COLLEGE OF ENGINEERING

Electrical Engineering and Computer Science

ML-BREAKOUT

https://github.com/Minkus-14/ML-Breakout

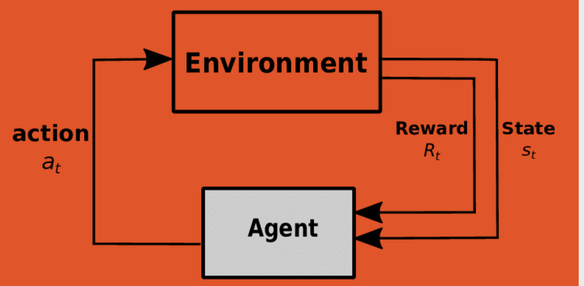
Machine Learning vs Human in a replica of Atari's Breakout

PROJECT OVERVIEW

- This project exposed us to programs and languages that our team had little to no experience with prior.
- The Breakout clone was created in Unity with C# scripts and modeling in Blender.
- Machine Learning was implemented with Unity's ML-Agents package and C# scripts to determine agent reinforcement framework.
- Results were analyzed with TensorFlow's TensorBoard visualization tool.

REINFORCEMENT LEARNING

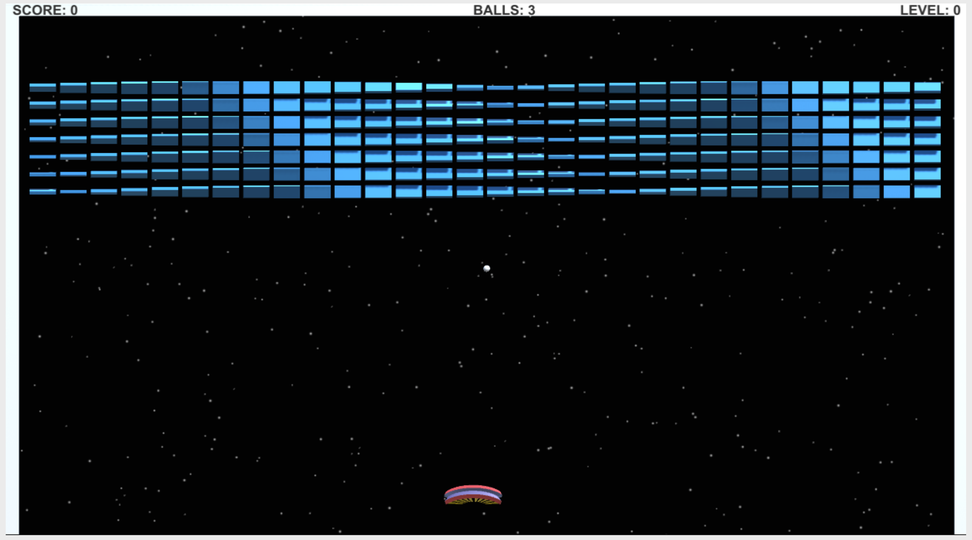
- Training learning agents to take actions in an environment to maximize the notion of cumulative reward.



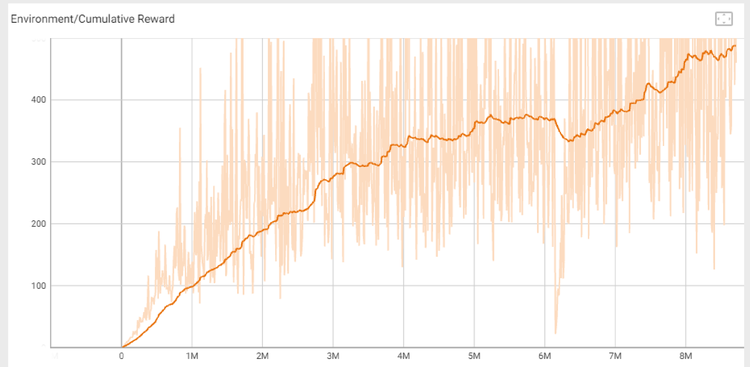

```
void BeginState(State newState)
{
    switch (newState)
    {
        case State.MENU:
            Cursor.visible = true;
            panelMenu.SetActive(true);
            break;

        case State.INIT:
            Cursor.visible = false;
            panelPlay.SetActive(true);
            Score = 0;
            Level = 0;
            Balls = 3;
            Instantiate(playerPrefab);
            SwitchState(State.LOADLEVEL);
            break;
        case State.PLAY:
            break;
        case State.LEVELCOMPLETED:
            Destroy(_currentBall);
            Destroy(_currentLevel);
            Level++;
            panelLevelCompleted.SetActive(true);
            SwitchState(State.LOADLEVEL, 2f);
            break;
        case State.LOADLEVEL:
            if (Level >= levels.Length)
            {
                SwitchState(State.GAMEOVER);
            }
            else
            {
                _currentLevel = Instantiate(levels[Level]);
                SwitchState(State.PLAY);
            }
            break;
        case State.GAMEOVER:
            panelGameOver.SetActive(true);
            break;
    }
}
```

A snippet of the Game Manager C# code handling the single-player game logic whenever a new game state has been entered



TensorBoard visualization of the ML-Agent's cumulative reward after over 8 million training runs



BREAKOUT CLONE

- Built from the ground up, the Breakout clone is modeled in a 3D environment to take advantage of Unity's powerful game engine. This allows the bricks to rotate, adding some visual flair and additional challenge, as the ball can bounce oddly off of the moving surface.
- The game is controlled with C# scripts associated with the Ball, Paddle, Bricks as well as a Game Manager which handles the logical flow between game states.
- A two-player game mode allows the player to challenge friends with a different control scheme or go up against our well-trained Machine Learning agent.

MACHINE LEARNING

- Utilizing Unity's ML-Agent package, an agent has been trained in order to play Breakout through reinforcement learning algorithms.
- A C# script controls all aspects of the reinforcement learning process specifically laying out the reward and penalty functions for the agent to measure progress.
- The agent has the ability to track of the position of the ball, the left and right walls, and the number of remaining bricks, helping to drive the decision making process to maximize its reward.
- Whenever the agent breaks a brick, the algorithm is rewarded based on the number of remaining bricks. By rewarding the agent more when there are fewer bricks remaining, it has a non-linear incentive to break bricks quicker and finish levels in the shortest amount of time possible.
- A small reward is provided whenever the paddle hits the ball, ensuring that the algorithm prioritizes keeping the ball in play.
- Utilizing these parameters, the machine learning agent can optimize the intermediate movements to keep the ball in play, while optimizing strategy to efficiently break bricks and beat the human player

Team Members:

Jared Connor connorja@oregonstate.edu

Tingting Fang fangti@oregonstate.edu

I Chun Hurng hurngi@oregonstate.edu

Jacob Silverberg silverbj@oregonstate.edu

Oregon State University