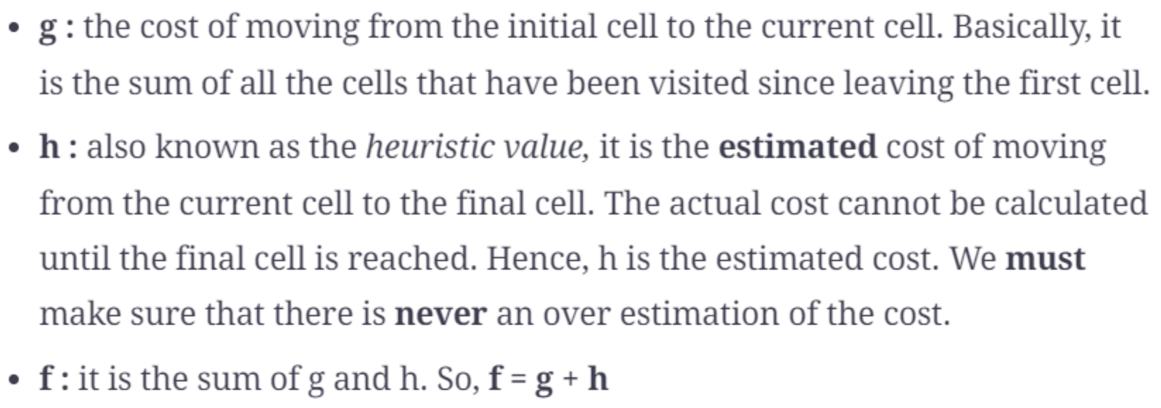
AI

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.

Flocking Algorithm

Each object considered an AI agent part of the same flock, updates every frame with different behaviors. When they move the velocity is created by taking these behaviors into consideration. One behavior function in my implementation is for the flock to move towards a target.
They also have functions to keep themselves separated from each other, separated from static objects in the world and functions to be cohesive and aligned with each other.

For them to know about collisions, I use the same grid as with the Astar for them to know areas they should stay away from before they need to update in real time.
For them to stay away from each other, every unity checks in real time what they collide with within a certain radius.
If these objects are other agents they will add them in a list of “neighbors”. Then this list is used as constraints for how to update the velocity in the behavior functions.
To promote one's behavior more than other weights are used. Each velocity from each function is scaled with a weight and clamped to not go above a threshold.
Some velocities are also interpolated to avoid jittering.
To further avoid jittering they wont move in a direction where there is an obstacle ahead of them.
And if they separate from static collisions they won't try to move the same direction again.

Decision Tree

I made a finite state machine first but it became too complicated.
There is one Decision Tree per AI component.
Every tree has one ambiguous dictionary with data for every node to find.
When traversing nodes, the tree either selects between a different sets of nodes, or chooses a node based on multiple child nodes all returning true.
Some nodes are leaves they have no children, these nodes uses the data to manipulate the ai agent they are attached to.

Different Nodes are created for different functionalities.