Jeffrey LeRoy, Lauren Fuller, Lance West, Eliezer Cabrera
Team 3
Multiplayer Game
3/28/14
CSCE 315

Functional Prototype – Game Mechanics

The first thing that was needed to even start playing the game was for us to figure out a way to log in to the client and server and then wait until another player wants to play. This was done by first having the log in screen be the first screen the user sees. This screen is very simple all it does is prompt the user to enter their login name. After this is done the user is then sent to another screen which is called the waiting screen which just simply waits for the server to send a message to the client that another client has logged in. The waiting screen also emits a message that basically just says you're waiting. After another player has logged in the server sends the client a signal and the gaming screen with the game board is shown.

Now that you are actually in the game the first thing to be done was the make the game board which was done using the game engine and placed spots in the correct position. Then using those positions of the spots we were able to then place the marbles in the right place. To allow the players to actually play the game and get the game mechanics working one of the first things needed was to be able to pick a marble to move. This was accomplished using an event handler for a mouse click and then dragging the marble somewhere until the mouse is released. We also had to find a way to program into the game what is a valid move and what was not. Hopping over pieces and chain hopping did not help in this case. The way this was done is there are two functions 'isValidMove' and 'getValidMoves' which determine when the piece moves if the move was valid for the rules of the game. The function 'getValidMoves' is what is used to chain hop that's why it is recursive as well. Once a player makes a move the client they are using sends a message to the server called 'move' with parameters such as

xPosition and yPosition of the new marble. The server then sends a message to all the other clients in that game and updates their board with marble's new position on the game board.

Now to actually get meaningful play there has to be a way for the players to take turns for moving their pieces. This was accomplished by having a Boolean variable called 'myTurn' which is set to true for the second player that entered the game not the first that was waiting on the second player. Once the second player makes a move it sets its 'myTurn' to false and the server sends the move message to the other client which when received turns that clients 'myTurn' to true. You cannot move a marble if the variable 'myTurn' is false. The gaming screen also emits a message that changed to "Your Turn!" when it is your turn and "Opponent's Turn!" when it is the opponent's turn.

Finally now that the players take turns moving their marbles, the marbles can only make valid moves, and are able to make moves in the first place, a player must be able to win the game. This is done by after every move a function called 'hasWon' is called which tests to see if all the marble's home variable, which determines if the marble is in one of the home bin spots (triangle opposite to starting area), is true. If one of marble's home variable is false it will return false because you have not won the game. If the player has won the game because 'hasWon' returned true, then the client emits the message "You Won!" and sends a message to the server. The server then in turn sends a message to the rest of the clients that they have lost and their client emits "Game Over! You Lose! Player (number) Wins". After that the game is over.

This pretty much explains how our code works in a nutshell and how it allows for at least two players, at least, player Chinese Checkers and it all actually be correct. It also helps the players play by the rules since the client checks to see if the move is always valid and if it isn't it will not allow the player to move to that spot.