Summary

"BallPath" project uses A* search algorithm for pathfinding. This algorithm make it possible to search optimal (shortest) path. A* is *complete* and will always find a solution if one exists.

Algorithm description

A* uses a best-first search and finds a least-cost path from a given initial node to one goal node (out of one or more possible goals).

It uses a distance-plus-cost heuristic function (usually denoted $f(x)$) to determine the order in which the search visits nodes in the tree. The distance-plus-cost heuristic is a sum of two functions:

- the path-cost function, which is the cost from the starting node to the current node (usually denoted $g(x)$)
- and an admissible "heuristic estimate" of the distance to the goal (usually denoted $h(x)$).

The $h(x)$ part of the $f(x)$ function must be an admissible heuristic; that is, it must not overestimate the distance to the goal. Thus, for an application like routing, $h(x)$ might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points or nodes.

If the heuristic h satisfies the additional condition $h(x) \leq d(x, y) + h(y)$ for every edge x, y of the graph (where d denotes the length of that edge), then h is called monotone, or consistent. In such a case, A* can be implemented more efficiently—roughly speaking, no node needs to be processed more than once (see *closed set* below)—and A* is equivalent to running Dijkstra's algorithm with the reduced cost $d'(x,y) := d(x,y) - h(x) + h(y)$.

Algorithm complexity (speed of search and worst case)

The time complexity of A* depends on the heuristic. In the worst case, the number of nodes expanded is exponential in the length of the solution (the shortest path), but it is polynomial when the search space is a tree, there is a single goal state, and the heuristic function h meets the following condition:

$$|h(x) - h^*(x)| = O(\log h^*(x)) \quad ,$$

where h^* is the optimal heuristic, the exact cost to get from x to the goal. In other words, the error of h will not grow faster than the logarithm of the "perfect heuristic" h^* that returns the true distance from x to the goal.

Implementation notes

As heuristic cost estimate function used Manhattan length: $l_M = |x_1 - x_2| + |y_1 - y_2|$.

To prevent a premature pessimisation BallPath implementation of A* uses next data structures (see `src/core/pathfinder.h`):

- for *open list*: `std::priority_queue`.

  Rationale: fast retrieving of minimal element; convenient usage for open list.

- for *closed list*: `std::unordered_set` (C++11).

  Rationale: fast element pushing and finding; convenient usage for closed list.