Andrew Rimpici

Jeff Timanus

EGP-410-101

10/22/2018

Assignment Three Overview

Overall Architecture

In my program, I separated the three algorithms into their own classes. They each extend from GridPathfinder. The GameApp class holds a GridPathfinder pointer that can be swapped out to the different subclass types and allows for the path algorithm to change. My input system uses the same architecture that my assignment two uses, and now supports PathToMessage that handles redrawing the path.

The NodeRecord struct is responsible for holding the Node, Connection, totalCost, and estimatedTotalCost of a certain path. The NodeRecord in combination with the PathfinderList class helps simplify the Dijkstra and A* code implementation as I was able to abstract away helper functions. My Pathfinder structure currently uses a list as it's underlying data structure to hold all of the NodeRecords. A list structure allows for faster insertion and deletion of NodeRecords than if I were to use a vector.

Challenges Faced In Development

Originally, I was dynamically allocating memory for NodeRecords* and inserting them into my PathfinderList that way. I was having trouble deleting the NodeRecords* from the list as it would

invalidate my iterator. I eventually switched to regular NodeRecords and it fixed the invalidation issue and took away the added worry of tracking NodeRecord* memory.

Areas Of Improvement

I think the PathfinderList data structure could be greatly optimized by either converting it to use a priority heap or bucketed priority queue like the book suggests. I am curious to see how these data structures would speed up the algorithms if I had more time to implement them.