Matthew Duerr

May 1, 2016

Final Project

Checkers

The rules of checkers is quite simple, two teams are started on different sides of the board and the object is to capture all of the opponent’s pieces by jumping over them. A jump is a move when on the same diagonal an enemy piece is located between the user’s piece and an empty square. When this occurs the player can move their piece to the empty square and capture/remove the enemy piece. Each player at the beginning of the game can only move in the forward direction, if a piece makes it to the other side of the board then that piece is a considered a king, and a king piece can move in either direction. If a player cannot jump a piece then by default they are allowed to move to any empty diagonal square located next to the piece.

The program checkers is consist of six different classes, and an interface. The DataCollection interface contains methods that the collection class will use. The collection class contains the node class, in the collection class are methods to modify or get data from the node data structure. The node class is the programs data structure, since a node is a reference to a node and a checker piece, the class has all methods to create a node and get data from a single node. The Checker piece class is the design of a checker, how a checker is created or painted, and all its properties are stored in that class.

Those four classes were outlined first so the Checkers and CheckersGame class could be better clarified. When the program is started the Checkers Class is the first to be called. When it is called it creates a game of checkers by calling the CheckersGame class. Inside the checkers game class, is the GUI and all methods needed to convert some user’s interaction, in other words the board and buttons are displayed on screen, and since the player can only interact using a mouse, the coordinates of where the mouse was clicked must be converted so the program knows what column and row were clicked.

The CheckersGame class contains a doClickSquare method that handles all valid user interaction, the doClickSquare has to convert the coordinates clicked into some useful action wanted by the user. There are helper methods that doClickSquare uses in order to do this, doClickSquare test of the user selected a piece or an empty square. If it was a piece then select it, since the user plans on moving this piece. If the user clicks on an empty square, then most likely they are trying to move their piece there. The helper methods tell doClickSquare if the attempted move is valid, if the move is valid and a valid jump then the piece being jumped must be removed.

The program must handle whose turn it is, what piece is selected, whether or not the game has ended and how a game is one. The CheckersGame class is the bulk of the program, it contains the rules of the game and interprets user interaction. Each possible move must be tested and each click has to be interpreted.