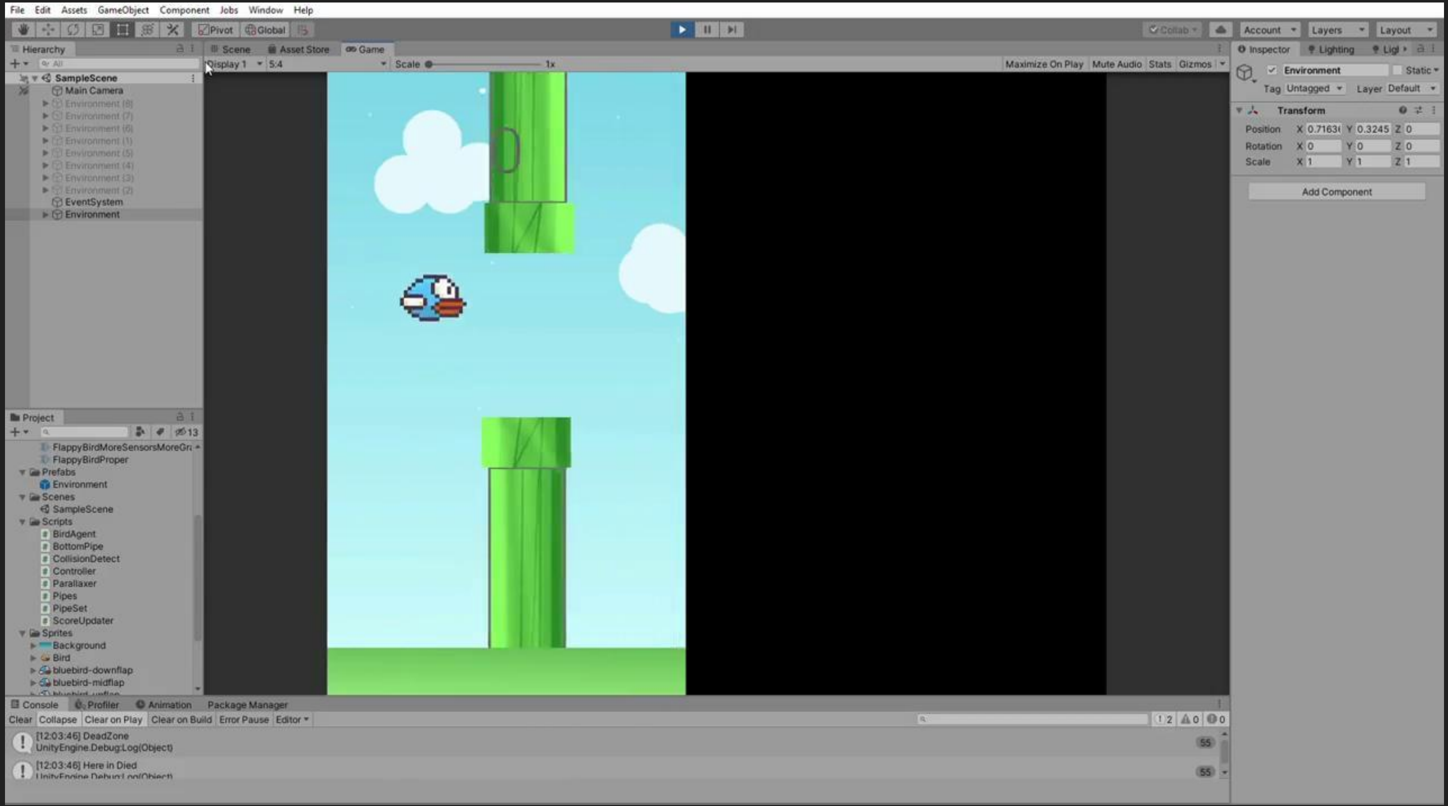
- What is better than One agent?
- Multiple agents that contribute to the same policy
- Faster Training
- We replicated the same environment 9 times
- All agents contributing to the same policy or the “Brain”



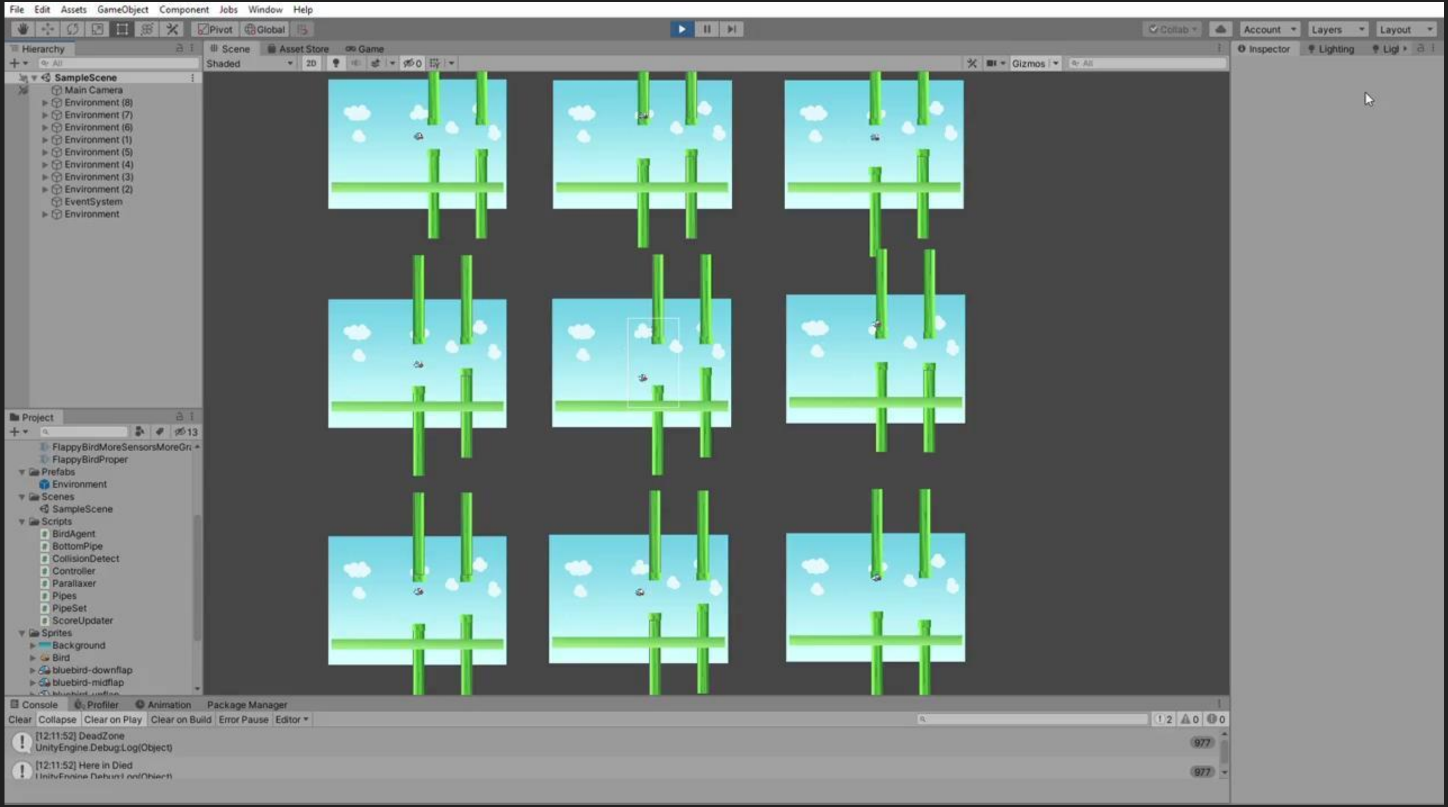
Now let’s start training!



Let’s slow it down to understand whats happening

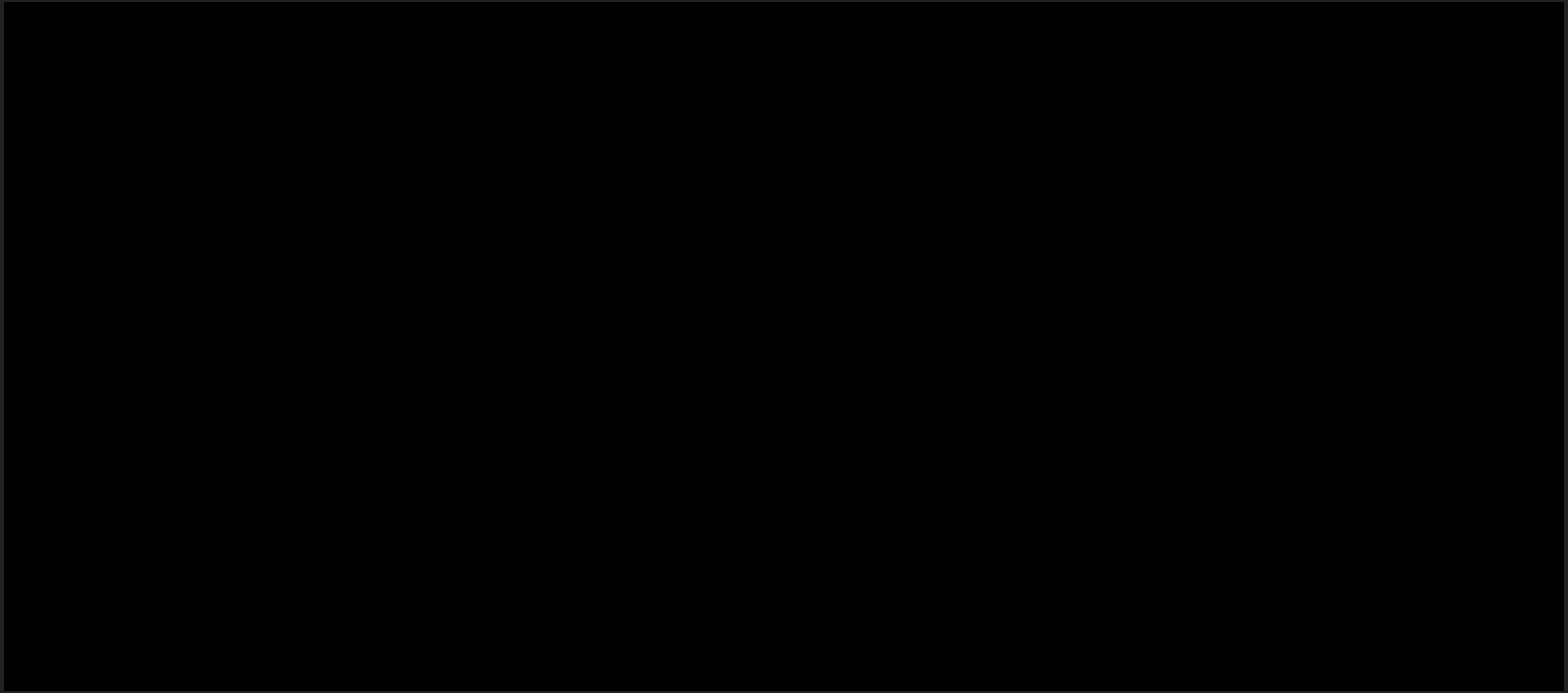


Let it train



File Edit Assets GameObject Component Jobs Window Help
Hierarchy
SampleScene
Main Camera
Environment (8)
Environment (7)
Environment (6)
Environment (1)
Environment (5)
Environment (4)
Environment (3)
Environment (2)
EventSystem
Environment
Scene
Asset Store
Game
Inspector
Lighting
Project
Prefabs
Scenes
SampleScene
Scripts
BirdAgent
BottomPipe
CollisionDetect
Controller
Parallaxer
Pipes
PipeSet
ScoreUpdater
Sprites
Background
Bird
bluebird-downflap
bluebird-midflap
Console
[12:11:52] DeadZone
UnityEngine.Debug:Log(Object)
[12:11:52] Here in Died

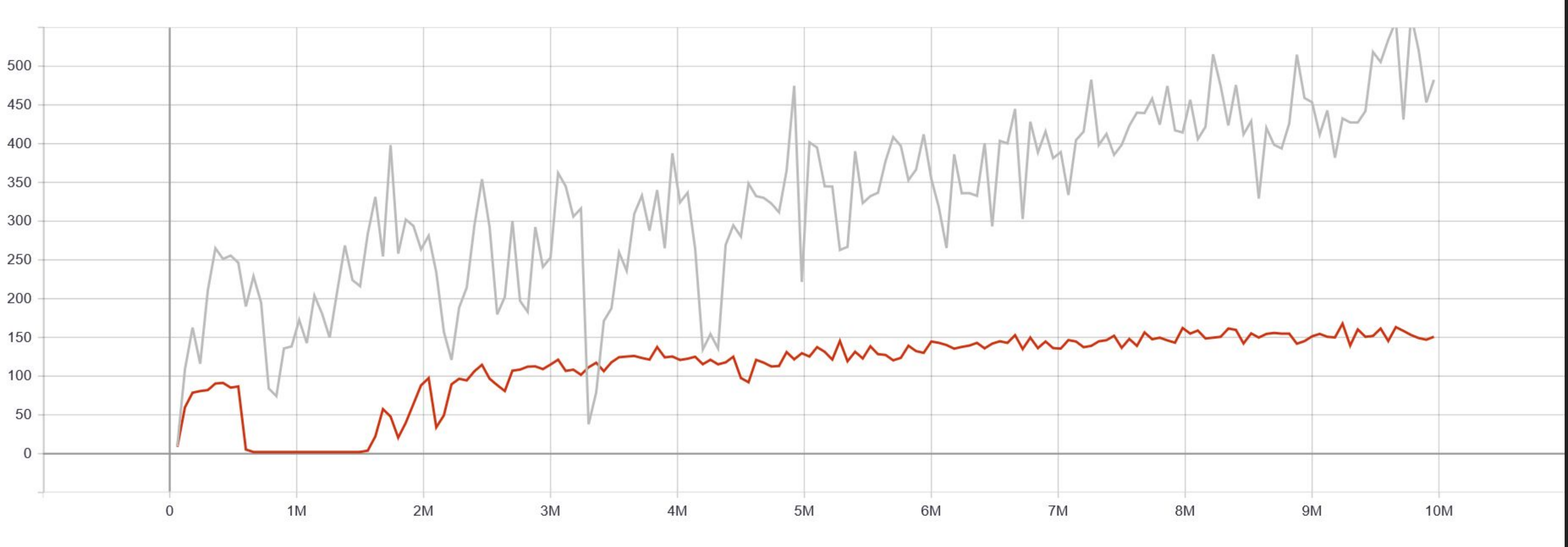
Halfway through the training



Final trained Model

- Use the final trained model as a “.nn” file that we can use for performing inference
- Rewards (Must Increase)
  - At 60k Steps: 9.618
  - At 9.96M Steps: 482.3
- Episode length(Must increase)
  - At 60k Steps: 20.46
  - At 9.96M Steps: 951.5

Cumulative Reward Graph



It’s after all just a simple game!

Real world problems are a lot complicated and difficult to simulate

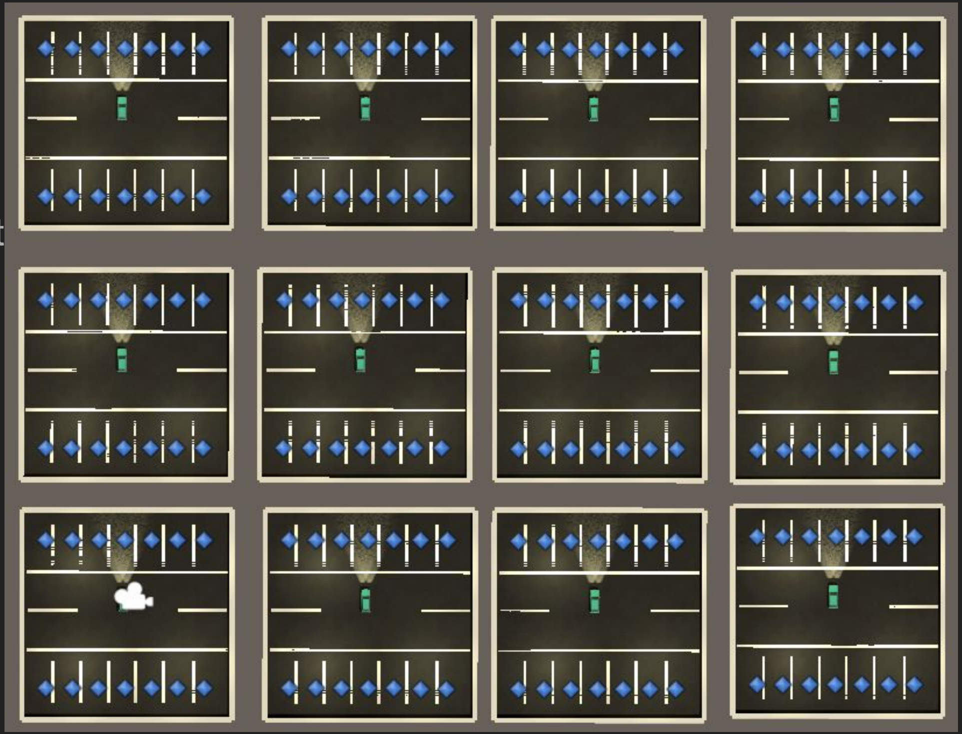
Inspired by Autonomous cars, Tesla’s Autopilot

- Car that can find an empty parking spot

- Simple path finding but we added a lot of randomness to it

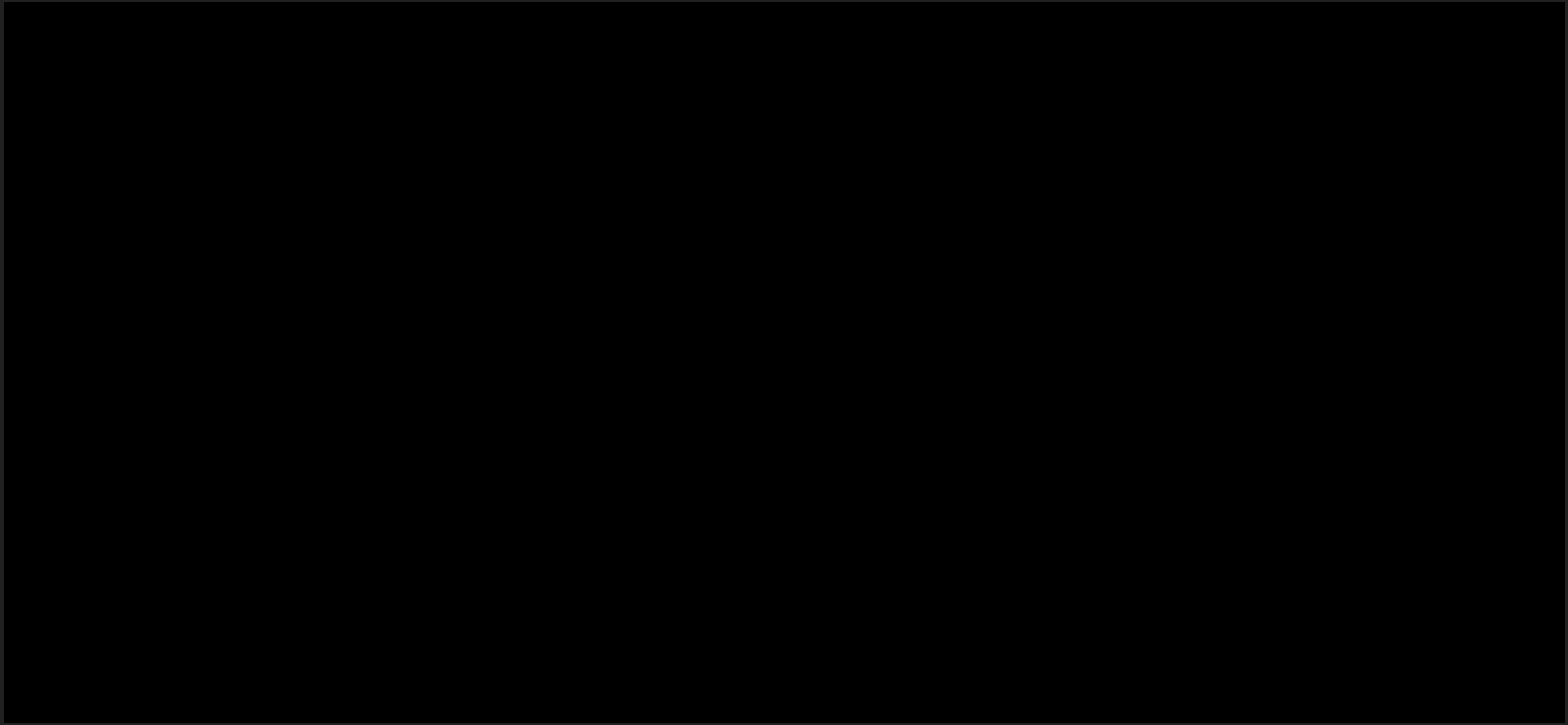
Configuring the Agent

- Adding Ray Perception sensors
- Rewards
  - If crashes on to another car, punishment of (-0.1f)
  - If finds and drives to the empty parking spot reward of (+5f), and episode ends
  - Punishment for taking too long to find the parking spot (-1/5000f)
- Goal:
  - Find the empty parking spot
  - Drive there in least possible number of steps



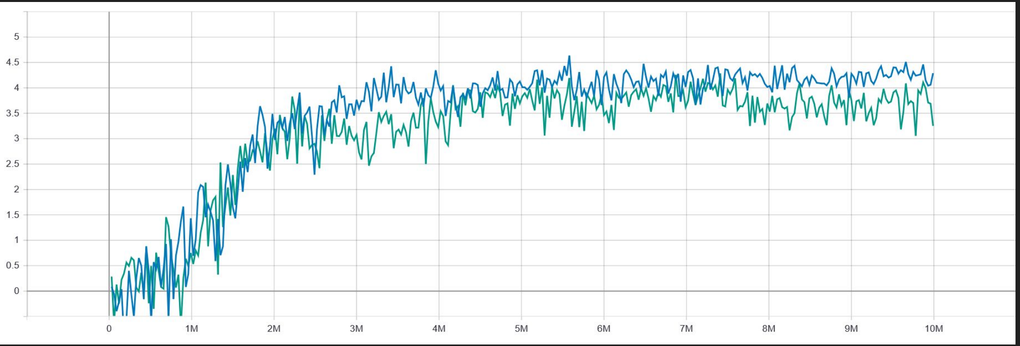
Training(Beginning)

After a few hours of training



Fully trained AI (after 10 M steps)

- Cumulative Reward
  - 3.25 (without GAIL)
  - 4.29 (With GAIL)
- Episode Length(must decrease)
  - At 30k steps
    - 840.2 (Without GAIL)
    - 896.2 (With GAIL)
  - At 9.99M steps
    - 370 (Without GAIL)
    - 179.1 (With GAIL)



Graph Showing the Cumulative Reward

Final trained AI

Conclusion

- PPO and SAC both perform well
- PPO paired with GAIL gives even better performance

On a general note

- Complex world problem can be easily simulated
- General purpose RL algorithms are versatile
- With the applications in transportation, robotics, we can create AIs that can be trained to perform a variety of complicated tasks without having to hard code its actions

THANK
YOU :)