## **CLASS DIAGRAM**

## **MIDTERM PROJECT**

## VisualizeMapSDL **AStar** - width: int = 640- screenWidth : int = 640 - height: int = 480- screenHeight : int = 480 - xStart : int = 10- closed : bool = false - yStart : int = 10- title : string = "Environment" -xGoal: int = 620- delay\_: **int = 1000** -yGoal: int = 460- renderer\_: SDL\_Renderer = nullptr - mapFileName\_: string - window : SDL window = nullptr - occupancyMatrix\_: pair<int, int> VisualizeMapSDL() - occupancyMatrix : vector<vector<int>> VisualizeMapSDL(const std::string &, int, int) - openSet : list<Nodes> ~VisualizeMapSDL() - VisualizeMapSDL visualize - init(): bool + isClosed(): bool AStar() **AStar(const std::string)** + clear(): void **AStar(const std::string &, int, int)** + detectEvent(): void ~AStar() + callDelay(): void + generateMap(): void + drawPixel(int x, int y, int color): void + computePath(): void + updateWindow(): void + sort(const Nodes &, const Nodes &): bool + setDelay(int): void + checkNodeID(): bool **Nodes** # nodeID : bool = false # parentID\_: int # totalCost : double # costToCome\_: double # heuristicCost : double # xPose : int # yPose : int Nodes() ~Nodes() + computeHeuristic(): void + computeCostToCome(): void + computeTotalCost():void + checkNode(): void