Advanced Diploma of Professional Games Development

Assessment 2 - Complex Game Systems

Technical Design Documentation by Daniel Wormald

—

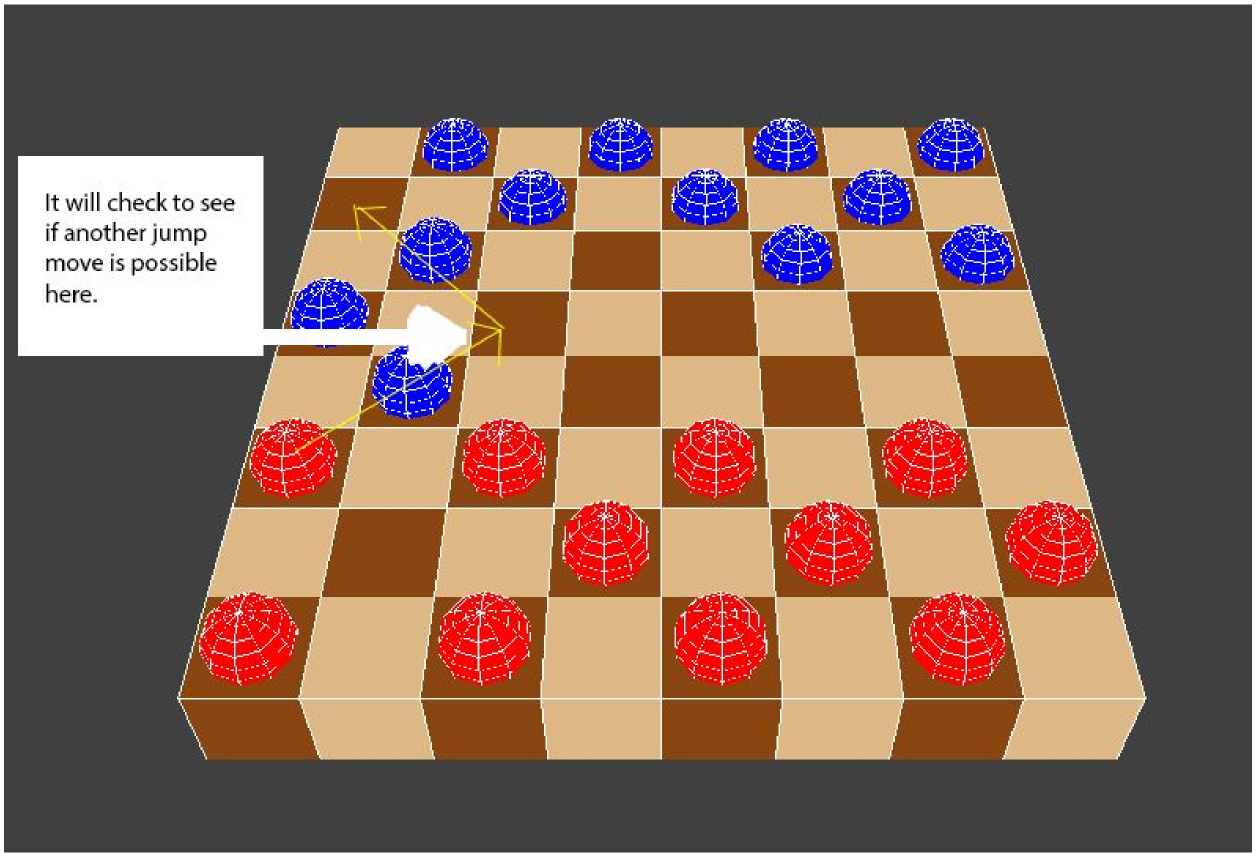
Summary

In this assignment I have used the knowledge gained from the Complex Game Systems subject to create a checkers game that uses different AI algorithms. In this documentation I will explain what techniques were used to implement the checkers AI.

I chose A.I. because it is a field of game programming that I'm interested in working in and I believe that good A.I. is necessary in any game.

The Checkers game was made using Gizmos "addAABBFilled" for the background squares and "addSphere" for the checker pieces.

The A.I. algorithm i chose would closely relates to the monte carlo tree search. When it is the A.I.'s turn the program starts off by iterating through all possible checker pieces on the A.I.'s team and checks whether those pieces have any possible moves available.
For the checker pieces that have available moves it then checks whether any of these possible moves will result in jumping over an opponent's checker piece (JumpableCheckPieces). If the checker pieces has any of these 'JumpableCheckPieces' it will loop through and check if there are any moves available for the checker pieces after they jump over the opponent's checker piece. It will complete the move that provides the best result for the A.I.



The only issue I had was not being able to fully implement the monte carlo tree search. My method doesn't simulate player actions and only relies on the current state of the game. If I had more time I would like to compare the Monte Carlo Tree Search and Minimax and pick the one that I thought would work better with a game of checkers. I would also like to include scores for each of the moves that are played out.