Evaluation

Advanced Modelling and Character Animation- COMP 1473

An Investigation into the Application of Machine Learning in Character Animation State Control

Kaisei Fukaya

Id: 000979004-7

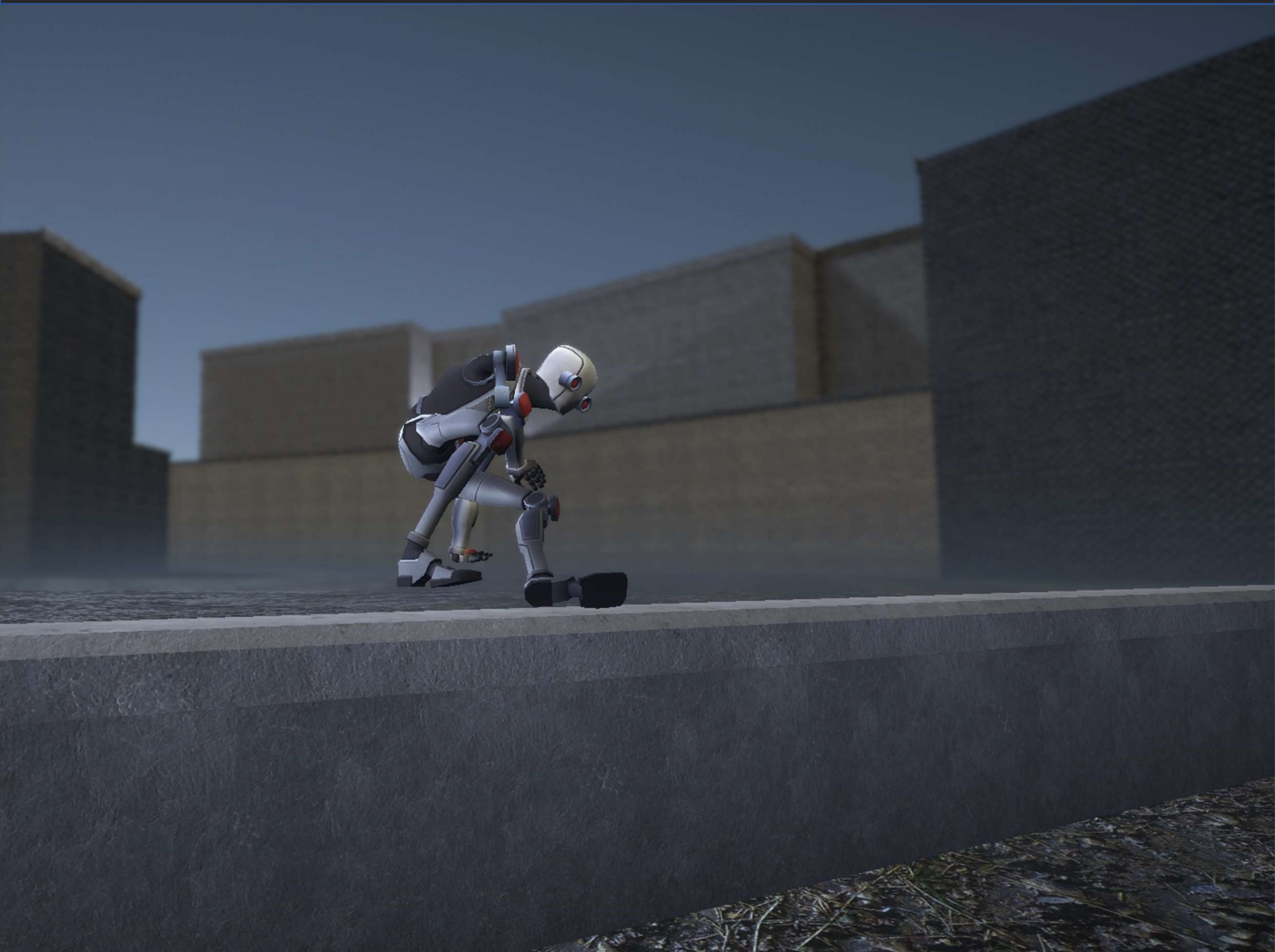

Table of Contents

Introduction

The intention of this project was primarily to investigate how machine learning can be applied to animation control in games. Existing methods involve the use of finite state machines (FSMs), where a state is selected using conditional logic. The primary inspiration for this project comes from the Phase-functioned(Holden, Komura and Saito, 2017) and Mode-adaptive (Zhang et al., 2018) neural networks designed for real-time locomotion within simulated environments. These systems are able to dynamically produce motion to a high degree of precision to where the amount of subtlety could not be feasibly replicated with a set of pre-made animations. This project attempts to apply machine learning at a higher level with a system that is limited to a smaller set of possibilities, with the goal of yielding similar results to an FSM. In this case reinforcement learning would be used to achieve this.

Reflection on Production

The primary prerequisite for training the agent was finding an appropriate set of animations that corresponded with the planned action set: jumping, vaulting, crouching, crawling and falling. Initially it was planned to use the Carnegie Mellon University motion capture dataset (2019) to source these motions. However, finding animations that could readily and seamlessly transition between each other as well as be applied without root motion was not possible. Instead animations were sourced from the website Mixamo (Adobe, 2020), where animations are more consistent and have the option of using in-place motion.

The training process involved a large amount of testing and adjustment, resulting in 59 iterations before the final result was achieved. This involved adjusting hyper-parameters, rewards, and the way the agent's actions are implemented in the environment, see appendix B and D for details. Additionally, major changes to the machine learning architecture itself were made to improve the training process. These include experimentation with curiosity rewards and Generative Adversarial Imitation Learning (GAIL). Although the agent could be trained without these included, they allowed for faster iteration and overall improvement of the agent.

GAIL works by taking a recording of human input in the form of a demo file. The system can then use this as an example to guide its learning. This saves time and allows the system to not over fit in any given curriculum stage. This is perceived by the main system as an additional reward that reflects how much its own actions match the demo. GAIL runs in parallel with the main system, constantly observing and feeding back with rewards.

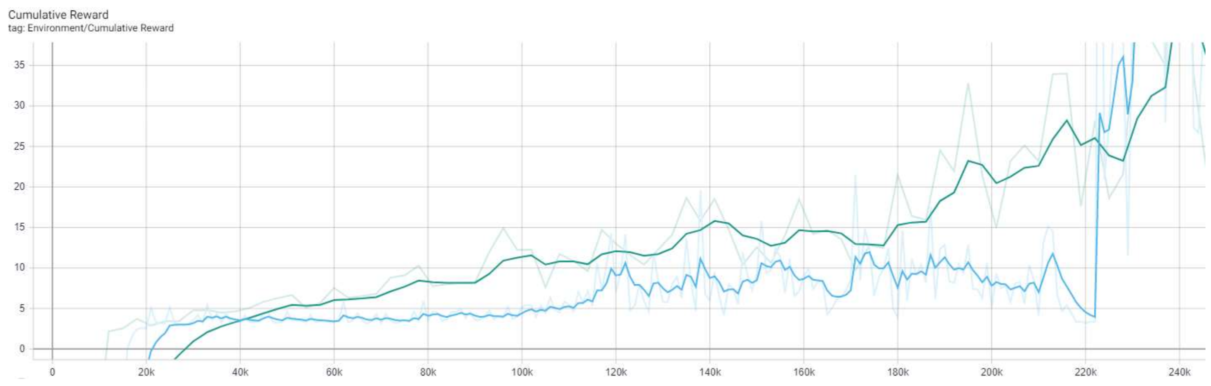

Figure 1: Graph showing cumulative reward over 240 thousand steps of training: run before adding GAIL (in blue) and after adding GAIL (in green).

The agent's progress over time can be illustrated with a graph showing cumulative reward over a number of steps. The reward comes from the reward system where the agent is given positive or negative feedback for actions that it performs. Figure 1 shows a comparison between two training runs, in blue is the agent before adding GAIL, and in green is the agent after adding GAIL. The agent using GAIL yielded more consistent improvement.

This change however, meant that the curriculum became less affective, with the thresholds at the time being detrimental to learning. This is due to the agent spending too long on a single stage and learning to never perform actions that it doesn't need to- so when it eventually reached the next stage it would never perform the action required for the new obstacle. This was fixed by reducing the thresholds significantly.

One issue encountered during training was that by giving the agent full access to all animations from the very start, it would quickly begin exploiting the animations that alter the character movement speed to earn more rewards. Crouching and crawling both reduce the character speed and falling stops the character for the duration of the animation. The agent is rewarded for every 60 steps that it runs without collision, so by using these animations before reaching an obstacle it could postpone collision to earn more of these rewards. A penalty for performing the wrong animation for a given obstacle was added to outweigh the benefits of this behaviour. Additionally, a timed penalty was added for crouching or crawling for too long outside of the appropriate obstacles, this would reset after a number of steps to give leniency.

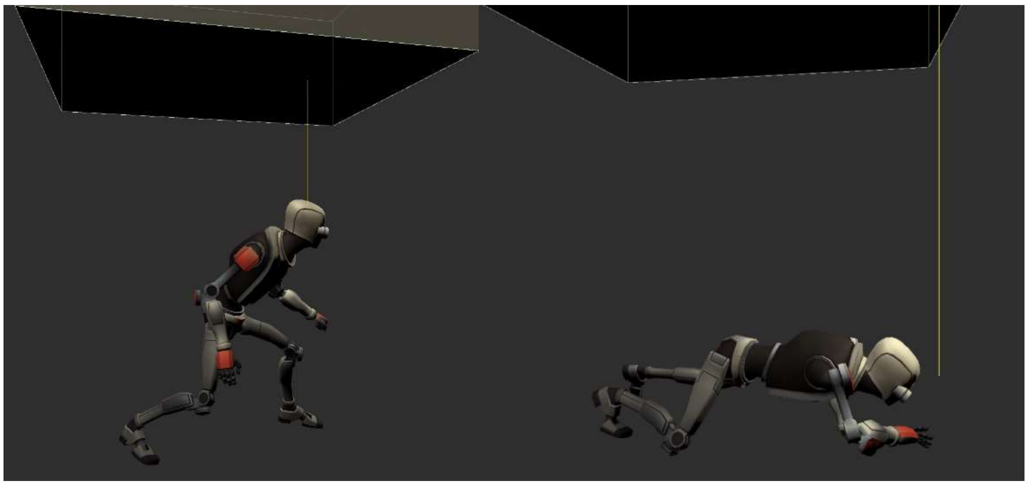
Figure 2: Final adjusted vertical ray-casts for when the character is crouching (left), and crawling (right).

Late in the process of training it was discovered that the vertical ray-casts were providing misleading information to the agent. These ray-casts are used to detect when an agent is beneath an obstacle, their value indicating whether or not the agent should stop crouching or crawling. The issue was that the width of the character is different when crouching from when it's crawling, so by using ray-casts at the same position the agent would believe that it is safe to stand when its legs were still beneath the obstacle. Prior to this discovery, attempted solutions included adjustments to the reward system, however these had no effect as this was an issue with the observation system. Instead, the positions of the ray-casts were made different depending on whether the character was crouching or crawling.

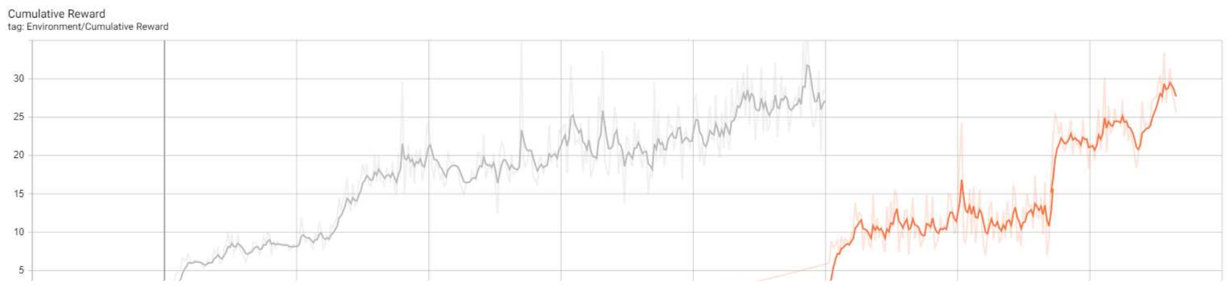

Figure 3: Graph showing cumulative reward over the length of the final run. The initial run (in grey) and the adjustment to real-time speed (in orange).

The final training run took approximately 6 hours to complete, and had to be retrained for some time, as it was discovered that the time scale affects how it learns. This meant that training the agent at ten times speed created an agent that worked well at that speed, but not so well in real-time. So, the final agent had to be trained for an additional hour to allow it to adjust to a real-time environment. Figure 3 shows the final training run, with the main run in grey and the secondary run- adjusting for timescale- in orange.

Conclusion and further work

To conclude, the final agent is able to successfully utilise the 5 animation states to overcome corresponding obstacles. Although the initial aim was to use curriculum learning, an alternative and more efficient solution was found in the use of GAIL. Although a curriculum is still used, it's affects are minimal, see appendix A. Although this system can mimic the ability of an FSM, there is no practical reason for using it in this way- as FSMs can be configured using simple logic- therefore taking less time to implement and debug.

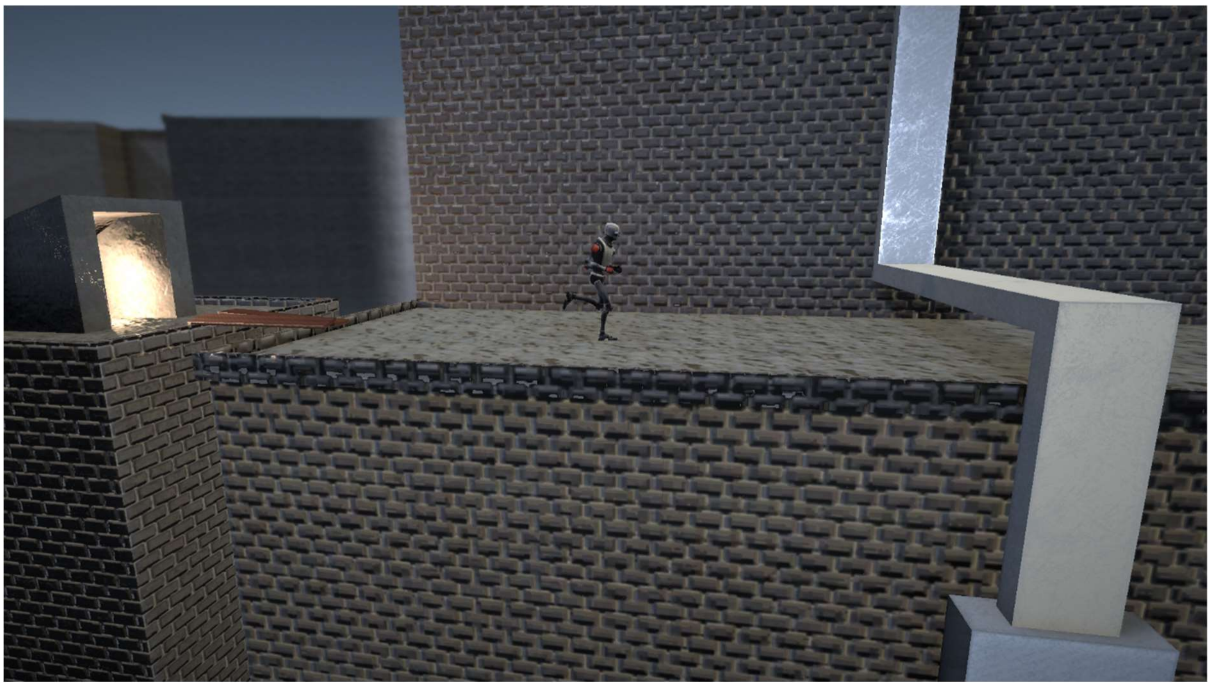

Figure 4:Still frame from the final showcase.

The final animation showcase attempts to present the capabilities of the agent in an environment that makes use of all animation states. It does this with contextualised obstacles such as debris for the character to jump over, a vent to crawl under and a wall to vault. However, a gameplay example should have been added to show how the character behaves with player input, this would have made the showreel more affective at presenting the implications of the project.

Although the scope of this project is limited in terms of the possible actions that the agent can use, the concept could be expanded to a larger set of actions and environmental variations. Potentially, by adding elevation changes such as slopes and a more variable movement speed as well as extending the pool of available animations- the agent could be trained to traverse more complicated environments. Applying this concept to an unsupervised solution, similar to that of the Mode-Adaptive neural network (Zhang et al., 2018), where the agent has access to a large library of motions.

External Assets

In this section all externally sourced assets are listed and referenced.

Asset	Source
Character model	Space Robot Kyle (Unity Technologies, 2012)
Unity ML-Agents plugin	Unity ML-Agents Toolkit (Beta) (Juliani et al., 2018)
Ground Material	Rocky Dirt (Free PBR Materials, 2020)
Brick Material	Brick With Stucco Wall (smmoore1189, 2019)
All sound effects: • Running • Jumping • Impact sound • Helicopter • City ambience	Zapsplat (Zapsplat, 2019)
Music: • Main track • Ambient track	Blue Sizzle by Kevin MacLeod (MacLeod, 2020a) Garden Music by Kevin MacLeod (MacLeod, 2020b)

Appendices

Appendix A

JavaScript Object Notation (JSON) file indicating the thresholds at which to change curriculum stage. This is configured to advance with progress, where the threshold corresponds with a fraction of the maximum number of steps- in this case 0.001 is step 1000 when the maximum number of steps is 1 million.

```
{
    "measure": "progress",
    "thresholds": [0.001, 0.002, 0.003, 0.004],
    "min_lesson_length": 50,
    "signal_smoothing": true,
    "parameters":
    {
        "curricStage": [1, 2, 3, 4, 5]
    }
}
```

Appendix B

Changelog starting at version 18, listing changes to the hyperparameters, reward system and agent observations.

Version 18	-added intrinsic reward signal -set learning rate schedule to constant -increase curriculum thresholds
Version 19	-changed approaching obstacle threshold from 1.5 to 2 -reduced pointless animation penalty from -1 to -0.2
Version 20	-increased pointless animation penalty from -0.2 to -0.6 -reduced height of vault obstacle -reduced height of waist observation
Version 21	-added reward for starting animation closer to obstacle -prevent other inputs if animation is running
Version 22	-increased pointless animation penalty from -0.6 to -2
Version 23	-made pointless animation penalty Done()
Version 24	-reset pointless animation to -0.6 -undo preventing other inputs during animation --increase distance reward multiplier from 0.2 to 1
Version 25	-re-enabled all animations at all times
Version 26	-decreased distance reward multiplier from 1 to 0 -curriculum thresholds increased to 60 -disabled access to wrong animations
Version 27	-removed good anim reward
Version 28	-pointless anim penalty increased from -0.6 to -1
Version 29	-adding imitation learning via GAIL
Version 30	-adjusted head raycast --changed max step from 10000 to 3000 --changed summary frequency to 3000
Version 31	-lowered curriculum thresholds -reduced batch size: 1200 to 512 -reduced buffer size: 12000 to 5120
Version 32	-adjusting Demos
Version 33	-minor adjustments

Version 34	-added pretraining parameter -new longer demo file "Recording3S5" -increase gail strength from 0.01 to 0.05 --increase gail strength from 0.05 to 0.1
Version 35	-reduced curriculum threshold -reduced min lesson length from 100 to 50
Version 36	-changed curriculum from reward to progress
Version 37	-brought back all animations
Version 38	-adjusted curriculum thresholds
Version 39	-good anim reward for standing after crouching or crawling -reduced some curriculum thresholds
Version 40	-changed curriculum thresholds back -increase good anim reward from 0.2 to 1 --tmp change curriculum 0.01, 0.03, 0.05, 0.06 >>> 0.01, 0.013, 0.015, 0.02 >>> 0.001, 0.002, 0.003, 0.004
Version 41	-added randomisation to crouch and crawl obstacles -press to hold crouch and crawl
Version 42	-fixed bug where being ducked under counted as a pointless animation --good anim reward for remaining ducked under
Version 43	-removed residual actions
Version 44	-learning rate shedule from constant to linear -increase GAIL strength from 0.1 to 0.2 -increase pretraining steps from 10,000 to 15,000
Version 45	-decreased time horizon from 1200 to 512 --learning rate shedule from linear to constant --gail encoding size from 128 to 256 --increase pretraining
Version 46	-fixed crouch not working
Version 47	-learning rate schedule from constant to linear -increased time horizon from 512 to 1200 -gail encoding size from 256 to 128 -removed ducked under reward -adjusted prone sensors
Version 48	-added layer mask for sensors

Version 49	-adjusted prone sensor
Version 50	-new demo
Version 51	-punishment for stopping crouching or crawling while ducked under fixed quirk with crouching/crawling where it can be spammed
Version 52	-new demo
Version 53	-removed v51 punishment
Version 54	-fixed reward for passing obstacle to work with crouch and crawl -removed constant reward for ducked under -crouch/crawl no longer punished as a pointless animation
Version 55	-new demo
Version 56	-added check so that crouch or crawl doesn't get stuck -uncrouch/unprone can now only happen when action == 0 --added pointless animation punishment for crouch/crawl again
Version 57	-removed curiosity reward -set gail reward to 0.1 from 0.2
Version 58	-crouch/crawl no longer punished as a pointless animation -made punishment timer only reset if passing an obstacle
Version 59	-train at 1x speed -increase max steps from 1m to 1.2m

Appendix C

Final training hyperparameters.

```
MLSTATES:
  normalize: false
  batch_size: 512
  buffer_size: 5120
  summary_freq: 3000
  time_horizon: 1200
  learning_rate: 3.0e-4
  max_steps: 2.2e6
  beta: 0.001
  learning_rate_schedule: linear
  pretraining:
    demo_path: Demo/Recording12S5.demo
    strength: 0.5
    steps: 10000
  reward_signals:
    extrinsic:
      strength: 1.0
      gamma: 0.995
    curiosity:
      strength: 0
      gamma: 0.99
      encoding_size: 256
    gail:
      strength: 0.1
      gamma: 0.99
      encoding_size: 128
      demo_path: Demo/Recording12S5.demo
```

Appendix D

Final agent reward system.

```
private void WrongAnimPenalty()
{
    //Done();
    AddReward(-1f);
    //Debug.Log("Wronganim");
}

private void PointlessAnimPenalty()
{
    //Done();
    AddReward(-1f);
    //Debug.Log("Pointlessanim");
}

private void GoodAnimReward()
{
    AddReward(0.2f);
}

//Penalty for crouching or crawling for too long

if (!duckedUnder && animControl.isCrawling || !duckedUnder && animControl.isCrouching)
{
    if (punishCounter >= punishMaxCount)
    {
        Done();
    }
    else
    {
        punishCounter += 1;
    }
}
else if(duckedUnder)
{
    punishCounter = 0;
}

//Reward for surviving a set amount of time

if (!IsDone())
{
    counter += 1;
    if (counter >= maxCount)
    {
        AddReward(1f);
        counter = 0;
        //Debug.Log("rewarded");
    }
```

References

Adobe, 2020. *Mixamo*. [online] Available at: <https://www.mixamo.com/#/> [Accessed 27 Mar. 2020].

Carnegie Mellon University, 2019. *Carnegie Mellon University - CMU Graphics Lab - motion capture library*. [online] Available at: <http://mocap.cs.cmu.edu/> [Accessed 5 Nov. 2019].

Free PBR Materials, 2020. *Rocky Dirt PBR Material*. [online] Available at: <https://freepbr.com/materials/rocky-dirt/> [Accessed 27 Mar. 2020].

Holden, D., Komura, T. and Saito, J., 2017. Phase-Functioned Neural Networks for Character Control. *ACM Transactions on Graphics*, 36(2), pp.491–533.

Juliani, A., Berges, V.-P., Vckay, E., Gao, Y., Henry, H., Mattar, M. and Lange, D., 2018. Unity: A General Platform for Intelligent Agents. [online] Available at: <http://arxiv.org/abs/1809.02627> [Accessed 18 Oct. 2019].

MacLeod, K., 2020a. *Blue Sizzle*. [online] Available at: <https://incompetech.filmmusic.io/song/3452-blue-sizzle> [Accessed 27 Mar. 2020].

MacLeod, K., 2020b. *Garden Music*. [online] Available at: <https://incompetech.filmmusic.io/song/3796-garden-music> [Accessed 27 Mar. 2020].

smmoore1189, 2019. *Brick With Stucco Wall*. [online] Available at: <https://share.substance3d.com/libraries/767> [Accessed 27 Mar. 2020].

Unity Technologies, 2012. *Space Robot Kyle*. [online] Available at: <https://assetstore.unity.com/packages/3d/characters/robots/space-robot-kyle-4696> [Accessed 27 Mar. 2020].

Zapsplat, 2019. *Zapsplat*. [online] Available at: <https://www.zapsplat.com/> [Accessed 27 Mar. 2020].

Zhang, H., Starke, S., Komura, T. and Saito, J., 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics*, 37(4), pp.1–11.