Kelly Herstine

Jeff Timanus

AI Programming

October 23, 2018

Assignment 3

Overall Architecture of your Assignment:

All Pathfinding behaviors are inside of their own classes. These classes include: Astar and Dijkstra. Astar consists of a constructor and destructor, a NodeRecord struct which consists of different identifiers, a getSmallestElement function that returns a Node Record, a findPath function that gets the path for the Nodes, and a getHeuristic function that gets the estimated distance between a two different Nodes. The NodeRecord struct is used for comparing different Nodes within the Open and Closed lists inside of the class. The NodeRecord Struct also consists of a Node, a Connection, a CostSoFar, and an EstimatedTotalCost. Dijkstra also uses this struct as well. Dijkstra consists of a NodeRecord struct and a findPath function that gets the path for the Nodes. I also changed the architecture of my InputSystem for this project. I was originally using the Input system that I have had since Game Architecture of last year but I needed to change it due to the fact that it wouldn’t work with the architecture set up for Pathfinding. The major difference is that I used the GameMessage class to send messages throughout the system. In the InputSystem, I get the inputs from the keyboard and then add a message to the GameMessage class sending it to its corresponding classes. The four classes I have for messaging in the InputSystem are AStarModeMessage, DFSModeMessage, DijkstraModeMessage, and ExitMessage. All these classes either switch the game mode to their own mode or they exit the program. This is then processed in the GameApp class and switched when it is needed.

Challenges Face in Development:

I really struggled with trying to understand how pathfinding worked. The hardest part that I had was getting Dijkstra down. I needed to make sure that my Dijkstra’s architecture was solid before I went to A\* since A\* is an optimization of Dijkstra. It took me a while to try to understand the pseudo code of the book and translate that into what I wanted. Once I had Dijkstra down, my next objective was trying to get A\* to work. This also took a bunch of debugging and attempting multiple algorithms.

Areas where further improvements could be made:

I think that I could still definitely improve on how both of my Pathfinding classes look. There are a bunch of for loops that could be moved to functions and made to look nicer. I was originally afraid to start like this because I was afraid that moving things to functions might cause the program to break. If I had more time, I would definitely spend more time on making some of the Remove, Add, and Find loops into functions which would clean the code up a lot.