Pathfinding

How pathfinding works

Finding the path, or the sequence of steps, from A to B might not be as easy for an AI Agent as it is for a human. Luckily, we have pathfinding to save the day.

Pathfinding is the method of searching the shortest path from a point to another. At the core, pathfinding searches a graph (see the bonus section, data structures) starting from one of its nodes and, from neighbor to neighbor.

An algorithm that appears everywhere in gamedev is A*. In real-life use cases, this algorithm is mostly hidden behind layers upon layers of other systems that make use of it.

And it should be simple - because otherwise we will be reinventing the wheel over and over again. The most common issue here is that while modern game engines do provide a simplified process to it, it removes the proper understanding regarding this system.

Why is this a possible concern for a game developer? Because games usually need special cases like custom maps with hexagons, unique features and more. And just by using the tools at hand that task might be close to impossible.

Behind the pathfinder

At the core of any pathfinding solution, there are two main concepts:

DOMAIN + ALGORITHM

The Domain - Where it happens

Don't get tricked when thinking that the space means the terrain heightmap or the plane where the AI Agent needs to find its way. The space is the representation of that space in a way that is:

- **Fast** - having the full environment for running the algorithm over and over again is not really doable in a game project. We need a simplified version of it, similar to a High-Poly 3D Object's Low-Poly counterpart.
- **Reliable** - being fast but inaccurate is not ideal as well. Having AIs running through walls and other obstacles clearly diminishes the player's game experience.

- **Stored properly** - if we write the simplified version on a piece of paper it won't help our CPU process it. We need to use a proper data structure to hold all of this newly generated information.

The Algorithm - How it happens

There are many algorithms capable of doing pathfinding. Most of them work very similar to one another. The main difference is their implementation complexity and their CPU usage.

They can be categorized in two types:
- **UNINFORMED** -> does not know details about the map, deals with the situation as it unfolds (when an obstacle is encountered)
- **INFORMED** -> has additional information and does something extra for each search step

When speaking of pathfinding A* is the industry-standard because it produces fast results.

A* is an algorithm that belongs to the **INFORMED** type. That means, in order to properly run, A* does require more information about the map than just the obstacles.

In the next chapter let's look at a pathfinding algorithm, how it works and how it's implemented.