Magnet Racers AI Implementation

Team BABU

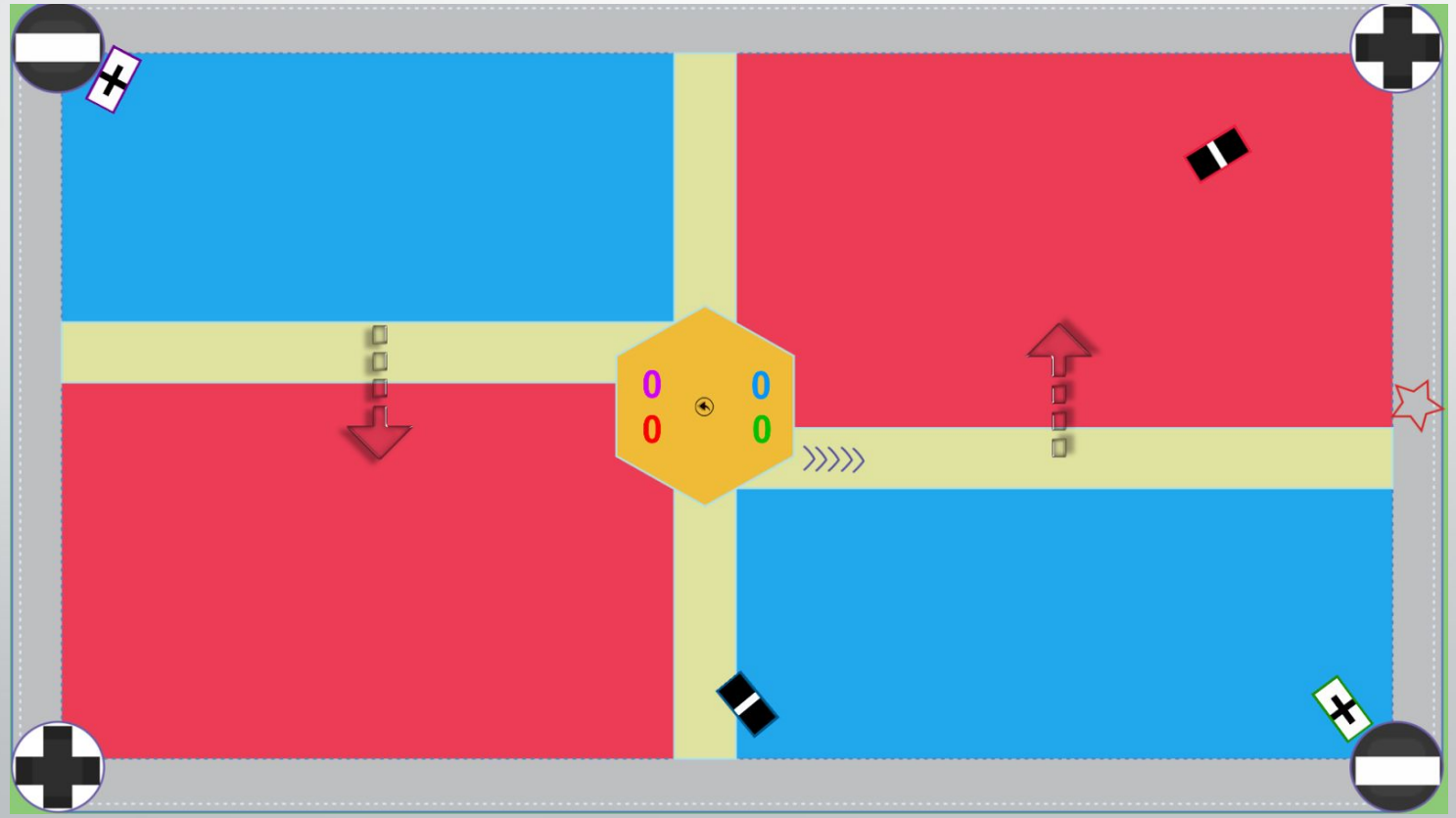
Ben Dunbar, Absinthe Wu, Bill Wisheart, and Uttkarsh Naraya

Objective

Our goal is to implement AI agents to a previously developed game, *Magnet Racers*, and evaluate the effectiveness of our agents

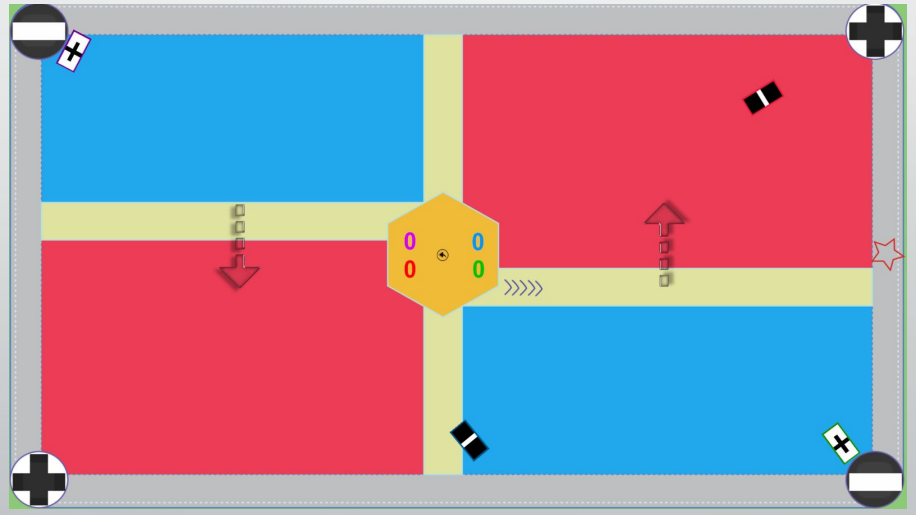
Game Overview

- Magnet Racers is a game in which 2 to 4 players race magnets around a track in a counterclockwise direction



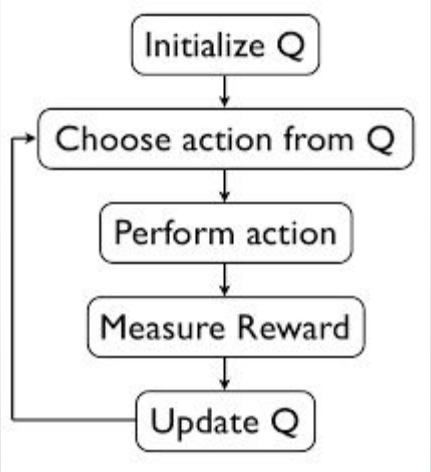
Game Overview

- Racers control their magnets by choosing to switch the polarity of their magnet so that it will interact with the surroundings and make them move.
- First racer to complete 5 laps is declared the winner



Vision Statement

- The main objective will be a successfully tested Q-Learning agent and Decision Tree agent

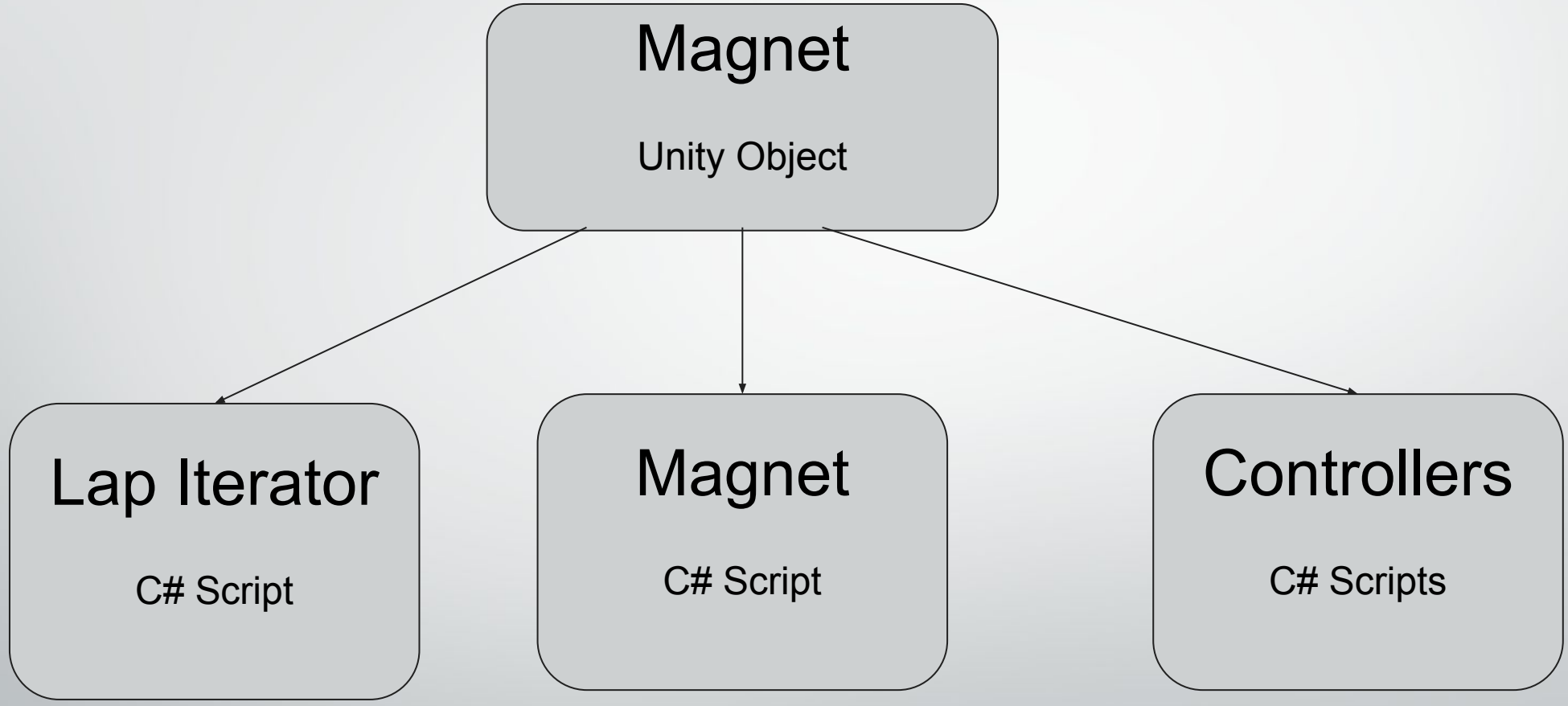


- A sub objective will be a randomly moving agent which will be tested against Q-Learning and DT.

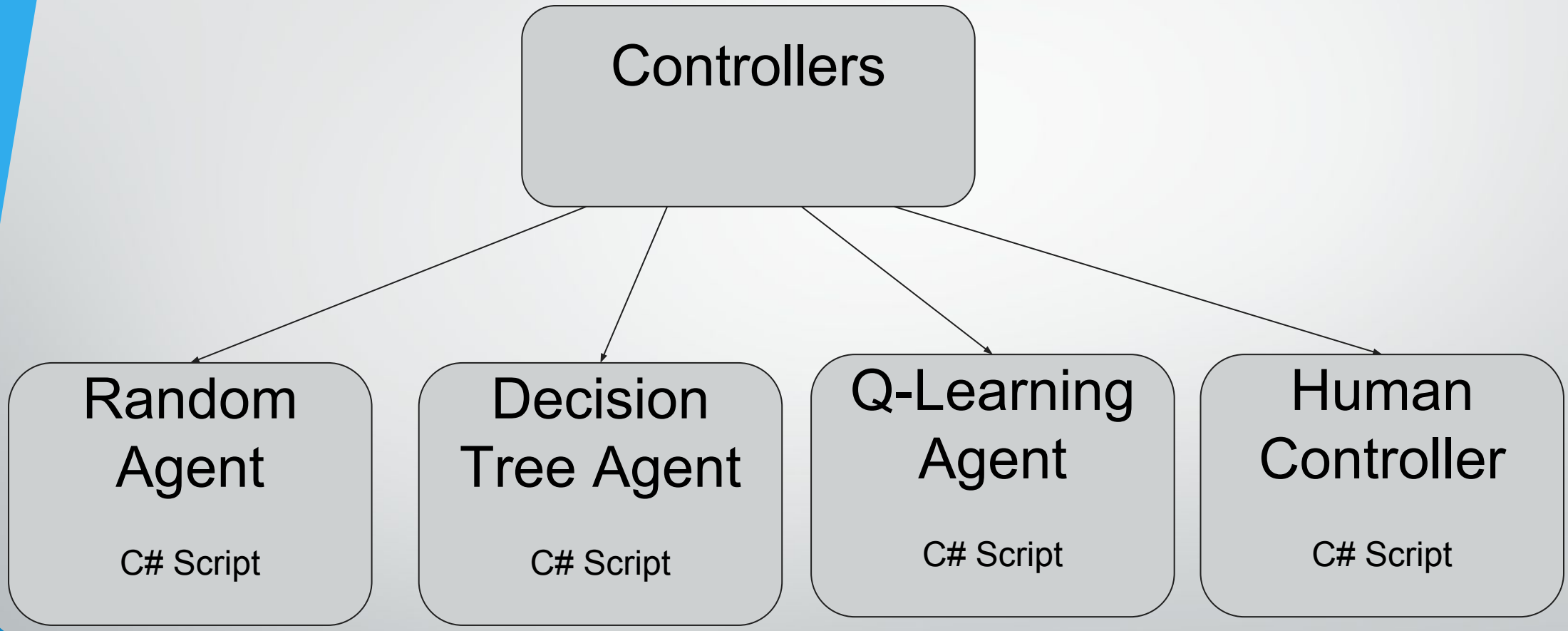
Progress To Date

- Examined code base to understand game
- Cleaned up existing code with object oriented practices
- Implemented Random Agent
- Implemented Decision Tree Agent
- Implemented Q-learning Agent
  - Motion Reward System
  - Force Reward System

Game Architecture



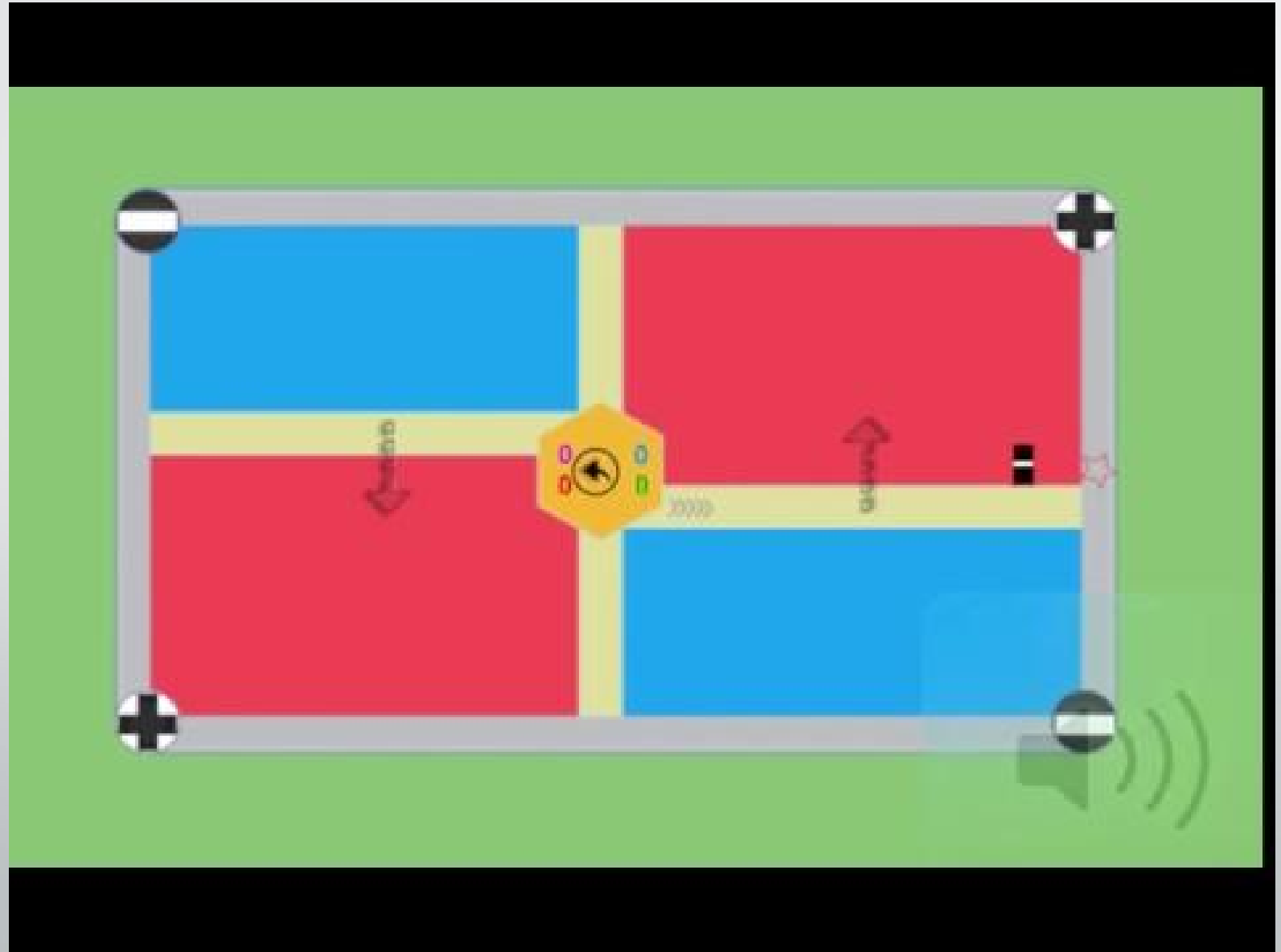
Game Architecture



Random Move Agent

- Random Move:
  - Decides to flip or not using random number generation
  - P(flip) = 2%
  - FixedUpdate() runs every 0.02 seconds
    - 50 chances per second
  - Result: Flip approximately 1 time every second

Random Agent Video



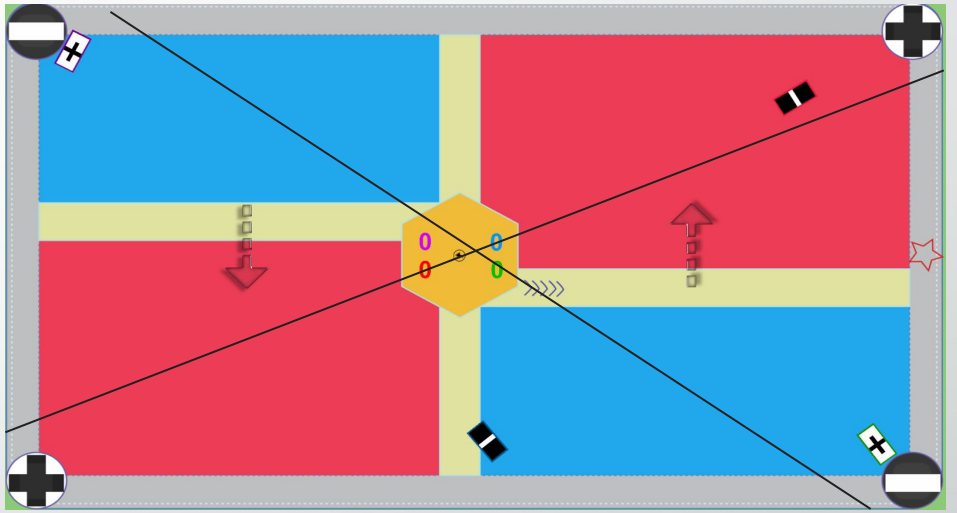
Decision Tree

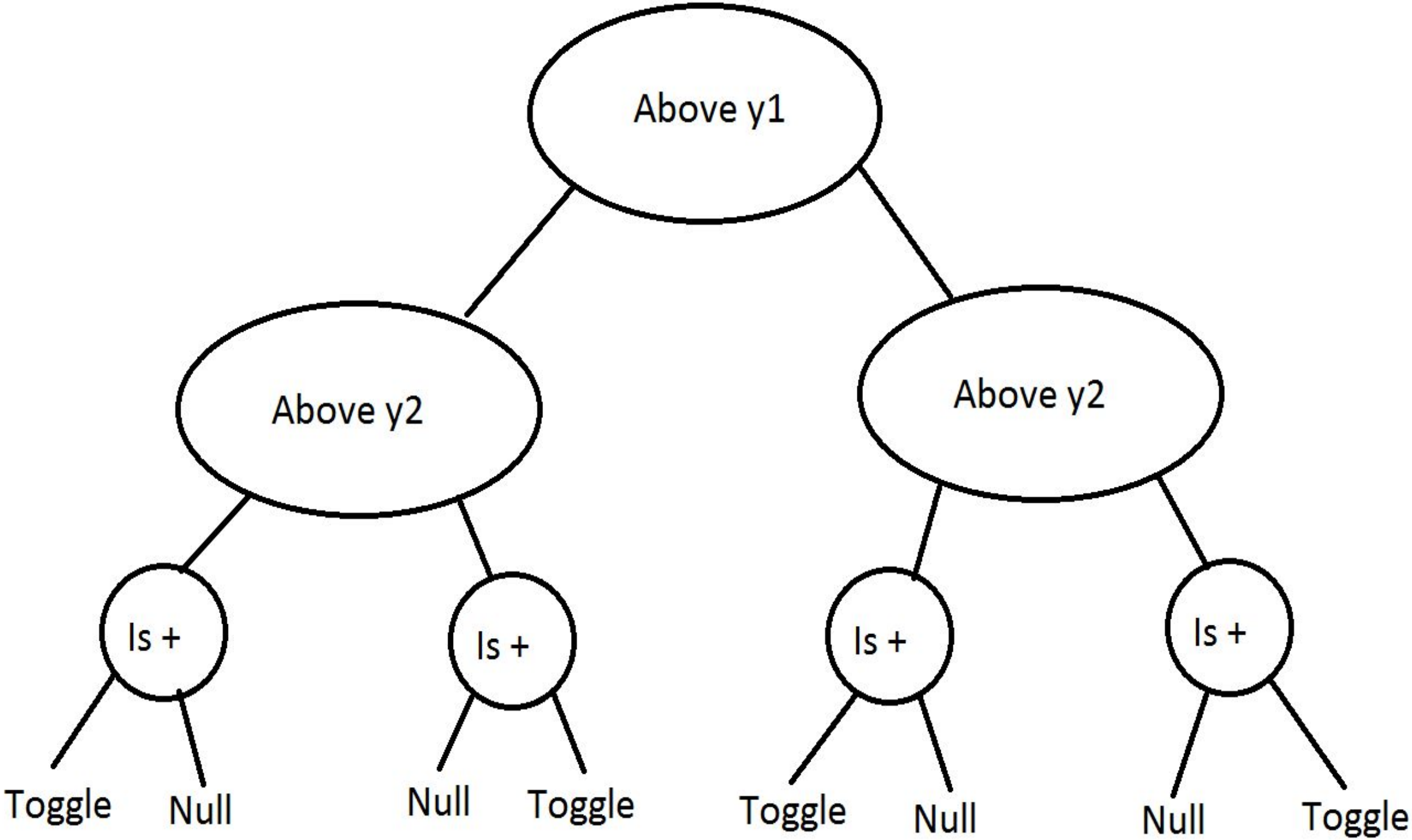
Decision tree uses two lines going corner to corner to compare against magnet position

$y1 = 0.3x + 1.25$

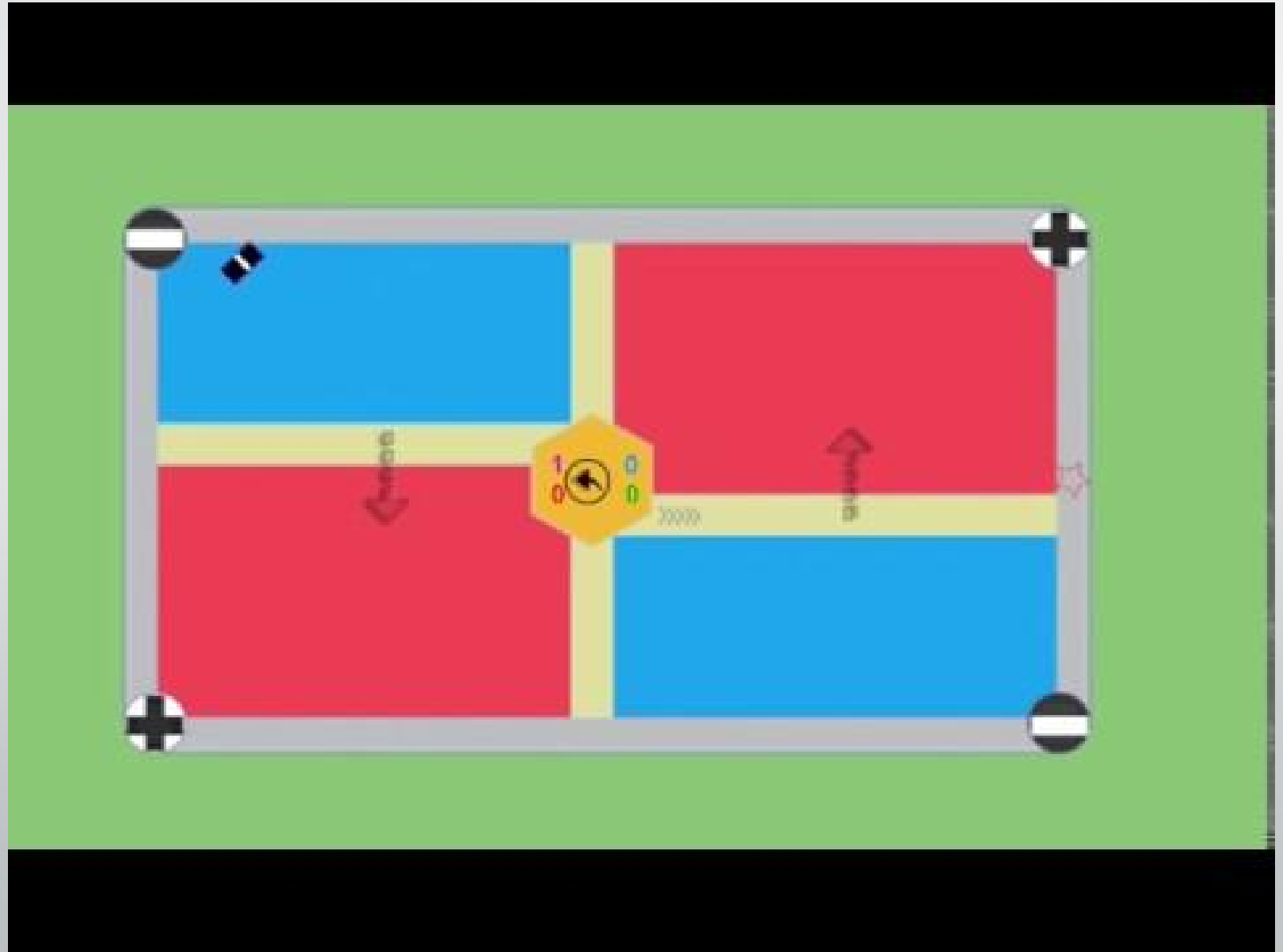
$y2 = -0.6x + 1.25$

Depending on which quadrant the magnet is located in, the charge is toggled




Above y1
Above y2
Above y2
Is +
Is +
Is +
Is +
Toggle
Null
Null
Toggle
Toggle
Null
Null
Toggle

Decision Tree Video



Q-Learning

- State
  - Poles are static - not required
  - Use position and charge
  - Use other magnet proximities and charges
- Action
  - Needs to choose to toggle charge or keep charge
- Reward
  - Reward for counterclockwise motion through the gates on the map
  - Reward for counterclockwise force through the top, left, bottom, right map sections. Help mitigate momentum errors

Q-Learning

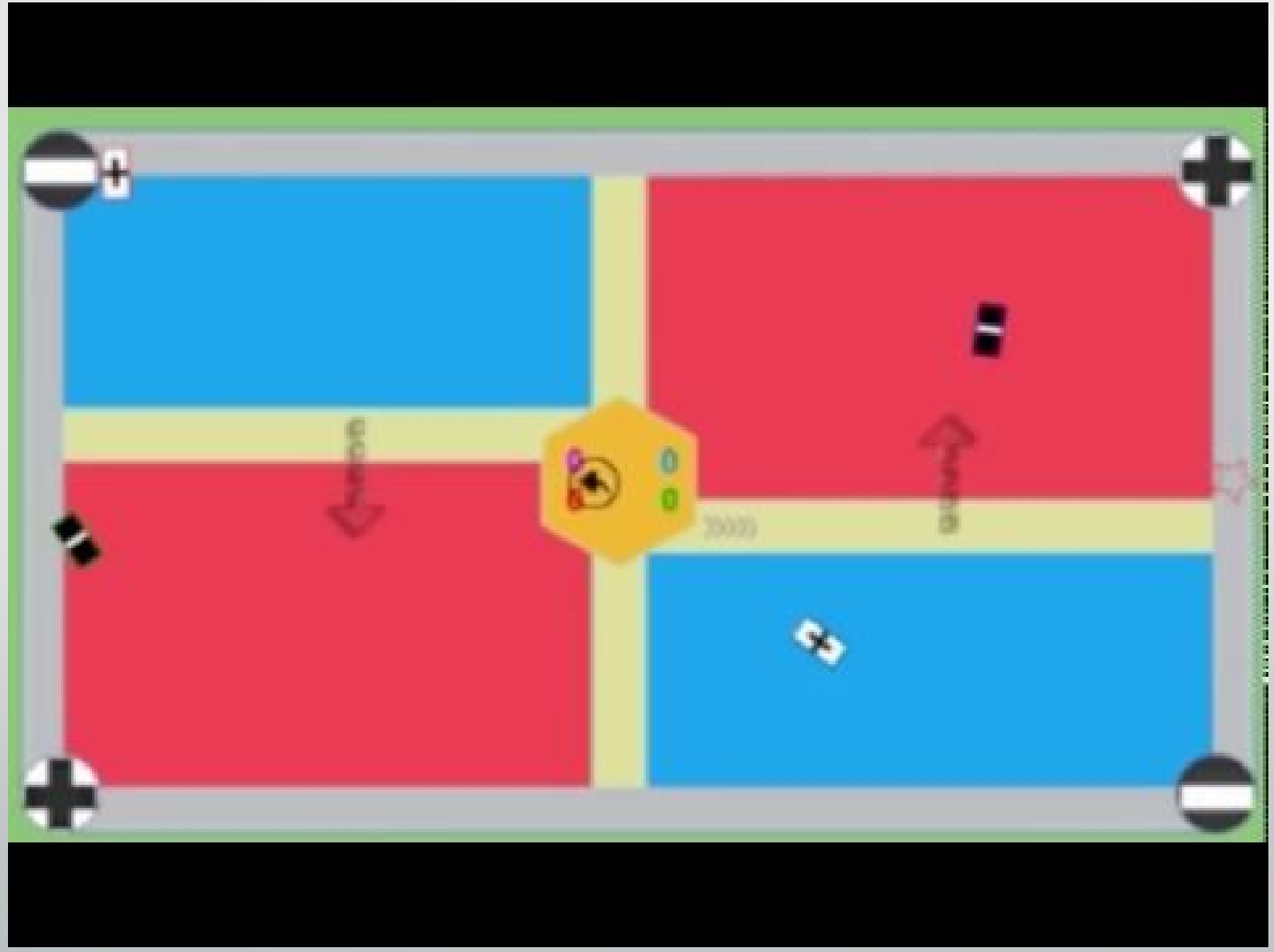
$$Q(s,a) = Q(s,a) + a(r + yMax(Q(s',a)) - Q(s,a))$$

Q(s,a) - state action pair to update Q value for
a - learning rate
r - reward gained
y - discount factor
Q(s',a) - best option from this point on

Tools

- Unity 5.5
  - Original alpha version of Magnet Racer was developed in Unity 5.5 and this will not be changed due to time constraints
  - Unity 5.5 provides a useful platform for the use of graphics and help with game development aspects.
- C#
  - Very compatible with Unity 5.5
  - Existing scripts are in C#
- Github
  - Allow for collaboration on different computers

VIDEO



Next Milestones - Completed!

- Implemented Q-learning to Magnet Racer.
- Debugged.
- Started competitive testing together with all the AI agents.
- Compiled test results.
- Submitted results.

Thank you!

Questions?