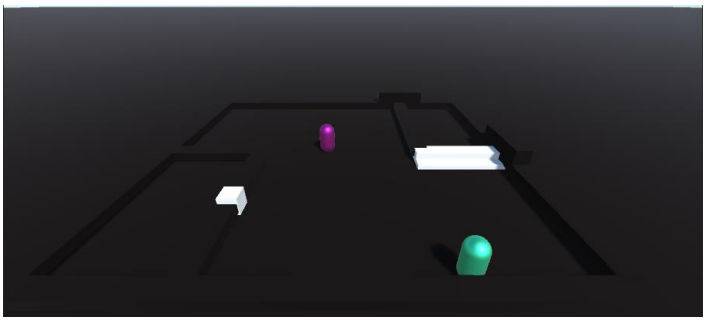
Catch me if you can

- Introduction

In this project, we want to see how two Reinforcement Learning agents learn to play catch and use different exploits on the map. We will use Unity to build an environment and PPO (proximal policy optimization) algorithm for learning target agent and catcher agent policies.



- Proximal policy optimization

PPO is one of the most popular RL algorithms right now and is considered by many as the state-of-the-art algorithm. PPO is motivated by the question: how can we make the biggest possible improvement step without stepping so far that we accidentally cause performance collapse? It does that by requiring the updated policy to be ε-clipped to a small region so as to not allow huge updates which might potentially be irrecoverably harmful. In short, PPO behaves exactly like other policy gradient methods. However, it also ensures that the old policy and new policy are at least at certain proximity (denoted by ε), and very large updates are not allowed.

Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

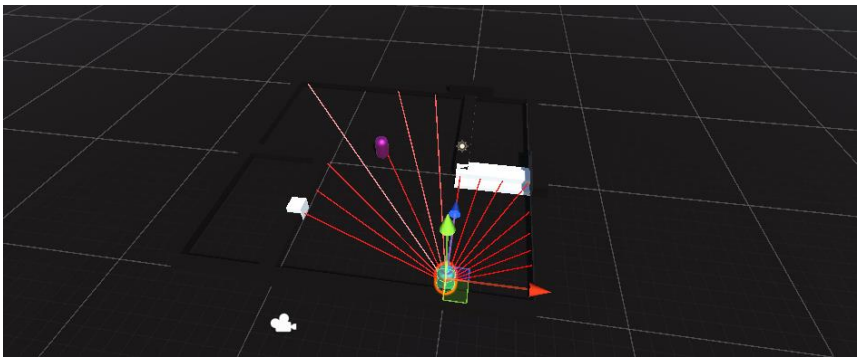
by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for

- Learning configuration

The states were given by a set of rays for each of the agents. Rewards were simple –1 for fail and +1 for win and the hyper-parameters were set to maximize the exploration.



- Learning Process

As expected, the agents started out by moving randomly on the map. After about 200k steps the catcher learned a policy that proved particularly good for him, winning most of the episodes.

After 1.5 million timesteps we saw no improvements in the target agent policy, so we made him 20% faster than the player. Interestingly enough, after the speed boost the target learned how to use the gate to hide from the catcher, this policy was abandoned fast after the player learned how to defend the gate (it requires some time to move the gate because of its mass). After the speed advantage was taken away, the catcher started to win again with ease.