COMP 4106 A1 report
Ankush Varshneya – 100853074

This game is highly customizable, the user can change the dimensions of the board, the number of pawns, and the frame rate for solving the solution.

The state space consists for the following a knight, an array list of Pawns, and the board dimensions. The knight object contains knight's position and the pawn object contains pawn position with the alternative direction.

This state is stored in a node object along with the parent node to represent a tree structure of nodes. For DFS these nodes were arranged in a stack, for BFS as a queue, and for ASTAR as a priority queue; as well all the methods contained a close list to maintain the visited nodes to avoid cycles in the tree.

As the number of pawns and\or board dimensions increased the search got slower, DFS was the only one that returned a solution fast enough for really high dimensions and/or number of pawns, the other searches took over 2 seconds.

For my ASTAR heuristics I choose:

1) To get the sum of all Euclidean distance(s) from the knight's current position and all the pawn(s) currents position(s). This expanded less nodes then BFS and DFS but was not the best. The reason for choosing this heuristics was because we can be sure that the knight will travel, in the best case, at least the distance between all the pawns and itself, so this works as a good under estimate of actual cost in the best case.

2) To get the sum of all Manhattan distance(s) from the knight's current position and all the pawn(s) currents position(s). This expanded less nodes then BFS and DFS and Euclidean heuristic and was the best. The reason for choosing this heuristics was because we can be sure that the knight will travel, is the best case, at most all the vertical and horizontal distances between all the pawns and itself, so this works as a good overestimate of the actual cost is the best case.

3) To get the sum of all Euclidean and Manhattan distance(s) from the knight's current position and all the pawn(s) currents position(s). This expanded less nodes then BFS and DFS and Manhattan heuristic and was the 2nd best. I chose to add the heuristics the better get an average of the two as I was under the impression that the average of an over estimate and under estimate would get me a closer overall estimate. But it turns out that the overestimate was slightly better than the average.

Some statistics for 10x10 board with 5 pawns:

| | | |
|---|---|---|
| Using BFS Auto Solve:<br>Awnser found in 8 move(s)<br>Took 0.404 seconds<br>Expanded 1788 nodes | Using DFS Auto Solve:<br>Awnser found in 69 move(s)<br>Took 0.001 seconds<br>Expanded 82 nodes | |
| Using A* Euclidean Auto Solve:<br>Awnser found in 13 move(s)<br>Took 0.015 seconds<br>Expanded 25 nodes | Using A* Manhatten Auto Solve:<br>Awnser found in 9 move(s)<br>Took 0.006 seconds<br>Expanded 11 nodes | Using A* Euclidean Manhatten Combo Auto Solve:<br>Awnser found in 13 move(s)<br>Took 0.002 seconds<br>Expanded 21 nodes |