Capture The Flag

💻 OVERVIEW

Create an Artificial Intelligence (AI) agent to play a PacMan[1] version of capture the flag (CTF) with python2.7.

📋 Why Game Playing

1. Uses foundational AI techniques such as Minimax, Markov Chain etc.
2. Observe the agent, identify poor decisions and improve.
3. Games are fun!

🔍 BACKGROUND

PacMan[2] is a popular video game and serves as a platform for creating AI agents that control the behaviour of PacMan and PacMan's enemies, the Ghosts.

Capture the Flag[3] (CTF) is a popular game style where two teams compete to capture a flag on the opposing side and bring it back to their side to score a point. At the end of the game, the side with the highest score wins.

As a combination we have PacMan CTF, where each side has 2 bots that are required to collect food pellets and bring it back to their side to score points.

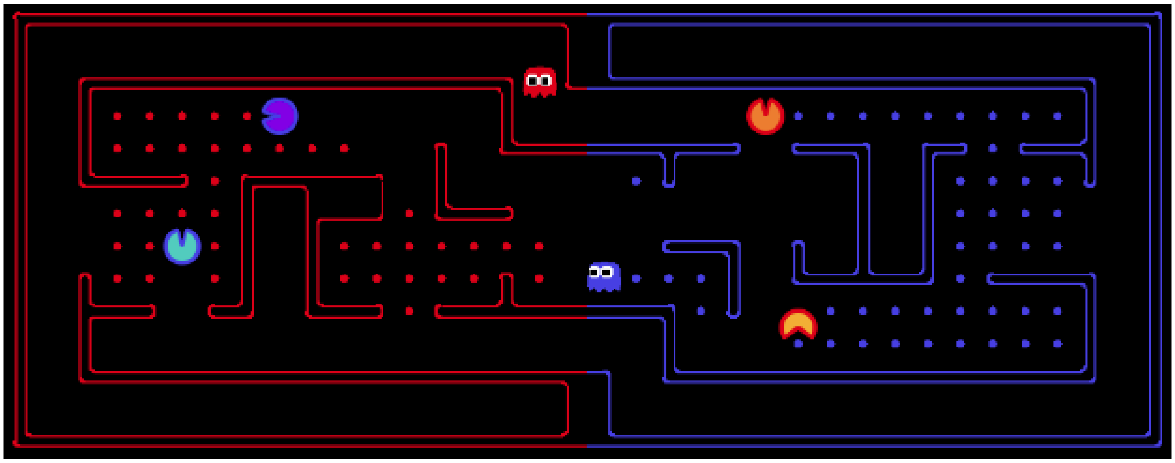
NEW LINK

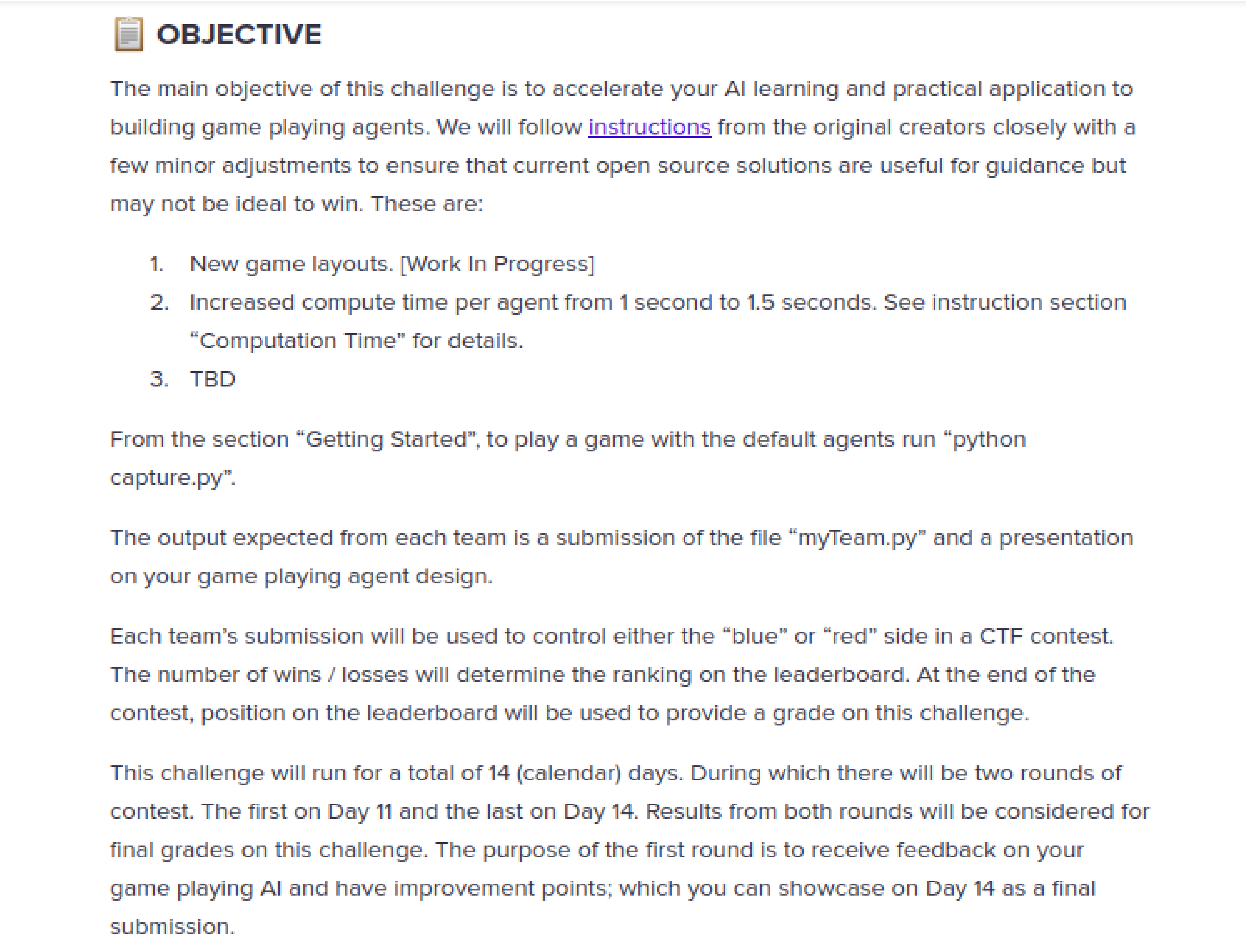
https://inst.eecs.berkeley.edu/~cs188/sp20/projects/

[1] "Pacman Projects - Berkeley AI Materials." http://ai.berkeley.edu/project_overview.html. Accessed 20 May. 2020.

[2] "Pac-Man - Wikipedia." https://en.wikipedia.org/wiki/Pac-Man. Accessed 20 May. 2020.

[3] "Capture the flag - Wikipedia." https://en.wikipedia.org/wiki/Capture_the_flag. Accessed 20 May. 2020.





📋 OBJECTIVE

The main objective of this challenge is to accelerate your AI learning and practical application to building game playing agents. We will follow instructions from the original creators closely with a few minor adjustments to ensure that current open source solutions are useful for guidance but may not be ideal to win. These are:

1. New game layouts. [Work In Progress]
2. Increased compute time per agent from 1 second to 1.5 seconds. See instruction section “Computation Time” for details.
3. TBD

From the section “Getting Started”, to play a game with the default agents run “python capture.py”.

The output expected from each team is a submission of the file “myTeam.py” and a presentation on your game playing agent design.

Each team’s submission will be used to control either the “blue” or “red” side in a CTF contest. The number of wins / losses will determine the ranking on the leaderboard. At the end of the contest, position on the leaderboard will be used to provide a grade on this challenge.

This challenge will run for a total of 14 (calendar) days. During which there will be two rounds of contest. The first on Day 11 and the last on Day 14. Results from both rounds will be considered for final grades on this challenge. The purpose of the first round is to receive feedback on your game playing AI and have improvement points; which you can showcase on Day 14 as a final submission.

Submissions on Day 14 will also include a presentation on your game playing agent. We would like each team to explain design decisions in significant detail.

ADVICE FROM PROFESSORS

There is a lot of material you would need to cover in order to build competent game playing AIs. To begin use the built in game agent and perform a code review to understand how the game operates. Spend upto 1 - 2 days on this. On completion do the following:

- Complete the MultiAgent Search project. This will introduce you to the basic building blocks of game playing, namely game states, game state tree and the minimax algorithm. Spend 2 - 3 days on this.
- Complete the Tracking project. This will introduce you to fuzzy position tracking of the players across the game map. Keep in mind that if your opponent is more than 5 manhattan distance away you will receive a fuzzy position of the opponent(s). Spend 2 - 3 days on this.
- Try to combine both projects into your game playing AI.

Once you have a basic game playing AI:

- When you think you have improved on your previous iteration, run 100 games (or some reasonable number) across all the maps and look at the win / loss statistics for your old AI vs new AI. Pick the winner. Iterate this.
- Try your AI on new PacMan game layouts. See “layouts” folder for design examples. Design your own as well.

This can be a fun project and perhaps Professors will also submit their game playing AIs into the contest.

Do respect the game and each other. Do not plagiarize solutions as your own; take inspiration and raise the bar.

🛠️ Code

Download the code here. This is a slightly modified version (as mentioned in “Objective") of the original code located here.

💎 Connection to ZestMoney

Customer onboarding can be thought of as a two player game between the customer and ZestMoney. The game states can be viewed as the pages / fields that the customer needs to fill to complete the journey. At every stage completion, the customer has an option to continue or drop off. The ZestMoney agent needs to maximize the probability of the customer to exercise the option to continue.

The ZestMoney actions can be page redesign (move around fields on page), or introduce “reward" pages such as “You already have a Credit Limit of 10K, continue to have it increase!". This can be further customized based on the type of Customer. This could be defined as a set of legal actions that the ZestMoney agent can perform.

Game playing has real world applications.

📨 QUESTIONS

In case of any concerns, reach out to me at **professor@zestmoney.in**