CS3600 Fall 2025 Tournament Rules

1 Basics

Each agent controls one chicken on an 8x8 chessboard. The agents take turns taking actions. After both have had 40 turns, the game is terminated.

Each agent gets a total of 6 minutes to come up with its moves. (Not 6 minutes for each move. A total of 6 minutes for all your moves. The game will not last longer than 12 minutes.) If a player runs out of time, they lose the game.

The chicken that has laid the most eggs when the game terminates is the winner.

2 Start

Colors are assigned randomly: one chicken is white (Player A), one chicken is black (Player B). White goes first.

Chicken A starts on a randomly chosen black square on the left edge or right edge (not on a corner). Chicken B starts on the mirror image: If A is born on the right edge, B is born on the left edge – but both are born into the same row.

There are 2 trapdoors – 1 is randomly placed on a black square, 1 is randomly placed on a white square – at the start of the game. Squares near the center of the board are more likely to be trapdoors.

The trap doors are chosen in a weighted manner: We choose the dropdoor from all the squares of the color, weighted by how far they are from the edge: Squares on the edge have a weight of 0, squares inside those are weight 0, squares inside those are weight 1, squares inside those are weight 2.

You can see how they are chosen in `trapdoor_manager.py`.

3 Play

When it is an agent's turn it can take any of the following steps in any of the four cardinal directions:

1. A Plain Step simply changes its position
2. If the chicken is on an empty square of its own color, it can take an Egg Step. That is, it lays an egg in the current square and steps out of it. If the row and column add to an even number, Chicken A can lay an egg on it. Chicken B gets the odds.
3. If a chicken is not in a square that shares an edge with the square containing its enemy, it may take a Turd Step. That is, it drops a turd in the current square and steps out of it.

Each agent only gets to make five turd steps during the entire game.

If a square contains an egg, a chicken can't put a turd or a second egg there. If the square contains a turd, a chicken can't put an egg or a second turd there.

A chicken may move into squares containing its own eggs and turds. It may not move into squares containing its opponent's eggs or turds. Furthermore, it may not move into any square that shares an edge with a square containing one of its opponent's turds.

On an agent's turn:

- If any action is possible, the agent must take one.
- If it can't take any action, the opponent gets five eggs and the trapped chicken is transported back to its starting square.

When a chicken enters a square for any reason, it has some chance of feeling and some chance of hearing the engine running under the trapdoors. The two trapdoors have different engines, so the chicken can feel/hear the difference.

- A square that shares an edge with the trapdoor? 50% chance of hearing it, 30% chance of feeling it.
- A square that is diagonal to a trapdoor? 25% chance of hearing it, 15% chance of feeling it.
- A square that shares an edge with either of the ones listed above? 10% chance of hearing it, 0% chance of feeling it.

However, the chicken will have no idea where the sound/vibration is coming from – just that it exists. The probability of hearing and feeling are independent given the chicken's proximity to a trapdoor.

If you enter the same square a second time, the outcome (hearing/feeling) will be resampled and may be different from the first time you stepped into the square.

If a chicken steps onto either trapdoor, it is transported back to its starting square and the other chicken is given a four egg bonus. (If the transported chicken cannot enter its starting square, the game is immediately terminated.)

4 Environment

Each agent will be submitted as a python script called `agent.py` in a zipfile. That zipfile can contain code or data that you created, but the total size of the zip file may not exceed 200 MB. (Code or data created by anyone outside of your team is forbidden, but you are allowed to use code that was created by an LLM prompted by someone on your team.) The agent may not make any requests via the network. It may not read or write outside of its working directory. Python and the following libraries will be installed on the machine: numpy, PyTorch, JAX, Plyvel, and Scikit-learn. The tournament machine will be a x86_64 linux machine.

The state of the game arrives in an instance of a class called `Board` which has two `Chicken` objects and one `GameMap`. The positions on the board are (x.y) tuples. (0,0) is considered the upper left corner. Together, they contain the following information:

- Your turns (including the one you are about to take) remaining : int
- Your position: (int, int)
- Your starting square: (int, int)
- Your dropped eggs: a set of (int, int) tuples
- Your dropped turds: a set of (int, int) tuples
- Your remaining turds: int
- Opponents turns remaining: int
- Opponent position: (int, int)
- Opponent starting square: (int, int)
- Opponent dropped eggs: a set of (int, int)
- Opponent dropped turds: a set of (int, int)
- Opponent remaining turds: int

You will also receive a list of two (bool, bool) tuples:

- The first is whether you (heard, felt) the trapdoor on white (i+j is even) in your current location.
- The second is whether you (heard, felt) the trapdoor on black (i+j is odd) in your current location.

You will return a tuple representing your action: (direction, move_type)

Directions are Up, Down, Left, Right Move types are Egg, Turd, Plain

These constants are in `engine/game/enums.py`.

You want to make a state/action game tree? Look in `board.py` at `forecast_move()` and `reverse_perspective()`.

Finally, you will receive a function that you can call at anytime to get your remaining time.

5 Development

Unzip the provided zipfile anywhere on your hard drive. The source for the game system is on there, so you will be able to test your agent locally.

You will be working with a partner. It is strongly suggested that you keep your project in a git repository that you both have access too. This will make collaboration much easier. Life will be easier if you name your repository "3600-agents". Inside that, make a folder for each of your experimental agents. We recommend human names like "Gertrude".

Clone your repository into the directory that comes out of the zipfile. That is, `3600-agents` should be a sibling to the docs and engine directories.

For your first agent, you can just copy the `docs/Yolanda` directory into `3600-agents`. This will make Yolanda a chicken that makes random moves and logs out some fun facts as it does.

To run the game with Yolanda as both player A and player B, you can run the following:

```
python3 engine/run_local_agents.py Yolanda Yolanda
```

You should see a log of the two Yolandas playing the game. The history of the game will be in `3600-agents/matches/Yolanda_Yolanda_0.json`. (You probably don't want to add that to your repository. You might want to add matches to your `.gitignore` file.)

Please report any bugs you find to the ed discussion.

(On that note, there is a non-zero chance that there are bugs in the game system that will be found/fixed after you have started work. Check for announcements of "Download the new zipfile, please." If I were you, I would not put all the source code for the game in your repository – just the code for your agents.)

5.1 Multiple Files

Python has rules about what directory you can import files from. If you partition your agent into several `.py` files, you will probably bump into these rules.

The fix is to create a `__init__.py` file in the same directory as your `agent.py` file. For example, if my `agent.py` file wanted to import my `trapdoor_belief.py` file, I would put this in an `__init__.py` file:

```
from .agent import PlayerAgent
from . import trapdoor_belief
```

Then, in `agent.py`, I could import the `TrapdoorBelief` class in `trapdoor_belief.py` like this:

```
from .trapdoor_belief import TrapdoorBelief
```

6 Submitting

You feel ready to see how your agent will do in competition? Zip up the agent directory. For example, you might zip up the `Yolanda` directory into `Yolanda.zip`. It should contain a directory with an `agent.py` file inside.

Go to `https://bytefight.org/` Register. Create a team and add both members of the team to it. (Please use the same email address that we have in Banner/Canvas for you. This will increase the probability that you will get credit for your work.)

Upload the zipfile. Pick an agent to scrimmage against. Good luck!

7 Grades and Awards

Whatever you have uploaded at 11:59pm on Dec 2 will be used in the competition and determine your ranking and your grade.

For grading, we have reference agents:

- If you are above Henny in the final rankings you will get at least a 70%.
- If you are above Max, you will get at least a 90%.

(There will be reference agents for 80% and 95% within the next couple of days.)

If you are the top team, you each get a $150 Amazon gift card. Also, I'm taking you out for a nice lunch.