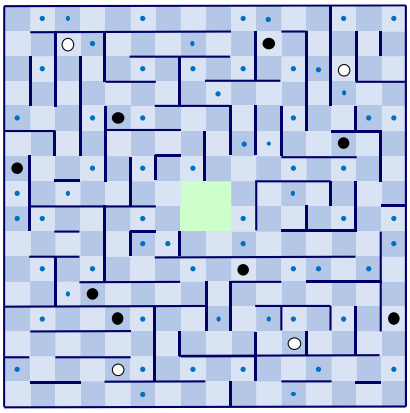
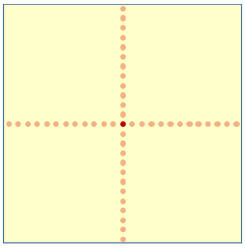
Undergraduate AI Capstone NYCU Spr2022 Group Game Project #2 Due 4/26/2022

The objective of this assignment is for you to design a game-playing agent. The game to be played is a modified version of multi-player Pacman game. The difference from the other project is that this game is not turn-taking (i.e., all the players move simultaneously), and decisions have to be made in real time.

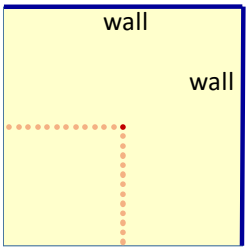
Description and rules of the game:



- Shown to the right is an example 16x16 field. The non-passable walls are randomly configured, with the constraints that all the cells are reachable from the center and there are no dead ends.
- Different items are initially located in the cells randomly (their numbers are to be determined):
  - Pellets (•): Eaten by the players for points.
  - Power pellets (○): Eaten by the players for super power.
  - Landmines (●): Collected by the players for later use to deter ghosts and/or other players.
- The central 2x2 region is the hub where all the players and ghosts start. There is no wall in and around the hub. No interactions between players/ghosts/objects occur here.
- Game process:
  - A total of 4 players playing together.
  - There are 4 ghosts in the field.
  - A round ends when all the regular pellets in a field are eaten. A new round is then initialized.
  - The game ends after a finite duration (e.g., 5 minutes (6000 time steps)).
- Movements of players and ghosts:
  - A cell has the size of 25x25 points (in-game coordinates).
  - Players and ghosts can only move along the horizontal and vertical center lines within a cell.
    - ✧ Positions at the four sides of the center are valid when not blocked by the respective walls.
    - ✧ Examples of valid positions (all discrete) are shown here.



  - Velocity (each time step = 0.05 second):
    - ✧ Player (regular mode): 5 points per time step
    - ✧ Player (power mode): 8 points per time step
    - ✧ Ghost: 5 points per time step
  - Player motion:
    - ✧ A player keeps moving along the original direction if command is received during the current time step, and stops at the center point of the cell if the current direction is blocked by a wall.
    - ✧ Turning left or right (when valid): If a turning command is received for the time step within which it passes the center point of a cell.
    - ✧ Reverse: No motion in the current time step, with direction reversed.



- Interactions between players/ghosts/objects:
  - "Overlap" between players/ghosts/objects is defined as a Manhattan distance of 12 points or less.
  - The player that overlaps with a regular pellet eats it and gains 10 points.
  - The player that overlaps with a power pellet eats it and enters power mode.
    - ✧ The power mode lasts until (1) the end of current round, (2) the player overlaps with an active landmine, or (3) 200 time steps, whichever occurs first.
    - ✧ If a player eats another power pellet while in power mode, the countdown of the maximum power-mode duration (i.e., 200 time steps) is reset.
  - The player that overlaps with an inactive landmine collects it for later use. A player can collect multiple inactive landmines.

- A player that owns a landmine can drop it at any time.
  - ✧ The landmine location is the player's location at the end of that time step.
  - ✧ A landmine dropped by a player becomes active after 5 time steps.
  - ✧ If a player or a ghost overlaps with an active landmine, it is immediately sent back to the hub and has to stay inside for 100 time steps.
- Overlap between players has no effect.
- Overlap between a player and a ghost:
  - ✧ Player in regular mode: The player is immediately sent back to the hub (random point) and has to stay inside for 100 time steps.
  - ✧ Player in power mode: The ghost is immediately sent back to the hub (random point) and has to stay inside for 100 time steps. The player gains 200 points.
- Any effects from the interactions are reset at the start of a new round.

Regarding the algorithm and implementation:

- You have a lot of flexibility in designing your game agent. With the players and ghosts all moving simultaneously, the classical minimax search for turn-taking games is not very suitable. Rule sets and route planning methods are useful. MCTS is still possible here, with all the game tree nodes for yourself, and the other players and ghosts handled by really simple models. You can also train your agent using reinforcement learning.
- It is required that you implement the algorithms yourself; you cannot use modules/libraries developed by others for game playing. When the TAs run the tournament, the game server and all the player programs will run on the same computer. There will be no outside connectivity and no GPU support, and there will be a time limit for each move.
- You can only implement the program in C++ or Python 3. The environments will be posted by the TAs.
- The communication between the game server and your program, running as a client, is via TCP. The TAs will provide instructions and sample codes on including the communication capabilities in your program.
- The TAs will provide a sample server program and simple client programs (to act as your opponents) for you to use during the development of your program.

Teams:

- The students should form teams of 1-3 members.
- Once teams are formed, provide the team information (team name, members' names and IDs, and team leader) on a discussion board in New E3.
- A team ID will be assigned to each team. You need to include this ID in both your filename and within the code.

The tournament:

- For each game, the players with the 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} highest scores receive 5, 3, 2, and 1 tournament points, respectively. When there are ties, the tournament points are evenly distributed among those players with tied scores.
- Each team's program will be played for the same number of games, with the same number of games in each of the 4 starting positions. The player combination of the games will be arranged randomly.
- Ranking is based on the total tournament points of the teams. Total game scores are used as tie-breakers for teams having the same total tournament points.
- The ranking will factor into 25% of your grade for this project.

Submission:

- The submission is through New E3. No late submission is accepted for this project. You should submit your program source code (following instructions from the TAs) and a report file separately. The report (maximum 5 pages single-spaced) should describe how your game AI works, your experiments and experiences, and contributions of individual team members. The TAs will announce later whether you need to submit executables. Only one submission is required for each team. Remember to list the team name, ID, and members in both the source code and the report.