**Pathfinders**

class DijkstraPathfinder – Handles pathfinding for the dijkstra algorithm

* struct DijkstraNode – Holds dijksta related data for nodes (connection & cost)
  + bool operator<(const DijkstraNode &\_toCompare) const – Compares cost of 2 DijkstraNode objects
* DijkstraNode getNodeInOpenList(Node\* node, std::priority\_queue<DijkstraNode> openList) – Returns the DijkstraNode that contains the passed in node in the open list. If the node isn’t found returns a default constructed DijkstraNode

class AStarPathfinder – Handles pathfinding for the A\* algorithm

* struct AStarNode – Holds A\* related data for nodes (connections, cost & estimated total cost)
  + bool operator<(const AStarNode &\_toCompare) const – Compares estimated cost of 2 objects
  + bool operator==(const Node\* &\_node) const – Compares the node of the AStarNode node to the Node that was passed in
  + bool operator==(const AStarNode &\_node) const - Compares the nodes of 2 AStarNode objects
* AStarNode getNodeInOpenList(Node\* node, std::priority\_queue<AStarNode> openList) – Same implementation of the DijkstraNode version of this function
* float getHeuristic(AStarNode \_node, Node\* \_goal) – Returns the heuristic calculated for the A\* algorithm

**Messages**

class ChangePathfinderMessage – Changes the pathfinding algorithm to the one specified in the constructor

class StartPathfindMessage – Starts calculations for a new path from the goal of the last path to the position specified

class QuitGameMessage – Quits the game app