Definition document

20.9.2013

Game AI using the A* algorithm. Making an implementation of the algorithm in Java to traverse an arbitrary grid (the vertices in a mesh etc).

I've found these resources quite useful when working with this algorithm.

A*

http://theory.stanford.edu/~amitp/GameProgramming/

http://www.policyalmanac.org/games/aStarTutorial.htm

Heuristics:

http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

PriorityQueue

http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html

ArrayList

No source. Simple implementation with an Object array.

A*

The algorithm chooses the next nodes based on the total score of the travelled distance and the remaining distance to the goal, calculated with a heuristics method. The score is then sorted so the method gets the lowest possible scores out of the list all the time. The default run operation uses my own implementation of an ArrayList and a PriorityQueue.

The speed and effectiveness of the algorithm is strongly related to the heuristic method chosen. With a better heuristic method one will get the shortest path to the goal, but this will also take up the most resources to calculate.

ArrayList

The ArrayList's implementation builds on an array that doubles in size when the maximum size is reached. (It's very poorly implemented for now, just the methods needed in this project are used.)

	Add	Delete	Get	Find		
Time:	O(1)	O(n)	O(1)	O(n)		
Space:	O(n)					

PriorityQueue

The priority queue is much similar to the ArrayList, with a array that doubles in size when the limit is reached.

The structure used is a binary heap (every node has at most two children) and the heap can be defined to be a max or min heap based on the Comparator provided during construction.

	Add	Delete	Get	Find	Poll	Peek
Time:	O(log(n))	O(log(n))	O(1)	N/A	O(log(n))	O(1)
Space:	O(n)					