Algorithms and Networking for Computer Games

Chapter 5: Path Finding

http://www.wiley.com/go/smed

Path finding

- common problem in computer games
  - routing characters, troops etc.
- computationally intensive problem
  - complex game worlds
  - high number of entities
  - dynamically changing environments
  - real-time response

Problem statement

- given a start point *s* and a goal point *r*, find a path from *s* to *r* minimizing a given criterion
- search problem formulation
  - find a path that minimizes the cost
- optimization problem formulation
  - minimize cost subject to the constraint of the path

The three phases of path finding

1. discretize the game world
   - select the waypoints and connections
2. solve the path finding problem in a graph
   - let waypoints = vertices, connections = edges, costs = weights
   - find a minimum path in the graph
3. realize the movement in the game world
   - aesthetic concerns
   - user-interface concerns

Discretization

- waypoints (vertices)
  - doorways, corners, obstacles, tunnels, passages, …
- connections (edges)
  - based on the game world geometry, are two waypoints connected
- costs (weights)
  - distance, environment type, difference in altitude, …
- manual or automatic process?
  - grids, navigation meshes

Grid

- regular tiling of polygons
  - square grid
  - triangular grid
  - hexagonal grid
- tile = waypoint
- tile’s neighbourhood = connections

Navigation mesh

- convex partitioning of the game world geometry
  - convex polygons covering the game world
  - adjacent polygons share only two points and one edge
  - no overlapping
- polygon = waypoint
  - middlepoints, centre of edges
- adjacent polygons = connections

Solving the convex partitioning problem

- minimize the number of polygons
  - points: n
  - points with concave interior angle (notches): $r \leq n - 3$
- optimal solution
  - dynamic programming: $O(r^2 n \log n)$
- Hertel–Mehlhorn heuristic
  - number of polygons $\leq 4 \times$ optimum
  - running time: $O(n + r \log r)$
  - requires triangulation
    - running time: $O(n)$ (at least in theory)
    - Seidel's algorithm: $\mathrm{O}(n \lg^* n)$ (also in practice)

Path finding in a graph

- after discretization form a graph $G = (V, E)$
  - waypoints = vertices (V)
  - connections = edges (E)
  - costs = weights of edges ($\mathit{weight} : E \to \mathbf{R}_+$)
- next, find a path in the graph

Graph algorithms

- breadth-first search
  - running time: $O(|V| + |E|)$
- depth-first search
  - running time: $\Theta(|V| + |E|)$
- Dijkstra's algorithm
  - running time: $O(|V|^2)$
  - can be improved to $O(|V| \log |V| + |E|)$

Heuristical improvements

- best-first search
  - order the vertices in the neighbourhood according to a heuristic estimate of their closeness to the goal
  - returns optimal solution
- beam search
  - order the vertices but expand only the most promising candidates
  - can return suboptimal solution

Evaluation function

- expand vertex minimizing

$$f(v) = g(s \sim> v) + h(v \sim> r)$$

- $g(s \sim> v)$ estimates the minimum cost from the start vertex to v
- $h(v \sim> r)$ estimates (heuristically) the cost from v to the goal vertex
- if we had exact evaluation function f^*, we could solve the problem without expanding any unnecessary vertices

Cost function *g*

- actual cost from s to v along the cheapest path found so far
  - exact cost if G is a tree
  - can never underestimate the cost if G is a general graph
- $f(v) = g(s \sim\!> v)$ and unit cost $\longrightarrow$ breadth-first search
- $f(v) = -g(s \sim\!> v)$ and unit cost $\longrightarrow$ depth-first search

Heuristic function *h*

- carries information from outside the graph
- defined for the problem domain
- the closer to the actual cost, the less superfluous vertices are expanded
- $f(v) = g(s \sim> v) \rightarrow$ cheapest-first search
- $f(v) = h(v \sim> r) \rightarrow$ best-first search

Admissibility

- let Algorithm A be a best-first search using the evaluation function f
- search algorithm is *admissible* if it finds the minimal path (if it exists)
  - if $f = f^*$, Algorithm A is admissible
- Algorithm A* = Algorithm A using an estimate function h
  - A* is admissible, if h does not overestimate the actual cost

Monotonicity

- h is locally admissible → h is monotonic
- monotonic heuristic is also admissible
- actual cost is never less than the heuristic cost → f will never decrease
- monotonicity → A* finds the shortest path to any vertex the first time it is expanded
  - if a vertex is rediscovered, path will not be shorter
  - simplifies implementation

Optimality

- Optimality theorem: The first path from *s* to *r* found by A* is optimal.
- Proof: see page 105 of the book

Informedness

- the more closely h approximates h^*, the better A* performs
- if A_1 using h_1 will never expand a vertex that is not also expanded by A_2 using h_2, A_1 is more informed that A_2
- informedness → no other search strategy with *the same amount of outside knowledge* can do less work than A* and be sure of finding the optimal solution

Algorithm A*

- because of monotonicity
  - all weights must be positive
  - closed list can be omitted
- the path is constructed from the mapping π starting from the goal vertex
  - $s \to \ldots \to \pi(\pi(\pi(r))) \to \pi(\pi(r)) \to \pi(r) \to r$

Practical considerations

- computing *h*
  - despite the extra vertices expanded, less informed *h* may yield computationally less intensive implementation
- suboptimal solutions
  - by allowing overestimation A* becomes inadmissible, but the results may be good enough for practical purposes

Realizing the movement

- movement through the waypoints
  - unrealistic: does not follow the game world geometry
  - aesthetically displeasing: straight lines and sharp turns
- improvements
  - line-of-sight testing
  - obstacle avoidance
- combining path finding to user-interface
  - real-time response

Alternatives?

- Although this is the *de facto* approach in (commercial) computer games, are there alternatives?
- possible answers
  - AI processors (unrealistic?)
  - robotics: reactive agents (unintelligent?)
  - analytical approaches (inaccessible?)