COMP2911 Quoridor Project Report

Team membership

z3361119 Lee Chun Yuan

z3376609 Daniel Cai

Summary

1. We have passed all the provided tests.
2. We have implemented Computer-Human and Computer-Computer plays. For the computer we can choose difficulty of either Dumb or Smart which uses the min-max algorithm with alpha-beta pruning. Dumb is depth of 1 and Smart is depth of 2.
   We have implemented save/load game, undo/redo moves and reset game.

More Details

1. The main four classes are the game, board, tile and abstract move classes. The board class has both player stats and the walls currently on the board. It has a current player number to track whose turn.

   The tile class holds the position of the player/wall. A player tile has an additional player number variable. The wall tile has two additional fields which are the top and right walls.

   The abstract move class has move parser, validate and generate move functions for both AI and Human. It has both apply and undo move functions which consists of both wall and player moves.

   The game class holds most of the user interface functions and has the AI players.
2. Our AI player only generates wall movements if the opponent has reached his/her 3rd row. The heuristic cost is calculated by getting the difference of the current player's distance to goal and the other player's distance to goal. Add this to the difference of walls each player has. That will be the final heuristic cost.
3. Save and load games things saved are human/computer for both players, difficulty of AI if any, undo & redo move lists, current player, turns for both players, round of game, positions of both players and lastly the wall count of both players.

   For undo/redo moves, it will undo the previous player’s move & it’s own move vice-versa for redo.

Design Reflection

We split Tile class into wall and player we felt that it allows better understanding of the program. We could improve by having more interfaces and more refactoring which will shortened the code for each .java file. We should try out generics too.

Implementation Reflection

Everything depends on the board class so to model outside of it is a limitation.

Teamwork Reflection

We did not schedule and dividing our work properly thus we have some overlapping tasks. In the end we both finished the game.