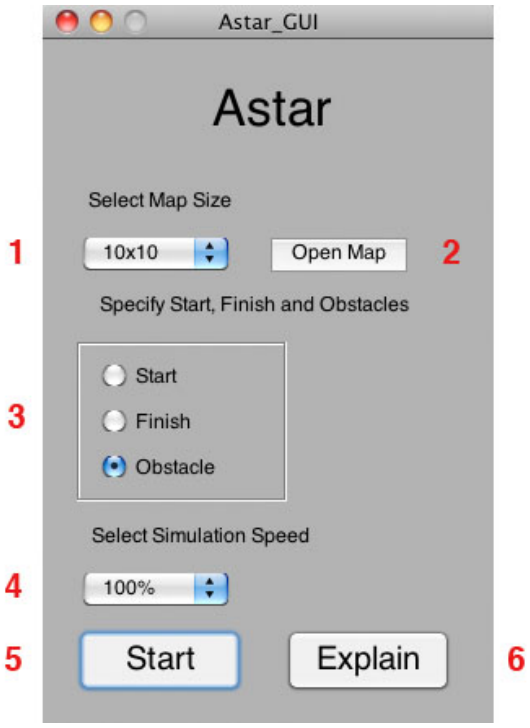
A* (A star) Path Planning Algorithm

1. GUI explained



1. Map Size - defines the size of a map in squares. Maps over 20x20 work slow because of large number of objects in map representation
2. Open Map - opens new window with map
3. Specify Start, Finish and Obstacles - select radio button and click with the mouse on the map to place start (green) and finish (red) points as well as obstacles (black).
4. Simulation Speed - option to slow down the simulation. For manual mode - once started, press any key to perform next step
5. Start - start the simulation
6. Explain - opens this help document

2. Algorithm explained

A* path planning algorithm uses simplified environment to conduct search. Area is divided into number of squares and the path is found by figuring out which squares should be taken to get from starting point A to finish point B. Centre points of these squares are called nodes.

Search begins at the starting point A by analysing the surrounding area - 8 neighbouring squares. First thing is to check if they are “walkable”, thus not an obstacle or already visited. All “walkable” items are added to the open list, which contains nodes that were seen and still can be visited, seen as yellow squares in figure 1. For each of these, point A is saved as a “parent” - it will be used for finding the shortest path. After all 8 surrounding nodes are analysed, current square is deleted from the open list and added to the closed list, containing all visited locations, seen as blue squares in figures 1 and 2.

Previously described process is repeated for every step taken. However, the question is how to determine, which square to choose next? Path scoring is used to solve this problem. Total cost of each node is determined by the following formula:

F = G + H

G - cost to move from starting point A to a given square on the grid following the generated path

H - estimated movement cost to move from a given square on the grid to finishing point B. It does not take into consideration any obstacles, just assumes that it is possible to move in a straight line - Euclidean distance.

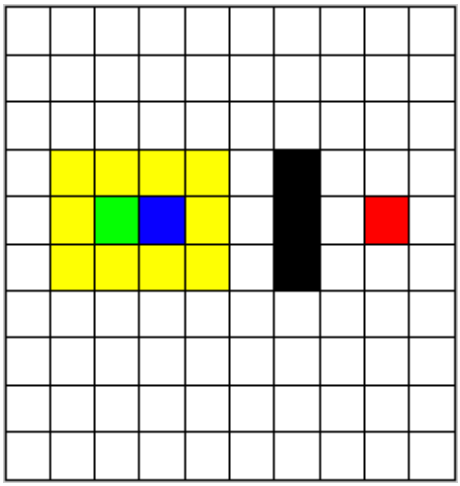


Figure 1: beginning of search
Green square - starting point
Red square - finishing point
Yellow squares - items in the open list
Blue squares - items in the closed list
Black squares - obstacles

These costs are calculated for each element in the open list. The node with lowest F score is chosen as a next best step to take. At the same time, previously mentioned square “parents” are adjusted at every step to ensure that shortest path through already visited area back to the starting point A can be found.

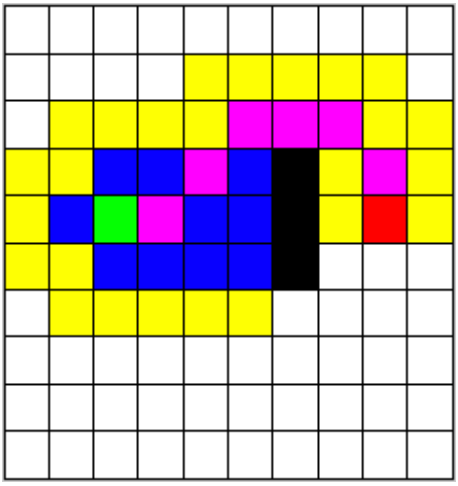


Figure 2: Shortest path found - shown in magenta

These steps are repeated by changing current square to the one having the lowest score from the open list and adding already analysed node to the closed list. Search stops when finishing point B is added to the open list, meaning that it is just next to the current one and goal is reached, or when open list becomes empty, meaning that all squares in the map were analysed and there is no possible path to reach the target.

If finishing point B is successfully reached, the shortest path is found by using hierarchy, checking parent nodes of elements in the closed list, and moving through them until starting point A is reached. Because parents are always updated when the algorithm progresses, this ensures that shortest possible path is found, seen in figure 2 as magenta squares. Path length is then calculated.