GAIT A3 - Reinforcement Learning

By Thomas Zoeff (s3721123) and Michael Reece (s3947763)

Episode Termination Logic

The episode is set up such that it terminates on player death. While this is an obvious approach, it also helps encourage the agent to avoid dying, as this cuts the episode, and its capacity to earn rewards, short.

State Representation

To record the state, we have recorded the player's position and number of lives, and have cast rays in all directions relative to the player's rotation. These rays separately detect collision with enemies, spawners, bosses, shield pickups, spreadshot pickups, and health up pickups.
The ray set up allows the agent to know the relative orientations and distances to all objects of interest. By making the rays orientation relative to the agents we help to group states that should be responded to similarly, as the player's response to most of these objects, but especially the enemies, bosses, and spawners, requires the agent to face the object and move relative to it.
We provided the agent its position so that it can know its position on the game field, and allow it to make strategic decisions such as avoiding the corner's, where mobility is lowered, that it would struggle to learn without position data.
In hindsight using this setup for agent 2 was a mistake which we elaborate on in the adjustments sections.

Reward Setup

In our final iterations, we settled on a reward structure that primarily rewards damaging bosses and spawners, with more modest rewards for acquiring power ups and killing regular enemies. The agent is punished for taking damage and for approaching bosses. The agent is also given a small reward for staying near the middle of the game field, as we found that our agents would often flee too or get stuck in corners.
Detailed below is the process we used to reach this setup.

Initially, the agent was only set up to be rewarded for killing spawners and bosses, and punished for taking damage. This led to effectively no progression in performance, we believe due to the sparsity of rewards for an untrained model.
To combat this, we switched to rewarding the agent for damaging spawners, bosses, and killing enemies. This was better, as the agent did learn how to orient itself and attack, but the rewards were much too high, which led to a very aggressive style of play that resulted in killing many enemies followed by a quick death.
After tuning these rewards over many iterations with only minor variance in performance, we observed two common mistakes made.

1) The agent was especially fond of running into the boss. This led us to implement a penalty that reduced linearly as the agent increases its distance to the boss. This led

the agent to greatly increase its ability to kill bosses, but was not overly impactful on its level of play during non-boss phases.

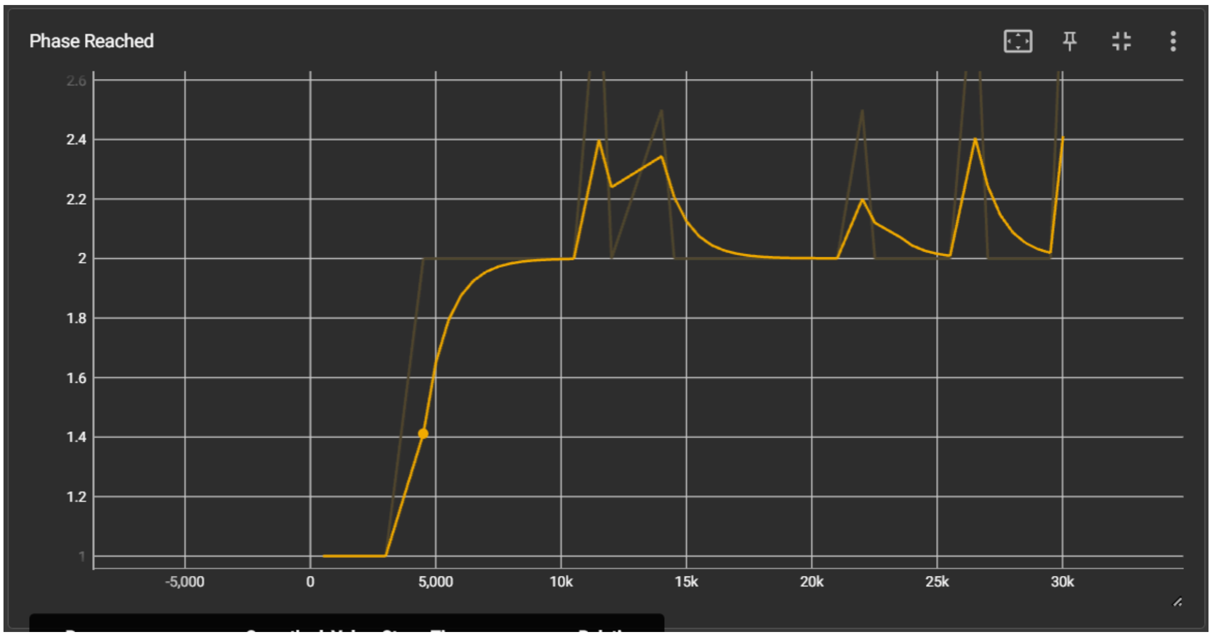
2) The agent would often run into the corner of the stage. To combat this we implemented a reward for the agent the closer it is to the centre of the screen. This approach seemed to do very little for agent performance and possibly even made it worse.

Hyperparameter tuning

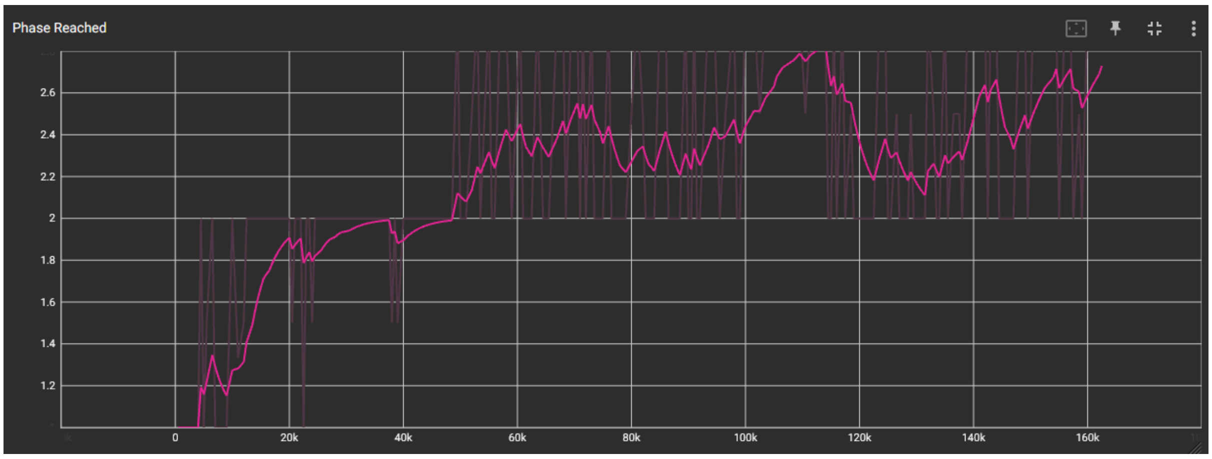
Initially, we set the model's hyperparameters to be the same as provided in the bird example project. From there, we tested disabling normalisation, which had a negative effect on the results as expected. We also tried adding an additional hidden layer, which had a small but negative effect on performance, though we believe that this may have boosted performance given more training time, as additional hidden layers allow for extra complexity, but may take longer to train. We also slightly reduced lambda, to bring it in line with the best practices provided by the unity ml-agents documentation, which had a small but positive effect on performance.

Control Style 2 Adjustments

Adjusting the action space primarily required modifying the action space from control style 1's 5x3 model (5 options for strafing, 3 for rotating) to a 9x2 model (9 options for movement direction, 2 for strafing), and mapping the inputs accordingly.
For the state representation, we didn't make any significant changes, but this was a mistake, as I failed to recognise that the rays should not rotate with the player under this control scheme until it was too late to correct. This resulted in the control style 2 model being quite poor, as it struggled to associate its actions with the result, as movement was inputted through a global rotation, and sensory input was gathered through the player's relative rotation.



Agent 1: Lambda = 0.99



Agent 1: Lambda = 0.95