Assignment 3

My architecture is based on the std priority_queue and the provided depth-first search pathfinding code. My architecture takes the same principle of breadth-first search and adds a weight parameter to the nodes, as well as a back-pointer stored as the node's ID. For Dijkstra, each node that is explored is checked against the closed list, and if it is open, is given a weight based on the connection cost and previous weights, and a backpointer to the previous node. A* works the same way, but a heuristic is calculated for the distance between the current node and the upper left-hand corner of the target node as the distance between two vector2D's, and that heuristic is added to the node's weight. This results in A* taking a much more directed and linear path towards the target node. To draw the final path, I used a Path (the same container the closed list was in) and I go back over the target node through each backpointer and add all of the nodes to the list. Then, once the node that is being checked is the same as the start node, the path is drawn.

My input system is the same one I used for the steering projects, but somewhat modified for the new architecture. I spent about 3 hours trying to figure out why the variables I made public in the GameApp class weren't visible under gpGame, which is instantiated as a GameApp class. I then looked in the moveTo function to find that I could have saved so much time by simply using a dynamic cast.

One of the biggest problems I had with this project was getting the path to draw after the pathfinding was done. In one case the list would iterate too far back sometimes and throw a null pointer error when my checks for when to exit the loop weren't as precise. Another problem I would have is that the mouse sometimes registers 2 clicks or a drag as a new path of length 0 and when the index to color the start node was called, it would throw an index out of bounds error. I was able to fix it by encapsulating the area that threw the error in an if statement that ignored paths of length 0, but ideally I would have some sort of limit on how quickly the input manager updated the input from a mouse click.