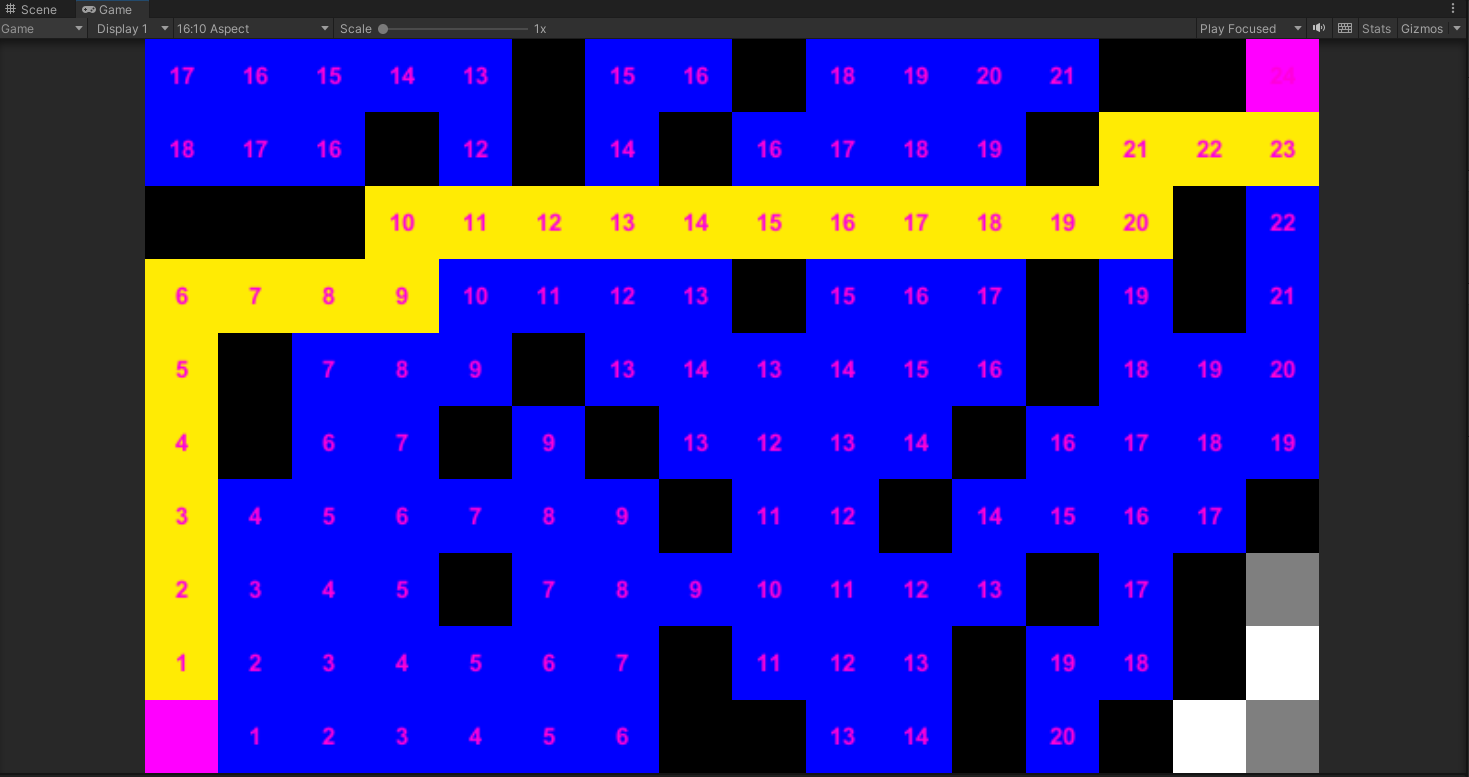
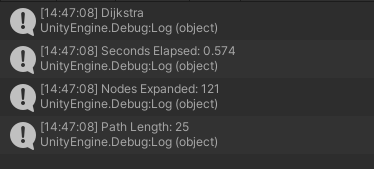
Dijkstra:

Scale: 1

Obstacle Probability: 0.25

Wait Time: 0





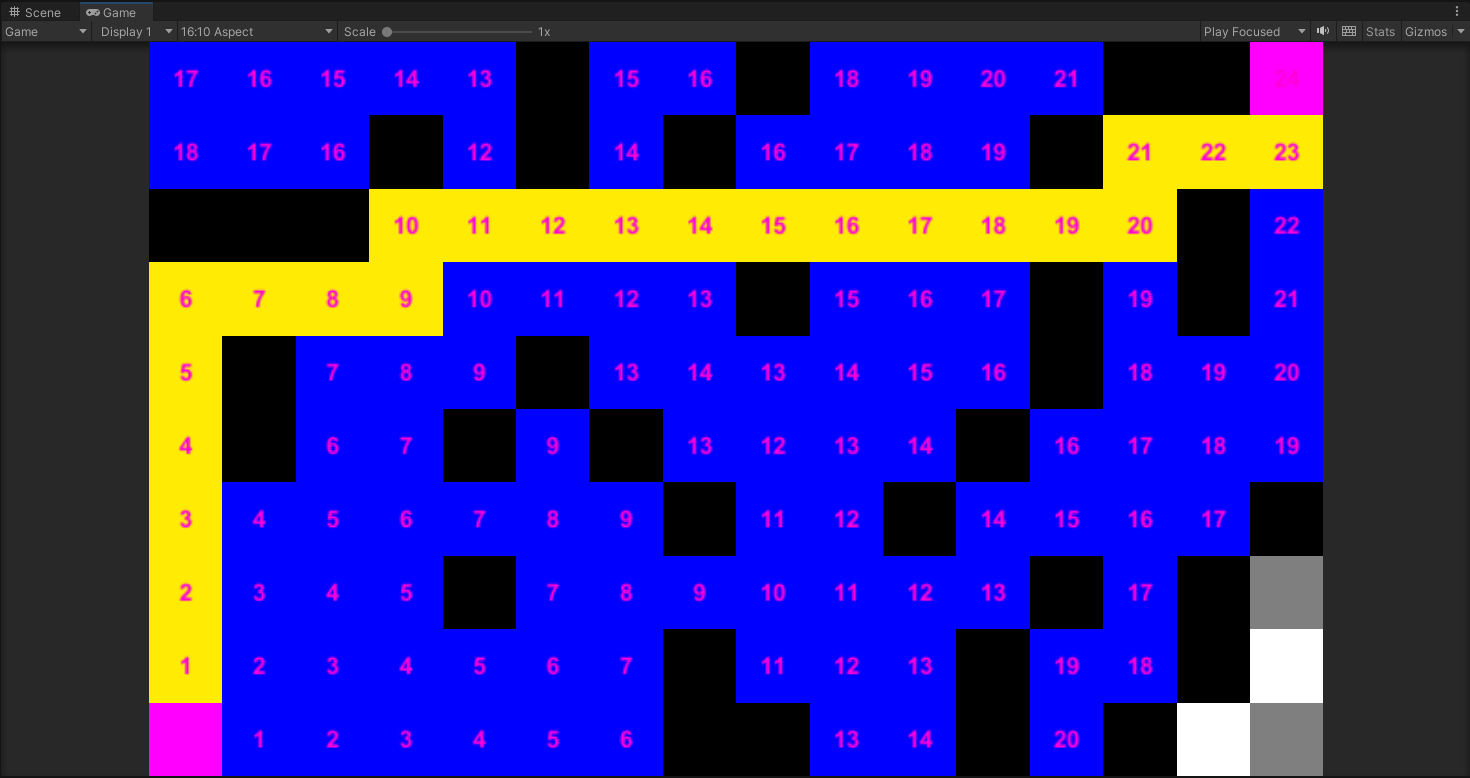
The slowest of all of the pathfinding algorithms used took 0.574 seconds to go through 121 nodes and find a path of length 25 tiles to the goal (including start tile and end tile)

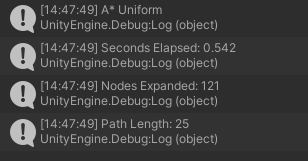
A\* Uniform:

Scale: 1

Obstacle Probability: 0.25

Wait Time: 0





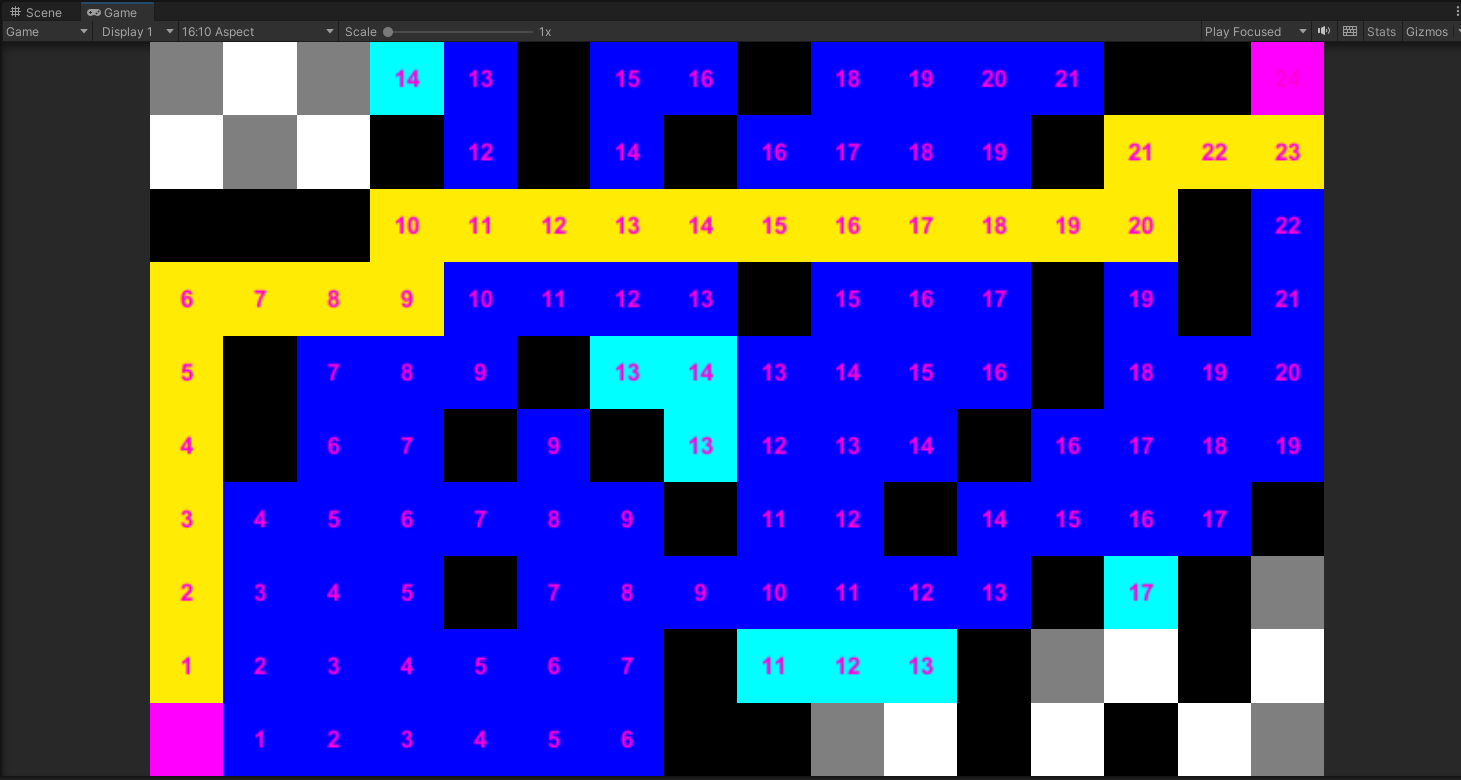
The second slowest of the pathfinding algorithms used, took 0.542 seconds to go through 121 nodes and find a path length of 25 (including start and end tile). Looks identical to Dijkstra because the weighted heuristic is 0 so the estimated cost is the same for all tiles.

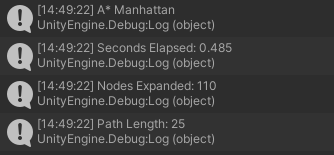
A\* Manhattan:

Scale: 1

Obstacle Probability: 0.25

Wait Time: 0





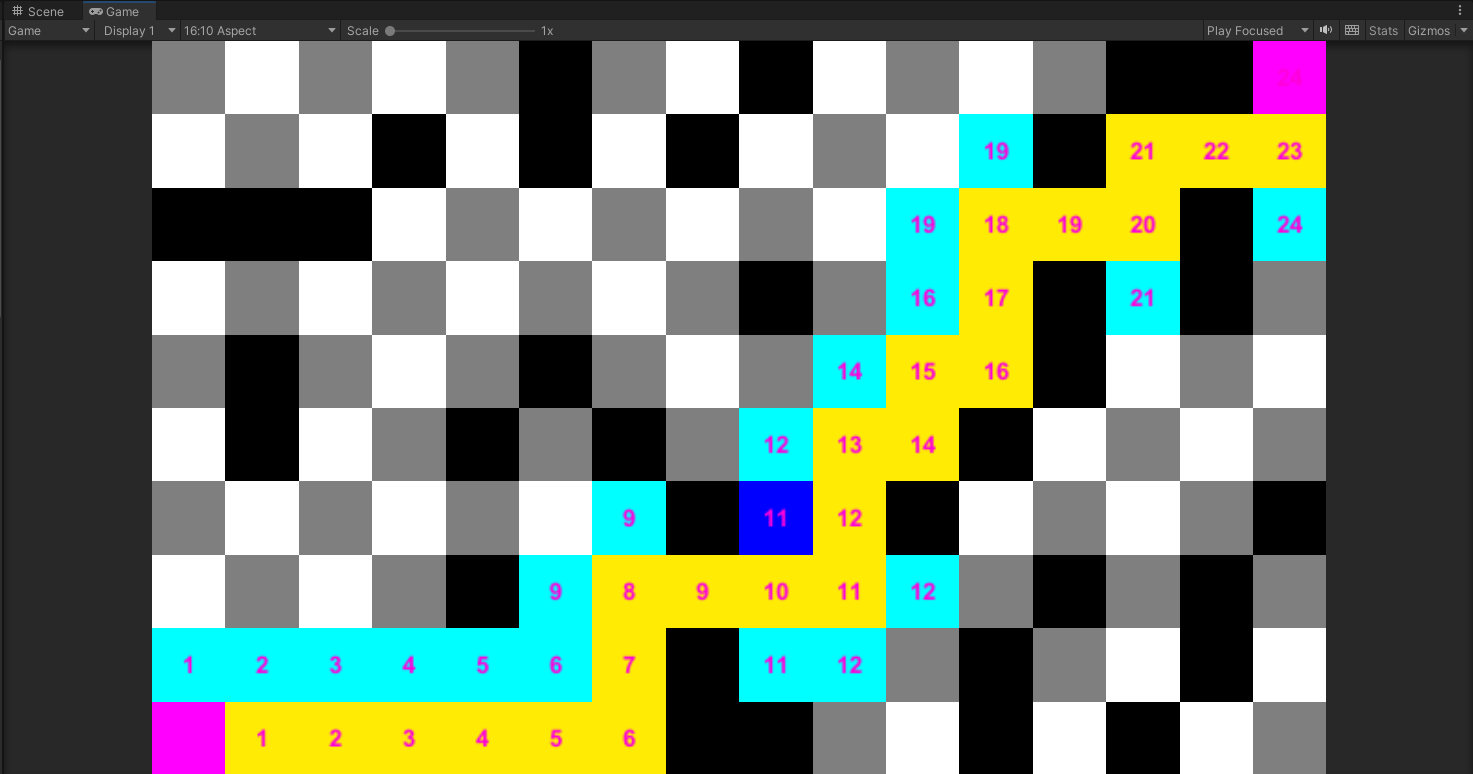
The second fastest pathfinding algorithm used took only 0.485 seconds expanding 110 nodes for a path 25 tiles long. It finds the same path that the other two found but didn’t check some nodes that were further away from the goal due to the heuristic.

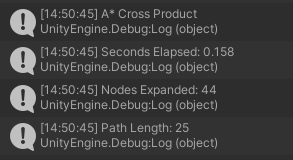
A\* Cross Product:

Scale: 1

Obstacle Probability: 0.25

Wait Time: 0





Finally the fastest path finding algorithm used was A\* Cross Product, it only took 0.158 seconds and only opened 44 nodes. The path was still 25 tiles long but took a much more direct route to the goal.

Dijkstra took the slowest out of the 4 algorithms as it checked every node around the start until it stumbled upon the end node. The A\* algorithm’s were all faster than Dijkstra, starting from uniform A\* which was almost identical to Dijkstra to Manhattan which chopped .1 seconds off by using a non-zero heuristic to A\* Cross Product which found the path in only 0.158 seconds by utilizing the cross product instead of just absolute values like Manhattan.