AI

A* Pathfinding

Divide the area that is walkable into cells/nodes. The start position is inside a cell and the goal position is inside another cell.
The idea is to travel from the start to the end node taking the shortest accessible path. Nodes that are not accessible because of collision are marked as that. The algorithm calculates the cost for moving between one node to another in relation to the goal node. It chooses a cell from the cells around the current cell, then it chooses a new cell until enough cells have been found for a path to be built to the goal.

- **g :** the cost of moving from the initial cell to the current cell. Basically, it is the sum of all the cells that have been visited since leaving the first cell.
- **h :** also known as the *heuristic value*, it is the **estimated** cost of moving from the current cell to the final cell. The actual cost cannot be calculated until the final cell is reached. Hence, h is the estimated cost. We **must** make sure that there is **never** an over estimation of the cost.
- **f :** it is the sum of g and h. So, **f** = **g** + **h**

The f value is constantly being updated and the minimum f value is the one that is chosen.
Use two lists of nodes. An open and a closed.
Add the correct node into the open list
Make a while loop while the open list is not empty and add the start node into it before you enter the loop
remove the node with the least f value from the open list.
for every neighbor node, calculate f.
choose one of these, if one of them is the goal node then end.
skip the node if a better one or equal one is already in the open or closed list. Otherwise add it to the open list.
Put the previous correct node into the closed list.
Use the closed list to get the path when the final node is found.

Optimizations

Quad Tree

To quickly find the correct index of whatever Astar Node/Cell the enemy is in.
A Quad is created around the entire grid.
This quad is divided in 4, and all of those are also divided by 4 recursively until the leaf quads are smaller than the AI cells or the same size.

Instead of looping through the AI cells when trying to find the one matching the world position of the enemy accessing it, the world position is checked with the relevant bounds of the quad node until a leaf node is found and the ai cell this leaf node is inside is returned.