Implementation of A* Search algorithm

What is the A* Search algorithm?

The A* search algorithm is a type of best-first search algorithm which means that it considers all possible paths to the destination to look for the path that has the least cost. Cost could mean distance travelled, time, or weight of nodes. The A* search algorithm is widely used for pathfinding and graph traversals because of its efficiency in finding the least cost.

Analysis

Time complexity of the A* search engine is exponential since there is finite space to be searched and the goal/destination is reachable from the origin. Each corner will only be visited, at most, once.

The difference between A* and other greedy best-first search algorithms is that it takes the cost/distance traveled into account in each iteration.

A* search was originally designed as a graph-traversal algorithm but is now commonly used for pathfinding in games. Starcraft and many other RTS games use A* for their AI pathfinding.

Implementation

Unity3D is the game engine used for this implementation of the A* Search algorithm. It is used as a pathfinding mechanism for units to avoid obstructions and finding the shortest path to the destination.

There are 4 scripts: PathEntity, PathingMap, PathingQuery, Moveaction

PathEntity is a script that is attached to an object for it to be seen as an obstruction. Adds costs to the tiles that the object occupies.

PathingMap is a script that divides the map into several chunks/tiles and assigns costs to those tiles. This is the script that assigns costs to objects that are attached with the PathEntity script.

PathingQuery is a script that does the A* implementation. On initialization, the algorithm applies a cost to the four corners of the object. (1,0) (0,1) (-1,0) and (0,-1) are given a cost of 1. Tiles that already have a cost of 1(PathEntities) are not considered since they are unpassable. By estimating the cost to the destination from each corner, the corner with the minimal cost will be the chosen direction. The cost is then calculated by (from_cost + 1). Costs are continually added until the target is reached.

Limitations

It cannot recognize terrains that are walkable, but at a higher cost, such as stairs or higher slopes.

Unused space of objects that are larger or less than 1 unit are considered as occupied since the tile itself is considered occupied.

Units specified as PathEntities are not able to move therefore you cannot assign the script to player controllable objects.

Illustrations