The path to the right decision: An investigation into using heuristic pathfinding algorithms for decision making

Ashley Smith

November 24, 2019

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras justo velit, vestibulum sit amet turpis in, interdum rhoncus magna. Proin pulvinar posuere iaculis. Duis vulputate tristique arcu, id pretium ante blandit ut. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nam augue tellus, mattis quis consequat id, facilisis eu lectus. Vivamus euismod non quam sed condimentum. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Phasellus vitae consequat nisi. Morbi vulputate tellus ut nibh vulputate, vitae blandit ex faucibus.

1 Introduction

As video games have evolved, so too have strategies used to design and implement artificial intelligence (AI) into the characters and logic necessary for the player's enjoyment. The requirements for game AI are not the same as academic AI; the characters involved simply need to interact with the environment in a manner which makes the game fun to play. One common requirement for game AI is for the characters to be able to traverse the areas of the game in a way which meets the player's expectations logically and efficiently - a task known as pathfinding.

Typically, the pathfinding side of AI is separate from the decision making side. Some algorithm will perceive the world and come to a conclusion about what to do and then pass this information on to the pathfinding algorithm in charge of navigating to the desired area. This means that the pathfinding algorithm doesn't do any 'thinking' and instead just generates a path from one point to another. The path is generated after a decision has already been made, and so any information gathered that could otherwise be useful to the decision-maker is lost.

Without knowing the path or any specific details about points of interests along said path beforehand, a decision-making process could easily make mistakes that could be considered wrong or 'glitchy' by both players and developers. Similarly, factoring in environmental details like these into the decision runs the risk of overcomplicating the process and making it harder to manage and maintain, or repeating calculations made in the following pathfinding process.

In this paper, the mechanisms of a typical pathfinding algorithm are examined and reengineered, through the substitution of input and output types, with the aim of bringing decision-making and pathfinding closer together.

2 Literature Review

Games are good for the economy. Graphics used to be the sales feature of games (Grid based pathfinding). Now, AI is more of a focus (AI in computer games) (Human-level AI's killer application: Interactive computer games). Its not just about making enemies.

Decision trees are the simplest way to implement AI but are basic.

FSMs are one of the most common ways of implementing AI (Three states and a plan: the AI of FEAR (228 cites), but they also don't provide a simple way of combining these tests together to make more complex queries.

Behaviour trees are an improvement over FSMs.

Current techniques do not scale up (Human-level AI's killer application: Interactive computer games)

A solution is an action sequence, so search algorithms work by considering various possible action sequences (Artificial Intelligence: A modern approach). These pathfinding algorithms can search for a solution for traveling from A to B, but could potentially do more with the redefinition of what an action is. In theory, you could apply 'best first' to game interactions in a scenario.

Normally, pathfinding takes a backseat when it comes to decision making and just generates the requested path. However, since pathfinding algorithms are just search algorithms being applied to graphs, there's no restriction that these nodes and edges have to correspond to locations and distances.

Dijkstra's algorithm is a simple pathfinding algorithm that finds the shortest route to every node in the graph. It's an uninformed algorithm.

AStar is the defacto algorithm for pathfinding. Its actually a family of algorithms where the heuristic dictates how it functions. AStar with no heuristic is dijkstras, and could also be used to become a best-first, breadth first and depth-first.

Changing how A* evaluates actions and applies a heuristic will allow it to operate differently. A good heuristic will allow the algorithm to discard 'bad moves' (Depth-first iterative-deepening: an optimal admissable tree search).

A* becomes inefficient on bigger maps and lots of techniques have been created to adjust to these problems. Having lots of edges per node could make A* very inefficient.

This paper will investigate the effects of different cost and heuristic functions when changing the types involved with A* to see how agents perform.