APPROACH

1. Generate Start and End points of the maze
   1. Place the start point at the edge of the maze – at most a distance of 10% of the total height and width of the maze away from the edge of the maze
   2. Place the exit of the maze at least 40% of the total height and width away from the start point.
   3. A screen shot of a computer

      Description automatically generatedProblems: for some values of the start position, there isn’t a valid end position given the criteria above. When this happened, the program would loop endlessly trying random points on the maze, none of which are valid.
   4. To fix this, an attempts counter was added, which stopped the search for an exit point after 10 failed attempts, and restarted the process with a new start point.

A screenshot of a computer program

Description automatically generated

1. Random walk algorithm until the player reaches the exit point of the maze. Populate all the other spaces with 60% ‘#’ and leave the path to the exit open.
   1. A black background with white dots

      Description automatically generatedAll directions have the same chance of being picked. The path taken by the random walk algorithm is shown as “.”.

A black background with white lines and dots

Description automatically generatedA black background with white dots

Description automatically generatedA screenshot of a computer

Description automatically generated

Way too much empty space around the start square. To see the differences between the maze path generations, we will use a maze of 10 x 30, with fixed start and exit positions. Also a lot of steps taken to reach the end of the maze. A black background with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

* 1. Instead of making it completely random, we could guide the walk towards the exit. By finding the distance between the adjacent (left, right, up, down) positions and the exit node, we can assign different probabilities for picking each movement direction. The smaller the distance the higher the probability of picking that square. A black background with white text

     Description automatically generated

A black screen with white text

Description automatically generated

A lot less steps taken to reach the end, but it takes almost 35 times longer per maze. The route is a bit more guided, but there is still a lot of empty space. 574 steps is also much more steps than needed. This is probably because retracing steps is valid. To fix this, the maze struct will be changed, so that the data points in the maze struct is a node struct containing the data and a variable to keep track of whether the node has been visited or not. Visited nodes are not valid moves.

A screen shot of a computer

Description automatically generated

* 1. Using this approach causes another problem, when the algorithm gets itself trapped by nodes it has already visited. The left, right, up and down values are the probability of each direction being picked – since it is 0 for all options, the code crashes. A potential fix could be to check if the nodes to the left and right, or top and bottom each possible node to move to has been visited or not. If either have been visited, then moving there could potentially trap the algorithm, hence in that scenario it wouldn’t be a valid move.

A screen shot of a computer

Description automatically generated

The algorithm still gets trapped, and sometimes it sections itself it a way that it is impossible to reach the exit node, without backtracking over previously existing nodes. However, introducing backtracking would give us exceedingly vast maze solutions from the previous iterations. Too many problems with this current approach, so I will try a new one.

* 1. The problem with this current approach is the even when I get lucky, and it works, the path to the solution is sometimes too wide, and even with the walls being added on, it is quite clear to see the path to the exit. This time, I will use the same algorithm to find the distance between adjacent cells and exit, but instead of using the values to assign probabilities, I will just choose the value with the shortest distance as the cell to move to. This is just a simple path finding algorithm to the exit node.

A black background with white dots

Description automatically generatedA black background with white letters and numbers

Description automatically generatedA screen shot of a computer screen

Description automatically generatedAs mazes go, this is quite boring, and the path is clear to see at a first glance. Hard mazes are interesting as there are different paths to go down, which seem like the correct solution but end up being dead ends. Thus if we create a bunch of dummy exits for the path finder to go to, before it reaches the end, then the maze will be more interesting.

The number of dummy exits generated is based on a percentage of the longest side of the maze – the larger the maze the more exit nodes will be generated. The only criteria currently is that the dummy nodes can’t overlap with each other or the start and exit square. While this maxes the maze more interesting, sometimes the dummy nodes are too close to each other or to the start and end nodes, leading to the path finder going over same paths throughout its iterations. By making the dummy nodes not being able to spawn on a visited node, and creating a minimum distance which the nodes should be away from each other, this problem can be fixed.

A screenshot of a computer screen

Description automatically generated

* 1. The mazes become a bit more interesting as there are multiple viable paths to go down, however, it is still quite straight forward. To have some more variations in paths, instead of finding a path from the start square to the exit, we can pick a random start path from cells which have already been visited and find the dummy exit from there. This will make the paths branch more.

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

* 1. This works okay for smaller mazes, but with larger mazes, there are too many straight lines, which make the maze look weird. Another thing to note, since the path finder just finds the most direct path to the exit, by just looking at the start and the exit, the path is pretty clear – the route to the exit has no detours to make the maze challenging. We could randomly generate points in the maze that the exit route must pass through before reaching the exit point to fix this.