CISC 489/689 Intro to Multi-Agent Systems
Programming Assignment 1
Simulation of Cooperative Agents in a Tileworld
Due: **Friday, April 22, 11:59 pm**

The Simulation Environment

- Please download the simulator from here.
- 0. Dependencies

  ```
  Python 3.10
  Pygame 2.1.2
  Numpy 1.22.3
  ```

  I recommend using PyCharm for your own development. Simply create a new conda environment with `Python 3.10` as the interpreter and open the simulator as the project directory, then run the following:

  ```
  pip install pygame
  pip install numpy
  ```

- 1. The Agent

  It's an NxN Tileword (modified from here). In the Tileworld, two agents from the following three types are collecting coins:

  - `demoPlayer()`: the agent you can control with arrows on your keyboard (shown as Mario on the map).
  - `randPlayer()`: the agent that will walk randomly on the map (shown as a colored square on the map).
  - `PlayerA() and PlayerB()`: the agent(s) designed by your team (you can use your own favorite icon or a simple different colored square).

  Each agent starts from the top left corner. Each agent can move one square distance at a time. The `demoPlayer()` and `randPlayer()` can move in four directions (up, down, left, and right in one square at a time). You can make your own designed agent(s) move diagonally (but still in one square distance at a time).

- 2. Utility and the Goal

  In the Tileworld, a certain number of coins with values ranging from 1-9 are randomly generated. They will disappear in a certain period of time and then regenerate another set of coins.
  When the two agents collide, each of the agents will be deducted 100 points (this is expensive, so your agent should find a way to avoid this if possible.
  The utility (for a cooperative game) is defined as the total value of coins collected by the two agents, subtracted by the number of points lost in any collision (i.e. "Total Score" printed when the program finishes).

The goal is to develop a good strategy of cooperation in order to make your utility as high as possible.

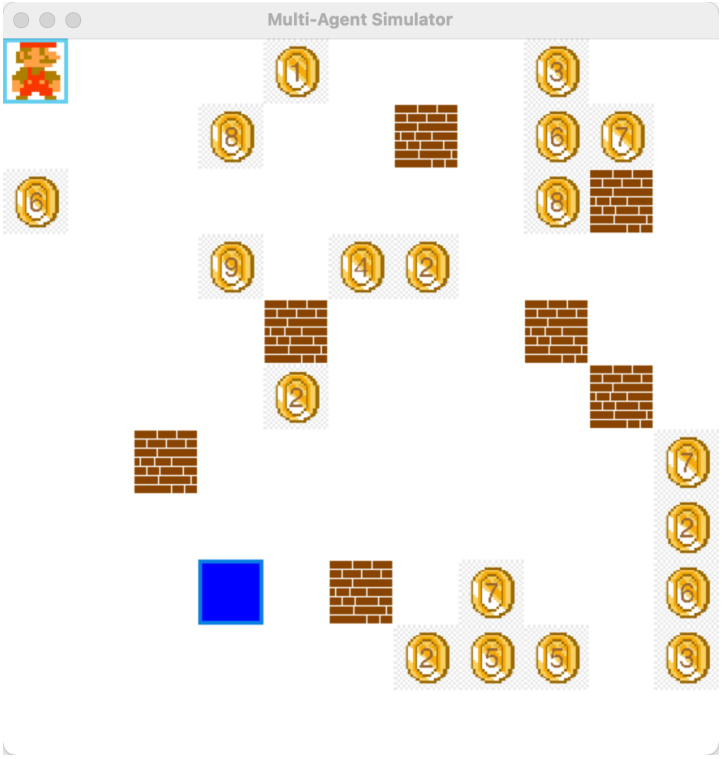

Figure 1. Example picture of the demo

- 3. Other Features
  No agent can get through the wall or get out of the Tileworld.
  Coins will be randomly generated and each coin may have a random lifespan (from 1-5 seconds). This means there's a possibility that your action may fail (i.e. coin not collected as expected since they can disappear).

  Before you start, please take a look at the README file that comes with the code.

Task Description

- There are three parts of this programming assignment, that is to make the following simulation cases work:
  - Case 0 (0 points): one `demoPlayer()` and one `randPlayer()` are in the Tileworld, where you can control the `demoPlayer()` and get familiar with the rules.
  - Case 1 (10 points): two `randPlayer()`'s are in the Tileworld, which can be used as the benchmark (this is a base case: there is nothing for you to write, just make sure this works).

    - Case 2 (40 points): one `randPlayer()` and one self-designed agent are in the Tileworld. Your design should show your agent is rational.
    - Case 3 (50 points): two self designed agents (can be the same or different) are in the Tileworld, where you test if your agents are cooperating well with each other. Your design should show how agents can benefit from cooperating.
- Note: This is a test environment for you to try out your strategies, you can make changes, say the size of the map (i.e. N), the number of coins generated each round, the length of time for each round, etc. Your code will be tested in a different set of variables (like a larger map, more rounds, longer game time).

Submission

- 1. Your code (`coopAgent.py`).
  Basically, only `coopAgent.py` will be tested to ensure fairness.
  If you made any changes to the other files in order for your code to work (e.g. certain Python packages), please point them out. In this case you may also want to upload the whole set of your code just in case.

- 2. A short report (no limitation on the number of pages, but preferably less than 2 pages not including appendices).
  Please describe how your design works (e.g. features, algorithms, etc), the comparison among the three cases, and the advantages of your design (e.g. how agents communicate and cooperate, how your design gets higher utility, etc).

Team Formation

Each team can have up to 2 students. You can find a teammate by posting on Discord.
Each team only needs to make one submission.

Bonus Point

- Try to polish your design. We will run your designed agent pairs in the same environment and rank all of them. The top three pairs will receive additional rewards (+20% for 1st place, +10% for 2nd place, and +5% for 3rd place.).

Integrity

Your code must be your own work. If you use ideas, code pieces from others, please cite them properly in the report.

Questions?

Please post them on Discord, and consider replying to your classmate's questions. You can also attend TA's office hours or contact the TA.

Resources

- Pygame Docs: https://www.pygame.org/docs/
- Pygame on github: https://github.com/pygame/pygame
- A brief Pygame tutorial: https://www.youtube.com/watch?v=FfWpgLFMI7w
- Python / Pygame tutorial: Collisions between static and moving objects: https://www.youtube.com/watch?v=1_H7InPMjaY