COSC 322 Progress Report

Team 02 L03:

- Taii Hirano
- Yuki Isomura
- Subaru Sakashita
- Justin Tom

The Goal of This Project

Implement the Game Player using a graphical user interface. By the end of this project, our program will be able to play Game of the Amazons against other Game Players created by the other teams with moves that are well-optimized by heuristics.

Game Flow Breakdown

- Initialize the game player and get connected with the game client
- Retrieve the information of game state and actions
- Let heuristic find the optimized next move
- Send information about the next move to the client
- Keep playing until one of the players cannot move anymore

What Has Been Accomplished

So far we have implemented several classes: Tile, Queen, and GameBoard classes. Tile class contains two private instances: row and column. These instances indicate the position of the tile.

The Queen class extends the Tile class, consisting of private integer instances storing the previous row, and column. Since the Queen is either black or white, there is an instance variable indicating the colour of the Queen.

The GameBoard class contains an 11 x 11 array. The actual game board only holds a 10 x 10 field but there is an extra row and column inserted before the first row and column to make the first index 1. Thus, any Tile that holds an index of 0 either in a row or column is not available for Queen and Arrow instances to stay on.

Initially, inside the array, there are four black queens and four white queens. The blank tiles sit as null variables in the array.

The player with white Queens commences the game. After the player moves Queen, it shoots an arrow in any orthogonal or diagonal direction. The arrow does not move once it has been shot. The arrow essentially blocks the opponent from landing on its tile. Every move of the Queen limits the movement of the opponent, so the last one to move wins. The Queen and board classes are not completed because we have not yet implemented the Arrow class.

Things We Know at This Point

Game Structure

<u>Class Extension Tree</u>

```
Object
← TimerTask
        ← GameTimer
← JFrame
        ← BaseGameGUI
← JPanel
        ← AmazonBoard
← GameClient
← GameMessage
        ← AmazonGameMessage
← GameModel
        ← BoardGameModel
← GamePlayer
        ← Amazon
        ← Spectator
        ← HumanPlayer
        ← COSC322Test
```

About Overall Game Flow

The game is controlled majorly by GameClient. When a user runs the program, the player instance is initialized based on the username and password he specified. Then, check that the player instance does not have the game GUI connected and call COSC322.Go() to get connected with the game server. In this process, GameClient takes over the game flow.
Once the game is started, GameClient sends a message to the player instance. The player checks the message type to track the game flow and update the game state based on the message detail sent.
With updated information of the game state, let heuristic determine the next move reference to the score it calculated. As it is determined, send the information to GameClient and wait for the other player's move.

Task Separation

- Taii:
  Implement the heuristic function
- Yuki:
  Initialize the game board
- Subaru:
  Implement the tiles
  Keep track of scores
- Justin:
  Implement the Arrows

Timeline From Now

- Implement Tile class, DONE
- Implement Arrow class by the end of this week(March 5th)
- Finish up the game board class and Queen class by the end of next week (Mar 6)
- Implement the heuristic function by the end of the following week (Mar 13)
- Finish up the whole program by the week of the tournament (Mar 20)

Issues and Difficulties

- Unexpected cases while setting up the MAVEN Project.
- This project is not fit for Mac users. Some members are Mac users so it is quite difficult for them to work at home. We need to figure out how we could fix this issue so our productivity would increase.
- Lack of information in the main method of the GameClient class.