

NPC VOLLEYBALL



JORDAN

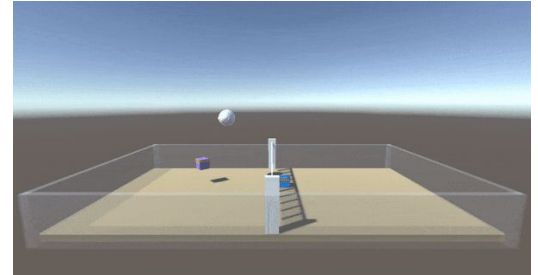
TAYLOR

CS Undergrad Senior

Minors in Mechanical Engineering, Mathematics

Programmer for USU IT Web

RL Methods Project in Unity



Sequential Decision Problem

Given

position and velocity of the ball

Determine


agent orientation and movement

Such That




the ball goes over the net

IN DEPTH

WHY

- Low risk
- Methods-based 
- First exposure to RL
- Within current skill set/computer specs

HOW

- Unity ML-Agents , C++
- Proximal Policy Optimization (PPO) by OpenAI 
- PyTorch
- CoderOne Tutorial Guide 

RL States



Observations

Agent: y-rotation
x,y,z-velocity
x,y,z-normalized vector
to the ball

Ball: x,y,z-velocity

Move forward/back
Rotate clockwise/ccw
Move left/right
Jump
Add force

Actions



Reward

PPO: +1 if ball goes over the net

Self-Play: +1 to the winning team
-1 to the losing team

Set Up

Install proper versions of
Python and Unity, clone
the ML-Agents repo

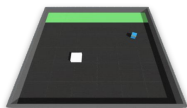
01



Practical Test

Load an example scene,
train an example agent
with the PPO algorithm

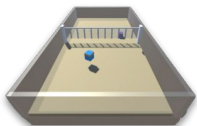
02



Create Environment

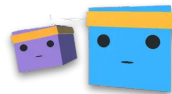
Clone Volleyball repo
Add physics, collisions...
Script env. behavior

03



Steps

04



Add Agents

Decisions, behavior, action
Set up observations
Set hyperparameters

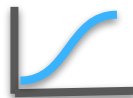
05



Train PPO

16 parallel environments
Simulate 20M steps
Gather results

06



Report Findings

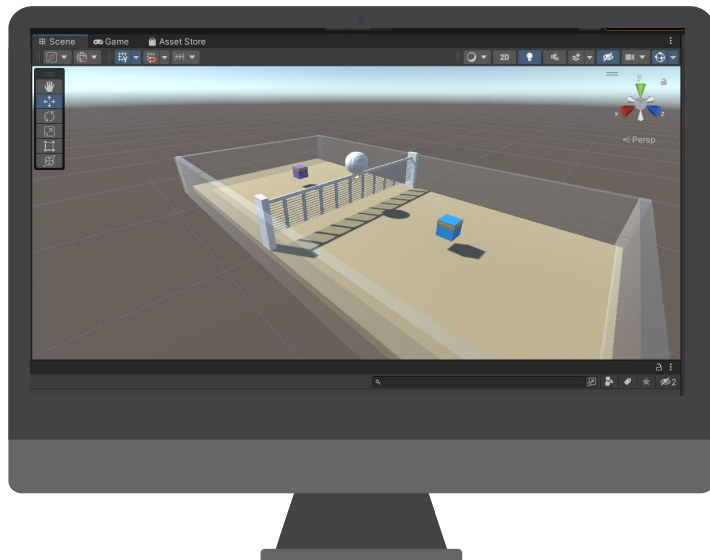
Graph results
Revisit hypothesis
Give scientific observations

Steps 1, 2: Test Environment - PPO



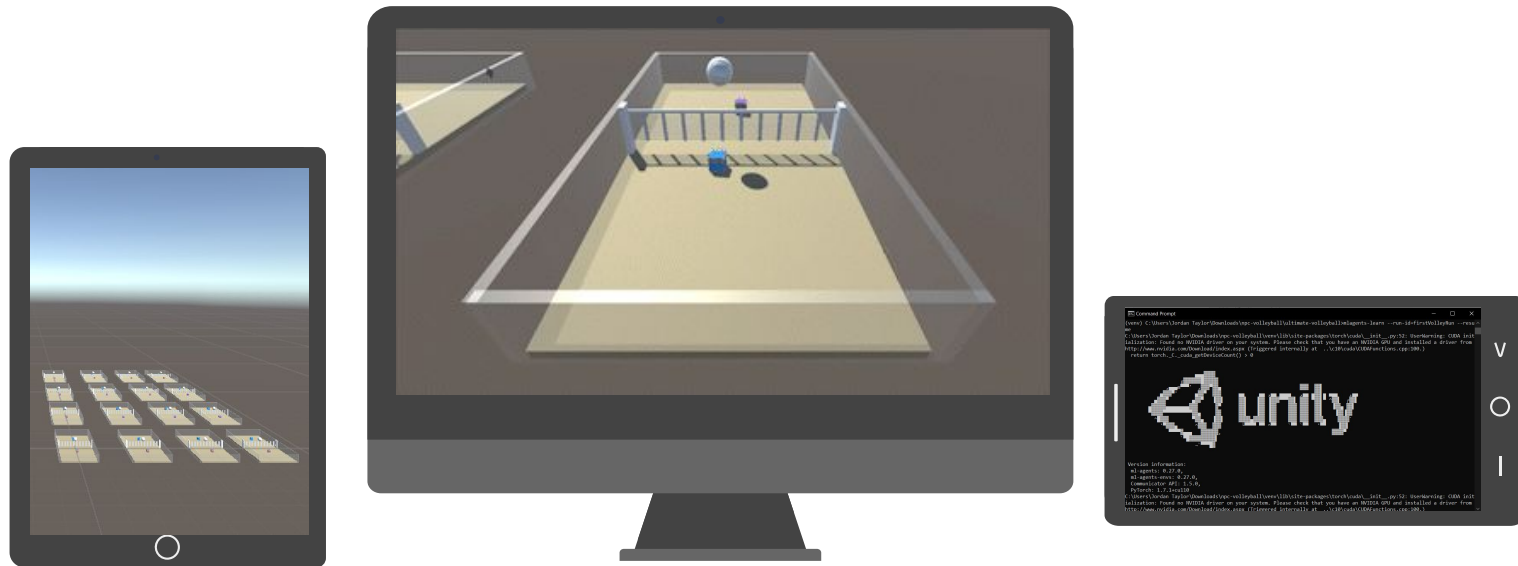
All software successfully integrated for this built-in ML-Agents example, proving that volleyball RL is feasible with this toolstack.

Step 3: Volleyball Environment Setup



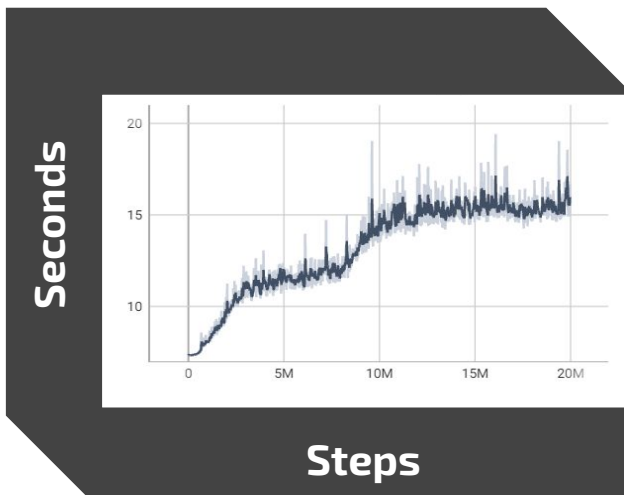
Collisions and physics have been configured. The 3D models were imported.

Steps 4, 5: Add & Train Agents (PPO)

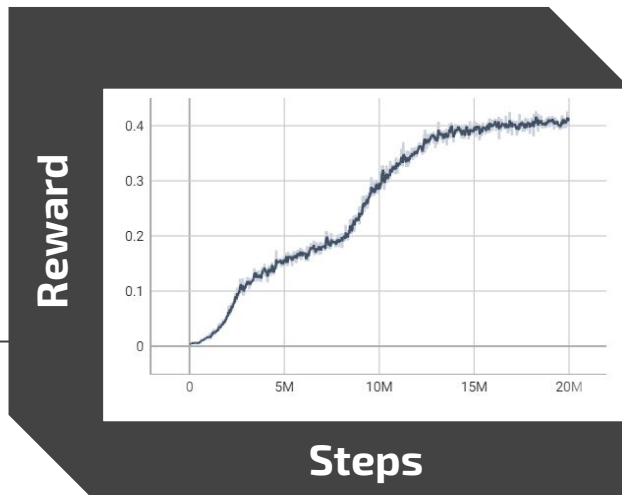


The agent was trained over 20M episodes. The learned behavior (hit ball over net) averages 15 seconds of continuous play. This lacks strategy or competition...

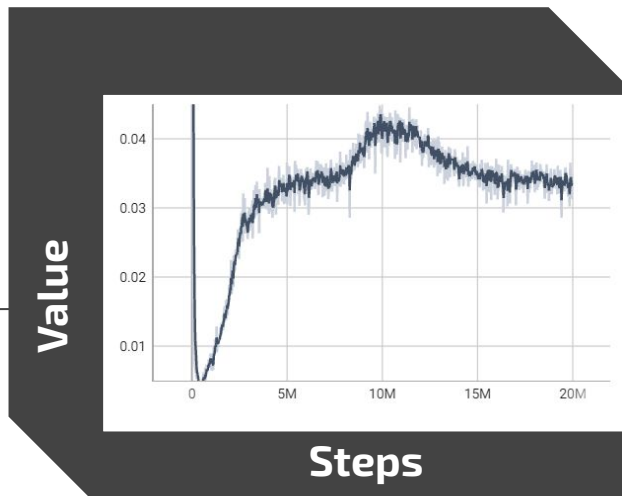
Step 6: PPO Results



**Cumulative
Reward**



**Episode
Length**



Scientific Observations

Algorithm Synopsis

- monotonic improvement in the cumulative reward
- rapid learning from 1-3M steps and 7-13M steps
- highest sustained level of cumulative reward: 40% of theoretically optimal behavior

Why does the method not find the “perfect” player?

- It can only produce an agent that performs as well as its training is set up
- the agent may have gotten stuck in a suboptimal solution

Things I'd Change

- run PPO another 20M steps
- Optimize starting parameters
- Use another RL algorithm for comparison