Patrick Lee and Charlie Huang

APCS Final: Super Othello

**Overview:**

In this APCS Final Project, Charlie and Patrick will be writing a super Othello game. The game must be a super Othello game because they have already coded an Othello game using a “stupid computer” for the artificial intelligence. With this new Othello game, Charlie and Patrick will implement different artificial intelligence through basic priority queue and Minimax algorithms.

Along with the different difficulty levels, the user who plays the game will also be able to choose the difficulty level as well as play multiplayer through the means of networking—meaning Charlie and Patrick will have to implement networking as well. Aside from implementing networking, Charlie and Patrick will also implement a way to create accounts and save user data as well as the state of a game so that the user can return to a saved game at any time.

Basic Priority Queue artificial intelligence will involve finding the best region for the computer to place a piece. The computer will base the area on which it places a piece based on areas known as “risk regions” which show where the computer has a good advantage. The computer will also see how many pieces it obtains from placing a piece there and multiply both values to get the final priority.

Design:

The Othello game will be really similar to that of what Charlie and Patrick coded in class. They will have an abstract OthelloPlayer class which both the human player and the artificial player will be based on. They will also have a Piece class which will constitute the pieces for the game. They will also have different classes for the different players—human and computer. In order for the game to run, they will have an OthelloWorld class which creates the game, and finally the OthelloRunner class which runs the game. This streamlined design helps ensure proper debugging and easy updates for when Charlie and Patrick decide to update or need to debug. \* More details are below in the final specification.

Testing:

In order to test the game, Patrick and Charlie will play the game several times to ensure that the artificial intelligence is working as specified. They will also write a JUnit tester class in order to test every method individually and ensure that it works as intended.

**Specifications:\***

**Classes**

* OthelloPlayer
* StupidOthelloPlayer
* MinimaxAI
* MediumAI
* HumanOthelloPlayer
* OthelloGame
* OthelloWorld
* OthelloRunner
* Piece/PieceDisplay
* PickLevels

**OthelloPlayer Class:**

**Function:**

This abstract class will assemble the player and have all the methods a player needs including whether or not it can play. This class provides the game with the players and without this, OthelloGame would not work at all.

**StupidOthelloPlayer Class**

**Function:**

This class is the easy artificial intelligence for those players who wish to play against a computer who isn’t very smart and picks any location possible to play. This class is not necessary as one could play against another player as well but this class would be helpful in order to play alone. This class extends OthelloPlayer.

**MinimaxAI Class:**

**Function:**

This class is the hardest artificial intelligence that we will implement in this game. This class uses a 2D array with set values in order to calculate which location is the best. It also calculates where it should place a piece to have the greatest advantage. This class also extends OthelloPlayer.

**MediumAI Class**

**Function:**

This class will be the medium aritificial intelligence that we put in. This class calculates where to place a piece by calculating which area will provide it with the best situation (most pieces and harder for the opponent to gain pieces). This class also extends OthelloPlayer.

**HumanOthelloPlayer Class**

**Function:**

This class is the human player. This class allows a human to play against the computer (or another human if so desired). This class uses one method, getPlay() to see exactly where the player placed in order to update the board. This class extends OthelloPlayer because it is also a player.

**OthelloGame Class:**

**Function:**

This class instantiates the world and creates the game. It also adds in the players and controls the game. It has the method playGame() in order to play the game until one person wins. When networking is implemented, it will be implemented in this class.

**OthelloWorld Class**

**Function:**

This class places the pieces at the starting location to start the game. It also gets where the player or computer clicked and is used in another method to place a piece.

**OthelloRunner Class**

**Function:**

This class sets up the environment and runs the game.

**Piece and PieceDisplay Class**

**Function**

This class creates the piece along with the PieceDisplay class to make the 3d effect in order to make the game look more appealing.

**PickLevels Class**

**Function:**

This class creates a GUI with radio buttons and two text fields for the player’s name. When the game is first run, this class is called in order to define what or how the player wants to play. Afterwards, the game shows up and the player can play.

