ENGR-102 Section: 507
Noah Saria
Minh Nyugen
Brendan Hon
Vijay Seetharam

Group Project Design Document:

Game - The game our group decided on for the group project is Checkers.

RULESET:
The game is intended to function with the standard ruleset of the game:

- Classic 8x8 grid type board with 24 pieces (12 pieces each for two colors).
- Red set of pieces will always open with the first move. Each color will alternate their turns
- Pieces are set to strictly move diagonally. (Single pieces are limited to moving forward and King pieces can move forward and backward)
- Capture will be performed diagonally by “leaping over” an opposing piece in a valid empty space.
- The program allows for multiple captures. However, each movement after a valid capture MUST also be a valid capture move. The player can choose to perform another capture if applicable.
- A piece is “kinged” when reaching the furthest opposing row. A kinged piece is allowed to move forward and backward diagonally. It may also combine captures in several directions on the same turn.
- The winning player is decided by capturing all their opponent’s pieces.

**Our group used this website in designing the rules for our game:
https://www.ultraboardgames.com/checkers/game-rules.php

Program Structure - This program utilizes module pygame 2.1.2 for the Graphical Interface of the board.

To install, users must type `pip install pygame` in the terminal of their Python IDE.

****User MUST have the Pygame module installed in order for the program to work properly!**

**Link to Pygame on pypi.org:
https://pypi.org/project/pygame/

Function
Class

The basic structure of the program utilizes several class methods to define the following:

- Grid Movement - This function determines move selection via mouse input. (Selecting piece and empty space with the mouse cursor)

- Board - This class is responsible for the display of the board and creating a functional layout for the game. Also updates pieces, capturing, physically moving pieces etc.
    a) Drawsquares - Creates the graphical display of the checkerboard
    b) createBoard - Creates the functional layout for gameplay via a list of lists
    c) draw - Draws pieces on the board
    d) remove - Removes pieces from the board via capturing movement options
    e) Move - Moves piece from current position to valid position.

- Piece - This class is responsible for the mechanics (such as movement, color, and kings) of each checker piece.
    a) calcPos - calculates coordinates of a piece
    b) make_king - modifies class variable king true
    c) draw - draws two circles (outline and main) for a piece given position, if king is true, draws crown on top of a piece
    d) Move - updates position, handles condition to become king
    e) posIfGoDirection - returns coordinates of new position based on possible moves, and jumping condition
    f) posibleMove - returns a list of total possible moves, calculated from posifgodirection
    g) displayPosibleMove - shows possible moves to the user as gray circles for clicking

- Game - Class that controls game conditions, turns, and event loop. It also has code for single-player mode AI.
    a) changeTurn - flips turns
    b) gameIsEnded - returns true if there are no more opponent pieces on the board
    c) play - contains event loop and conditions for user input such as quitting, selecting and moving pieces.
    d) ai - contains decision making for ai moves in single player
    e) winingWindow - displays victory screen

** Some additional functionality we added:

- AI is “smarter” and will go for capture or king movements when available
- Graphical UI for users to choose which mode they want (2P or AI play)
- Separate graphical display for victory screen

Work Statement:

Noah Saria

- Wrote design document
- Helped code AI behavior and two-player functionality of the program
- Helped code graphical UI for the board and pieces

Minh Nguyen

- Wrote code for board display and functionality
- Coded grid movement based on valid mouse input
- Helped code the king mechanic and condition for individual pieces

Brendan Hon

- Coded conditions for the program to determine win/loss/tie
- Coded graphical UI for the victory screen
- Document editing and docstrings

Vijay Seetharam

- Coded valid piece movement for the program (single and King piece)
- Coded single/multi “eating” options for pieces
- Coded alternate approach for AI and Functional Testing