Amazon Coding Challenge

Salman Ashraf

-Phase 1

Using the A* search method:

$$F(n)=G(n) + H(n)$$

- $F(n)$ = total estimated cost of the path through node n
- $G(n)$ = estimated distance from starting node to current node
- $H(n)$ = estimated distance from current node to goal.

Vertical and horizontal distance in 1 unit. Using Pythagoras, the horizontal distance is √2.

Make an open list containing only the starting node

Make an empty closed list

While the destination has NOT been reached: {

consider the node with the lowest F value in open list

if this node is the destination:

return the path

else:

put the current node in the close list and look at all of its neighbours

for each neighbour of the current node:

if neighbour has lower G value than current and is in closed list:

replace the neighbour with the new, lower, G value

current node is now the neighbour's parent

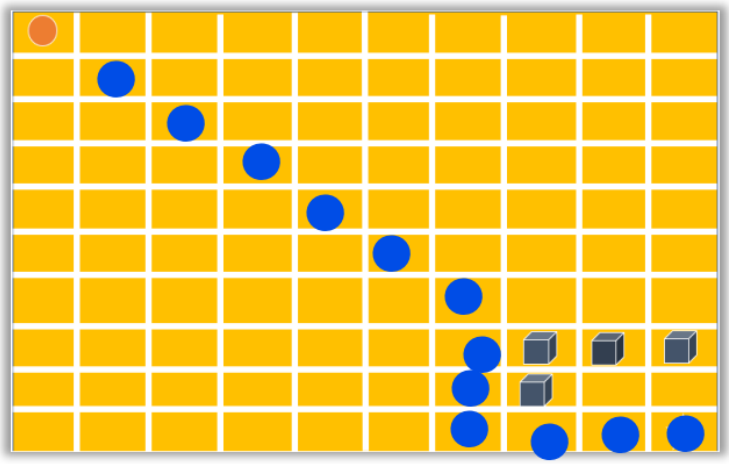
elif current g value is lower and this neighbour is in the open list:

replace the neighbour with the new, lower, G value

change the neighbour's parent to our current node

elif neighbour is NOT in any list:

add it to open list and set it to g



-Phase 2

The method used in phase one would work for any number of obstacles.

Make an open list containing only the starting node

Make an empty closed list

While the destination has NOT been reached: {

consider the node with the lowest F value in open list

if this node is the destination:

return the path

else:

put the current node in the close list and look at all of its neighbours

for each neighbour of the current node:

if neighbour has lower G value than current and is in closed list:

replace the neighbour with the new, lower, G value

current node is now the neighbour's parent

elif current g value is lower and this neighbour is in the open list:

replace the neighbour with the new, lower, G value

change the neighbour's parent to our current node

elif neighbour is NOT in any list:

add it to open list and set it to g

-Bonus:

If the open set is empty and the destination is not reached, then the problem cannot be solved.

Make an open list containing only the starting node

Make an empty closed list

While the destination has NOT been reached: {

consider the node with the lowest F value in open list

if this node is the destination:

return the path

else:

put the current node in the close list and look at all of its neighbours

for each neighbour of the current node:

if neighbour has lower G value than current and is in closed list:

replace the neighbour with the new, lower, G value

current node is now the neighbour's parent

elif current g value is lower and this neighbour is in the open list:

replace the neighbour with the new, lower, G value

change the neighbour's parent to our current node

elif neighbour is NOT in any list:

add it to open list and set it to g

print("Unable to reach delivery point")