alexela

Alex Grzesiak Alex Nelson

CS 470: Artificial Intelligence

Assignment 4.1

Halma Game Player

April 24, 2017

Overview:

The architecture of this program is broken up into a Board class and a Halma class. The board class is just responsible for setting up the board. This broad design is that the board class can be used to create any board for any game using a grid layout. The Board class has the following methods:

___init___	Gathers all images needed for the board, sets up the frames, initializes variables used for board keeping, sets every tile to a blank button, and then from the optional number of pieces passed in sets all the pieces for red and green.
clearAvailableSpaces	Take the board and clears all spaces that don't have a piece in them. Used to clear the highlighted spaces as well as the spaces with a temporary piece signaling where a move originated.

Then the Halma class uses the board to handle moves and determining the gameplay using the semantics of the game Halma. The Halma class has the following methods:

___init___	creates the board, initializes all variables used to keep gameplay
emptyButton	the function given to each button by default. When an empty button is clicked, this function will get called so it either does nothing is there was not a piece selected before, or handles the move piece procedure if there was
occupiedButton	once a piece is added to a square, it is given this function as its listener. This handles both green and red moves since there is only a couple of lines dealing with the two colors. This only takes care of adding a piece to the variable holding the selected piece and the potential someone tried to move a piece into an already occupied space
generateLegalMoves	Takes in a single space and generates all possible moves that piece cam make including jumps
generateAllLegalMoves	Takes the board and using the global turn variable, returns a list containing all legal moves for every piece on the board.
findJumps	Takes in a single space and recursively finds all spaces that are available to that piece via jumping
isWin	Takes the board and checks to see if either "end zone" is occupied by the opposing team

Effort Description:

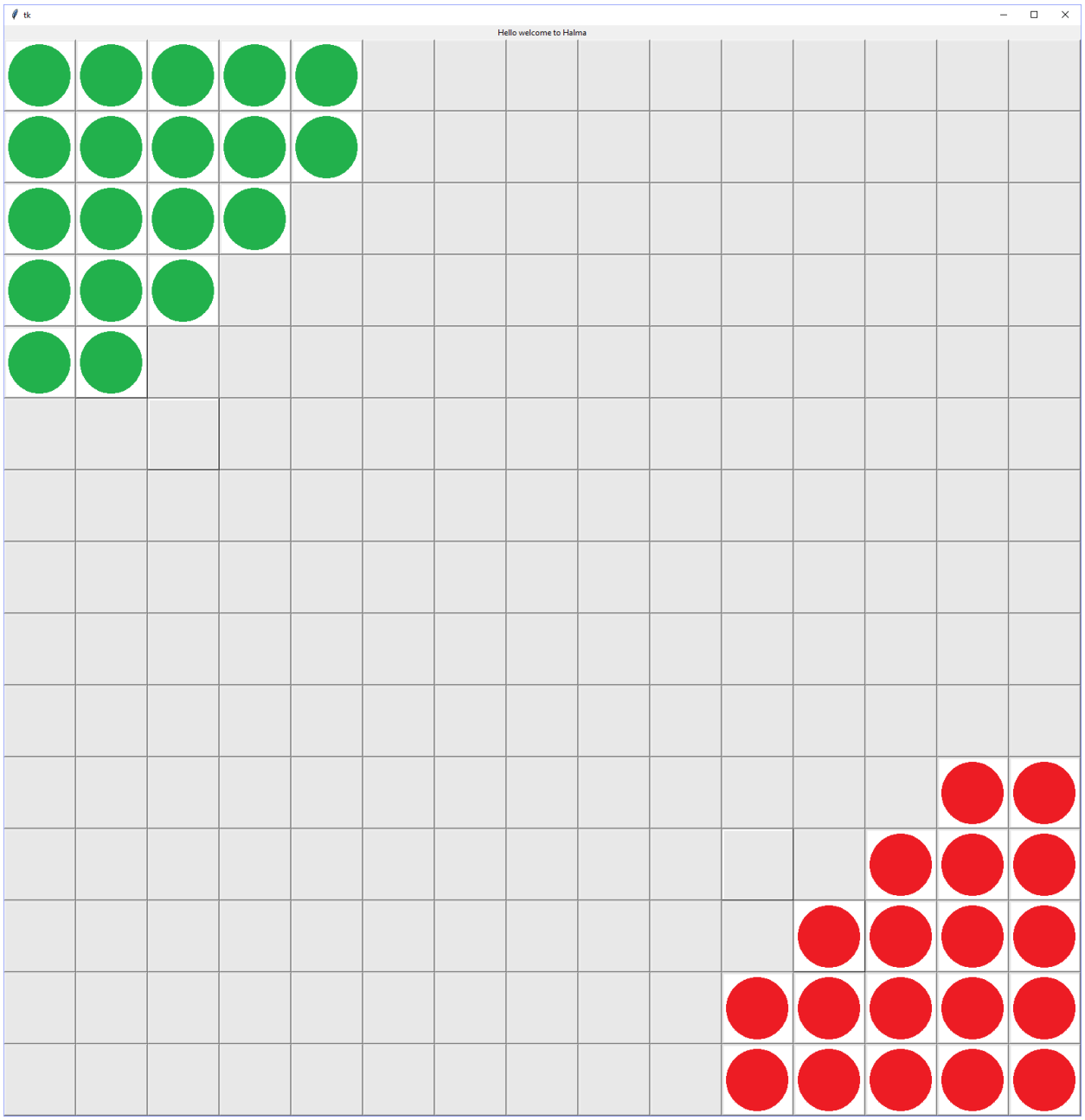
It was decided that Alex Nelson would do the GUI and Alex Grzesiak would do the implementation for move generation, win detecting and the moving.

Task Description	Assigned to	Percentage of Effort	Completion Notes
GUI design	Alex Nelson	40	
Move Generation	Alex Grzesiak	10	
Win Detecting	Alex Grzesiak	20	
Move Functionality	Alex Grzesiak	20	
Integration	Alex Grzesiak	5	
Writeup	Alex Grzesiak	5	

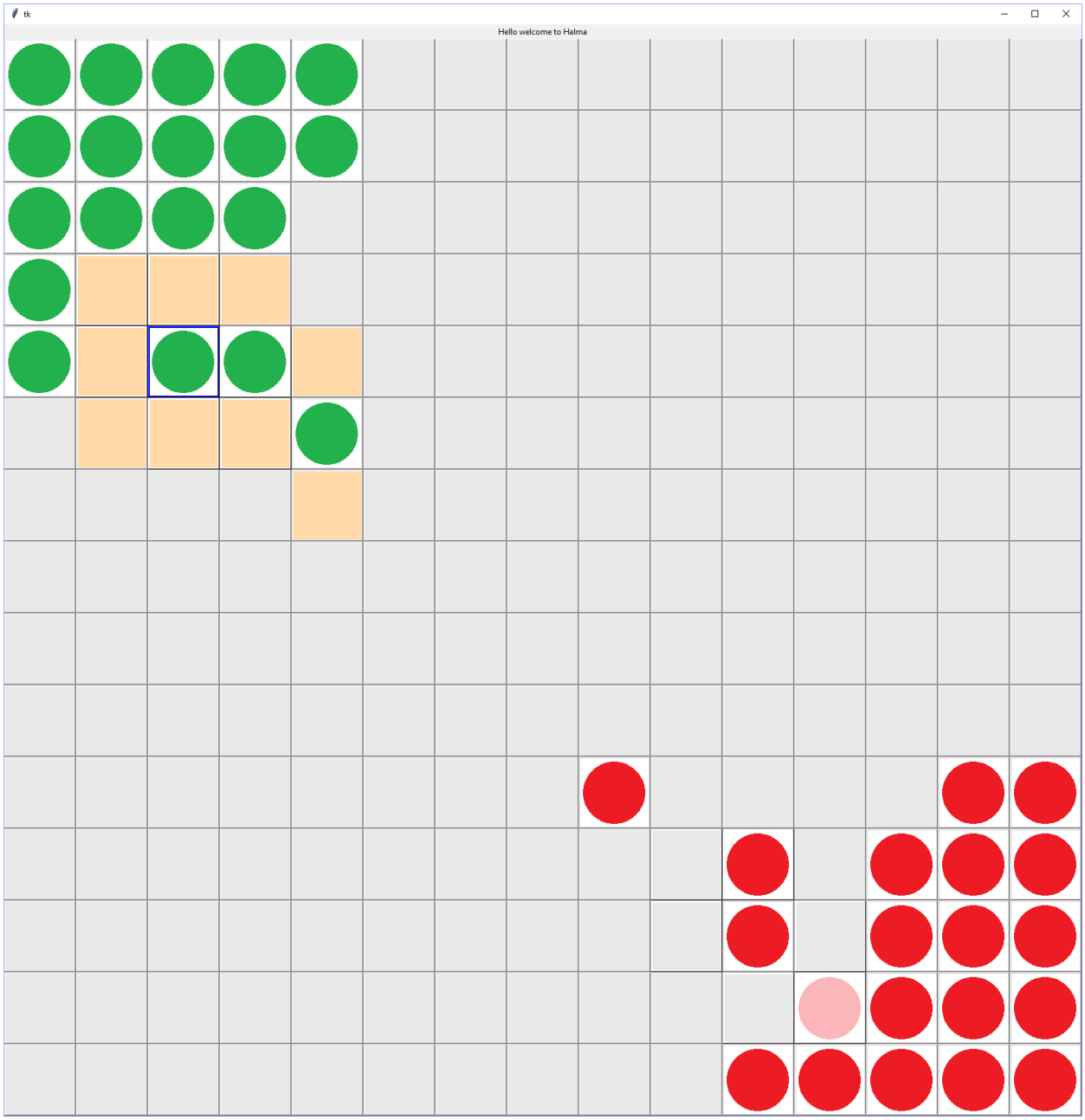
Functionality Checklist:

Functionality	Percent Complete	Notes
Graphical Board Display	100	
Board Updating	100	
Move Generator	100	
Win Detector	100	
Move Method	100/0	For human vs human, the piece movement is built into the buttons but when an AI player is introduced, the move function will take care of the movement when "thinking" is involved.
Fully Functional Play Mode	100	

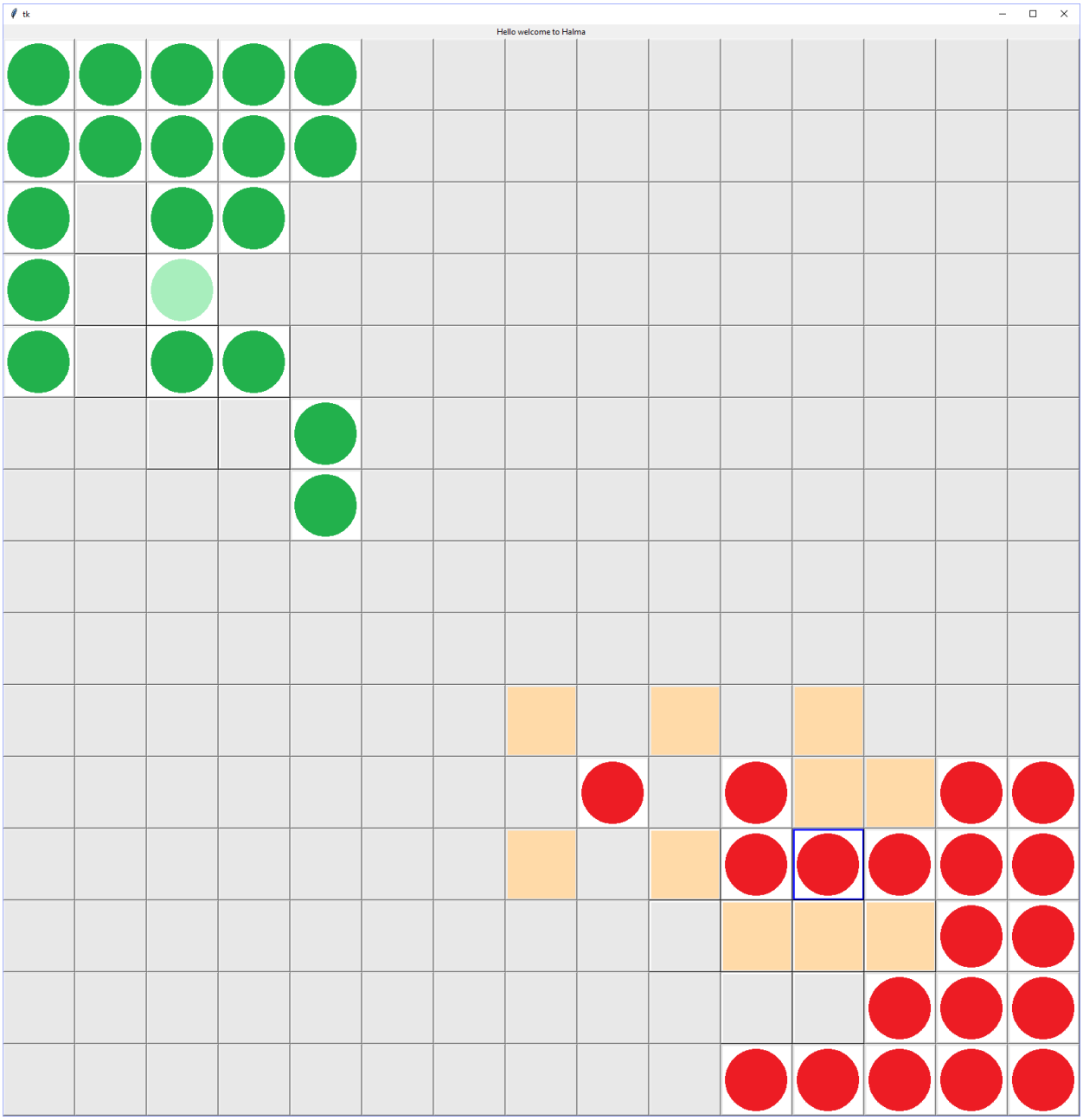
Demos



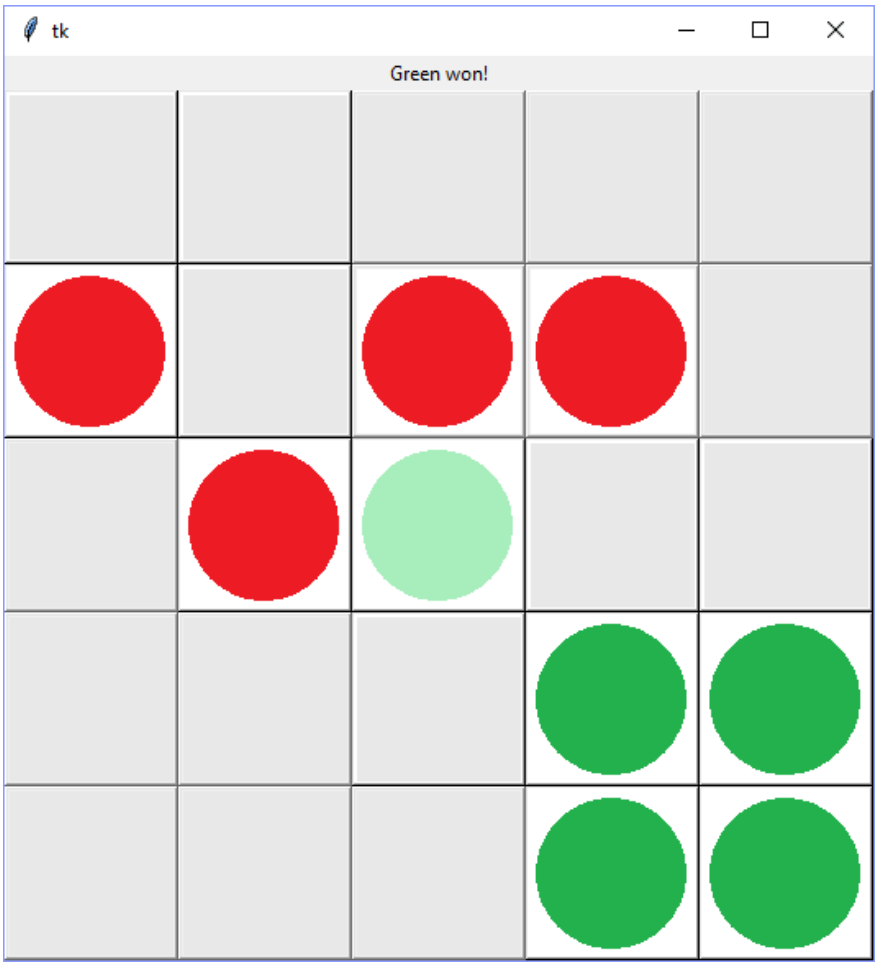
This is the opening screen upon starting the program.



This shows a bit of piece movement. Red just had a turn and shows how the space that the piece just came from is shaded a lighter shade of red than a regular piece. On the green side this is the view you get when a piece is clicked. The possible moves are highlighted which would include the adjacent spaces as well as the moves obtained by jumping.



This is another view of the piece moving and the highlighting of possible moves by a team. You can tell which piece is selected because there is blue highlighting around the piece that is selected.



Using a simplified board so the game woulnd't take so long, this is what it looks like when someone wins. There is a notification that Green won and the functionality of the buttons has been disabled so that no more moving can take place. A feature that is going to be added is an undo feature so that an error in the game playing won't result in a total loss in the game during tournament play.