Karim Elzanaty

Jeff Timanus

AI Programming

October, 2018

1. Overall Architecture

For the Pathfinding architecture I implemented A* and Djikstra's algorithms. For Dijkstra's, I have a findPath function that does all the work, a smallestNode function, a findNode function, and a contains functions. There is also a NodeRecord struct that contains extra information for each node, such as the cost so far, the connection, and the node itself. The findPath function creates an open and closed vector, that both hold NodeRecords. smallestNode finds the NodeRecord with the smallest costSoFar and returns it. findNode finds a NodeRecord that contains a specific node within one of the vectors, while contains just checks whether or not a vector even has a nodeRecord with a given node. The algorithm places nodes into the open list by getting the current nodes connections. It evaluates these connections first, then places them in the open list. Once all connections have been evaluated, the current node is placed into a closed list. Then, the node with the smallest "cost so far" is selected to be the current node, and the process continues. The findPath ends either once all nodes have been evaluated or the goal is found. At this point it, if the goal was found, it places the last node into a Path structure, and follows the nodes connection to place that node into the Path, and continues until it has reached the beginning again. Then it flips the path so it goes from beginning to end rather than the other direction. The A* uses the same structure, except smallestNode compares the estimatedTotalCost

instead of the costSoFar. A* also uses a heuristic that checks how far from the goal the current node is, regardless of obstacles. It also takes nodes off the closed list to update their values.

2. Challenges faced in development

Getting Djikstra off the ground was tough, as I ran into a big problem where it was just running DFS the whole time. Turns out, my inequality symbol was off and it was never checking if the cost was less than the costSoFar. A* also had a tough time actually performing faster than Dijkstra, because by solving the diagonal problem, the Estimated cost wasn't adding correctly. That is until I matched the heuristic return to the size of fix.

3. Areas where further improvements could be made

The A* could be improved. Using vectors slows it down a lot, so exchanging those for lists would be optimal. And there are a lot of local variables that could most likely become members for sake of cleanliness.