Chinese Checkers

Project Report

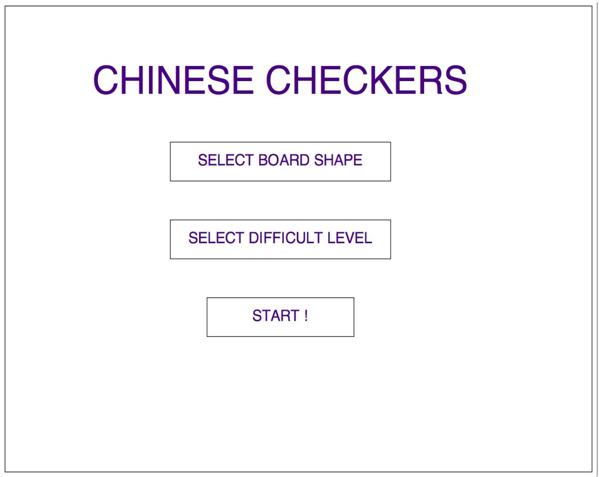
CS154 - Abstractions and Paradigms in Programming

Team

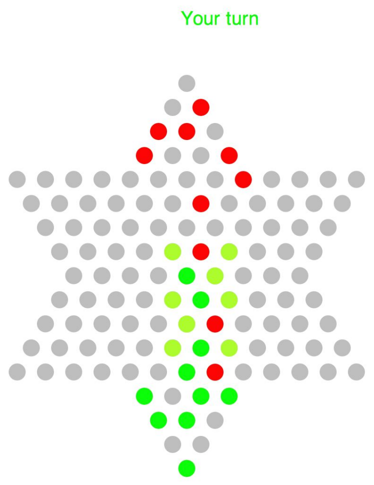
1.) Sriram Yenamandra : 16D070017
2.) Syamantak Kumar : 16D070025
3.) Kartik Khandelwal : 160070025

Problem Description

The aim of this project was to develop the game of Chinese Checkers and make a basic chinese checkers playing engine. The option of either a two-player game or a one-player game has been provided. We have also simulated game between two AI's to compare different heuristics.



Home screen



A sample instance of the game vs AI (red pegs), the possible next moves from a start peg have been shown in light green

Implementation Details

We have included two different boards (can add more too). We have abstracted the Board Design by a variable board type. To emulate different boards we have created a 2-d vector of size 30 by 30 and then selected a subset of elements to be included as part of the boards depending upon board type.

When you click on a peg, it highlights all the possible moves for that peg. To implement this we store a list of next-moves with their path for each selected peg. The paths are needed in order to animate the movement of pegs from the start to the destination slot.

We have used Minimax algorithm with Alpha-Beta pruning to get the best next move for the AI.

The evaluation of the board is based on the following heuristics :-

1.) The first heuristic tries to maximise the movement of the peg along the vertical direction.
2.) The second heuristic tries to minimise the horizontal distance of the pegs from the center line. This ensures that pegs don't stray too far from the center.
3.) The third heuristic tries to prioritise the forward movement of back pegs as compared to pegs far ahead. This ensures that the later on the back pieces don't have to make just single hops to catch on with the other pieces.
4.) The fourth heuristic tries to move the pegs in the destination slots towards the edges of the destination triangle so as to create space for the other pegs to enter.

The weights for each of these heuristics have been passed as parameter to the minimax algorithms so that we can easily toggle these weights to compare effects to varying them.

Clever coding techniques:

1) In order to improve the speed of minimax algorithm, at each level of depth we have filtered the next move list to remove those moves that move the pegs too far behind from the initial positions. It greatly cuts down on the time it takes evaluating this frivolous moves.
2) The next move function to compute all the possible next moves through multiple hops has been cleverly coded using higher order functions.
3) We have changed the ordering of next-move list so that alpha-beta cutoff happens as early as possible. This has led to significant reduction in time to compute the optimal move.
4) We have abstracted every element that can be reused like board, slots, pegs. By changing a single value we can display different boards with different slots and pegs colours/patterns.

Input and Output

When a player clicks on the pegs he wants to move, the window will highlight all the slots where legal moves are possible. The player can then click on one of these possible moves to move the peg there.
The window animates the movement of the peg from the start to the end slot.
There are also options for choosing board shape and different difficulty levels.

Limitations and Bug

The AI sometimes struggles during the endgame. Sometimes it has difficulty finding the most optimal moves to end the game. It however wins the game, if not in the optimal way!
Also you cannot choose the colour of your pegs manually.