Assessment for ICTGAM423, ICTGAM533
Simon Vannarath

Assignment 1: Tower Defence (Tasks 3 and 4)

Describe Dijkstra and A* algorithms, what are the difference between the two?

Dijkstra's algorithm finds the shortest distance from a start node to target node with the map represented as a weighted graph. The algorithm keeps track of the currently known shortest distance from each node to the source node and it updates these values if it finds a shorter path. Once the algorithm has found the shortest path between the source node and another node, that node is marked as "visited" and added to the path. The process continues until all the nodes in the graph have been added to the path. A* operates in a similar way but uses a heuristic function to optimise the search instead of Dijkstra's approach which will explore all possible paths until reaching the target.

Which game genres are you likely to use A* with and why?

Real-time strategy, RPGs; many games in these genres rely on the player to move characters indirectly e.g. mouse click on a traversable location. The game will move the player characters according to a pathfinding algorithm to that location. The opposition AI will require more complex pathfinding to generally seek to the player characters via the shortest path. Pathfinding using an algorithm like A* potentially produces a realistic response to navigation in a shorter timeframe by modifying the heuristics.

Which game genres would you NOT use A* with and why?

Fighting games, sport games, tower defence; fighting games require direct user input, sport games generally have simpler pathfinding (most sports are played on a flat field or court) and many tower defence games use a single path and do not require pathfinding.

What limitations do path finding algorithms put on game design and development.

Pathfinding algorithms have the potential to impact on memory usage as the map scales up. Larger maps will require more waypoints/nodes to navigate longer distances. To calculate such routes requires more memory to load the nodes and processing power to process each node. Simpler pathfinding may reduce overheads for memory and processing constrained platforms but may lead to movement that may appear "too artificial" where realistic movement is required.

List 4 algorithms and/or paradigms do AI use in games.

- Bellman–Ford algorithm
- Dijkstra's algorithm
- A* search algorithm
- Breadth-first search

Reflection

There could have been many things that could be improved (especially if I allocated more time then what I ended up giving it) but if implemented as a commercial product the AI would require more code and by extension more memory (additional data dictionary structures) impacting performance if space and memory constraints are issues for the target platform. Timeline wise it's core functionality and therefore critical; other systems depend on the pathfinding/movement of units on the field. In terms of technical feasibility, its use in a tower defence context is close to critically important in allowing for the units to traverse multiple paths toward their objectives, especially if there are obstacles that will hinder their progress. Using Dijkstra's or A* algorithms in the pathfinding implementation impacts user suitability; it provides a more realistic response to changes in the environment e.g. pathfinding updates due to obstacles placed on paths. This allows for a more engaging experience for a player against a 'smart' computer-controlled opponent.