CS4180 - Project Milestone Report

Tak Boardgame Environment & AI Player Algorithm

Pablo Kvitca - Fall 2021 - October 29, 2021

Team: Pablo Kvitca
Expected Milestone (from the proposal): "The project milestone for 11/16 will be a working environment for the game and learning on the specifics of the MCTS method and brainstorming on how to apply it to this task."

Report

So far I have successfully implemented an environment (that extends gym.Env) on Python. The environment ended up being more complex than I had initially anticipated. I attached a diagram below of how I implemented the game.

The more complex parts of the code are: **generating all possible actions at a given state** and **checking whether the game has ended** (see appendix for details)

I also went ahead and implemented a **RandomPolicy** to test my game, and I wrote **UNIT tests** for most of my environment code. The environment also has 2 render modes: "text" and "image". The first is imperfect, the game has a 3D layer but the text output only shows 1 piece per stack, so it is a "top-down" view of the board, not very useful. The "image" mode uses *Plotly* to make an interactive 3D plot for a *static* state of the game. This will be useful for debugging and testing. Additionally, I implemented a second **RandomPolicyLessLikelyPlace**. In most cases, there are more possible *place* actions than *move* actions, but *move* actions are usually higher value. The naive random policy usually selects *place* actions, this variation selects a *place* action only 50% of the time, the rest of the time it selects a random *move* action.

I also started **investigating MCTS** and brainstorming how to implement it for my environment.

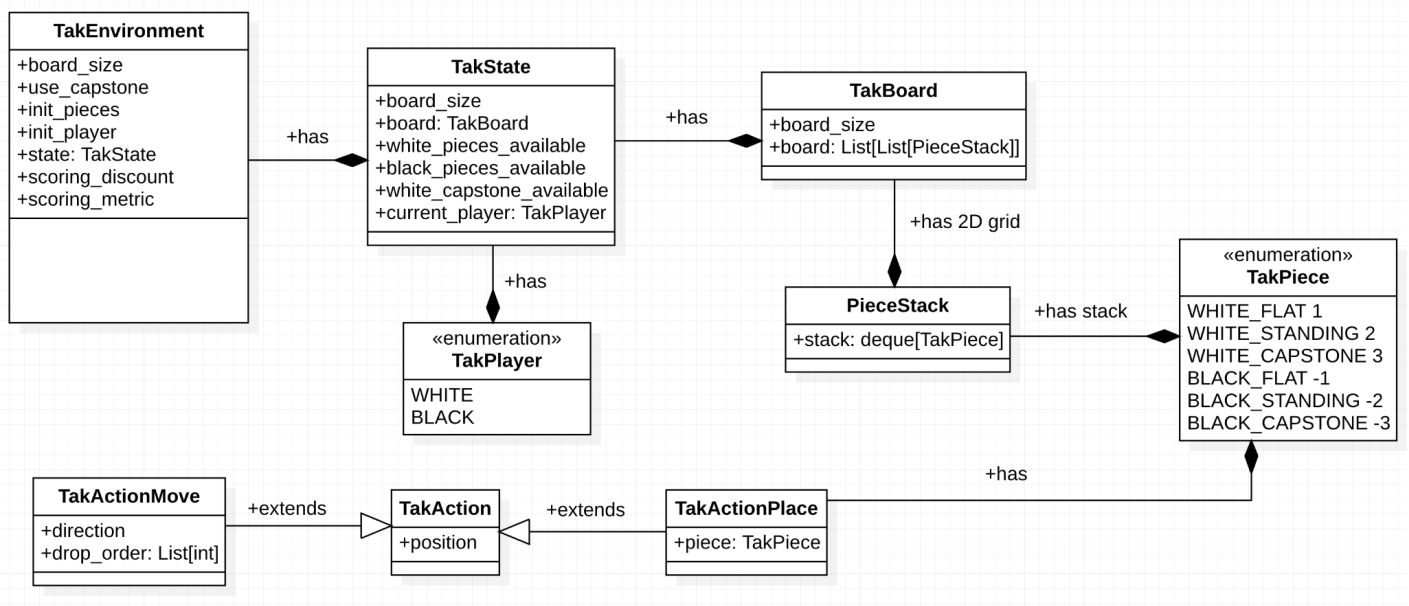
Reflection

I had not planned to implement the 3D rendering of the game (but it was necessary for easily debugging the environment). Also, I had not expected to implement a 2nd improved policy (even tho it is still random) yet. But it seemed quick enough that I decided to do it sooner.
I consider that I achieved the milestone, but I haven't spent as much time as I had planned working on understanding MCTS and starting to implement it, but I should be able to start soon.

Replan

I think I am in a good position with the project. My next step is to actually implement some learning algorithm in this environment. However, I was hoping to implement some algorithm other than MCTS, but I'm not sure that's going to be possible due to time constraints. I will still consider other options. I may be able to use existing implementations of other algorithms to compare to my own, but I am unsure they would work out of the box with my environment.

Diagram



Appendix

Generating all possible actions at a given state

Generates two types of actions: *place* and *move*. In the *place* case, for every type of piece (*flat*, *standing*, and *capstone*) it generates one action per empty position on the board. In the *move* case, for every position controlled by the current player (positions where the top of the stack is a piece of that player), it generates moves in each direction (*up*, *right*, *down*, *left*), for each of the possible partitions of the number of pieces on that position (up to the game carry limit = board size).

Checking whether the game has ended

eeds to check where there is a vertical path or a horizontal path from the borders of the board with road pieces (*flat* or *capstone*, but not *standing*) of the same color (only considering adjacent orthogonal pieces).