Cameron Belcher

Jeff Goldberg

A.I. for Games

24 October 2018

Assignment 3

This project was about pathfinding. I used the previous architecture of the game to make a path. There are three paths used in the project: depth-first search pathing, Dijkstra pathing, and A* pathing. Depth-first isn't in the same league as breadth first type of searches for path finding. This is because depth-first travels along the connections until one of the nodes happens to be connected to the end. A breadth-first kind of pathing checks all the nearby nodes before progressing to the next tier of nodes. That's what Dijkstra and A* path finding are. They use the breadth-first search algorithm to make a list of nodes that would connect to the goal node. If it doesn't then it returns a null pointer. They return the list by starting at the goal node, following back to the start node by accessing the from connection nodes, then adding the nodes to a path that holds the list of nodes. In this project the lists I used were vectors and they held node records, which contained the node, the connection of the node, and the cost of the node. When done you'll have to flip the path because it starts at the end. Now there is a difference between A* and Dijkstra path findings. Although they both use a breadth-first search to find their end goal, A* uses a heuristic to make the path finding more efficient. Using the heuristic finds the least costly path to the goal. In my project, I used distance to get the cost. Therefore, the closer the node was to the goal, the more likely it was to be prioritized.

There were some small challenges I had with this project. One of which was choosing a heuristic. There are so many heuristics out there to implement, but after some consideration, I thought it’d be best to do a heuristic that wasn’t too complex. This way it wouldn’t take much time to get the heuristic. Another problem I faced were tiny user problems. I would forget to delete the path. I would use an else if instead of just an if. I guess they weren’t project problems as much as they were user errors, but it took time debugging none the less.

Areas that could be further improved could be the heuristic. There are more complex heuristics out there that could fasten the time but were more complex and would take more time for me to implement. I also could also have used a priority queue instead of vectors. From other fellow programmers’ experiences the paths are much faster when implementing the queues.