Pathfinding for Grid and NavMesh

Author: Jiaming Liu 301626723

Introduction

I built an interactive pathfinding demo to compare grid-based search and a simplified NavMesh approach. Users place obstacles (grid mode), set start/goal, pick algorithms, and watch agents animate. Metrics show speed, nodes explored, and path cost. The goal is to show how different planners trade off optimality, speed, and smoothness.

Demo: https://766.jiamingliu.com/
Repo: https://github.com/Jiaaming/Pathfinding

Related Work

- A* Search: classic optimal graph search with heuristics.
- Dijkstra: uniform-cost baseline.
- Greedy Best-First: heuristic-only, fast but not always optimal.
- Jump Point Search (JPS): A* pruning for uniform grids.
- Theta*: any-angle extension of A*.
- Rapidly-exploring Random Trees (RRT): sampling-based planning.
- NavMesh: industry standard to abstract walkable space into a sparse graph for efficient search and funnel smoothing.

Algorithm / Method

Grid mode

- Representation: 2D grid, 4/8-neighbor moves, obstacles block cells.
- Algorithms: A*, Dijkstra, Greedy, JPS, Theta*, RRT.
- Costs: unit step (or Euclidean for any-angle), heuristic is Manhattan for A* and Greedy.

NavMesh mode (simplified)

- Representation: merge walkable cells into rectangles (polygons); shared edges are portals; polygon centers are nodes.
- Algorithms: A*, Dijkstra, Greedy on the polygon adjacency graph; NavMesh RRT samples polygon centers/goal.
- Funnel: portal sequence → funnel/string pulling → straightened path; then densified for animation.
- Costs: edge cost = distance from polygon center to portal midpoint; heuristic = center-to-goal distance; path cost = Euclidean length of the funnel path.

Implementation / Results / Evaluation

Implementation

- Frontend: HTML5 canvas for grid/NavMesh visualization and agent animation.

- Grid tools: add obstacles, set agents, pick algorithms, start/pause/reset.
- NavMesh: obstacle editing disabled; shows polygons and centers; start/goal selection only.
- Visualization: paths as dashed lines; explored nodes shaded in light agent color; polygon centers always visible (green).
- Metrics: computation time (ms), nodes explored, path cost/length.

Observations

- A* vs Dijkstra: A* expands fewer nodes; both optimal on uniform costs.
- Greedy: very fast, not always shortest.
- JPS: big speedup on uniform grids; retains optimality.
- Theta*: smoother, shorter-looking paths due to any-angle edges.
- RRT: finds feasible paths in tricky layouts; not optimal.
- NavMesh A*/Dijkstra/Greedy: fewer nodes than grid searches on large open areas; funnel yields straightened paths.
- NavMesh RRT: robust in open space; quality depends on samples.

Limitations

- NavMesh uses rectangle merging, not full triangle meshes; portal costs are approximated via centers/midpoints.
- Collision and mesh generation are grid-derived; not tuned for arbitrary continuous geometry.

Acknowledgements

Thanks to open literature on A*, Dijkstra, JPS, Theta*, RRT, and NavMesh funnel methods, and course staff for guidance.

Key References (≤5)

1. Hart, Nilsson, Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths.” IEEE TSSC, 1968.
2. Dijkstra. “A Note on Two Problems in Connexion with Graphs.” Numerische Mathematik, 1959.
3. Harabor & Grastien. “Online Graph Pruning for Pathfinding on Grid Maps (Jump Point Search).” AAAI, 2011.
4. Nash, Koenig, Tovey. “Any-Angle Path Planning.” AAAI, 2009.
5. LaValle. “Rapidly-Exploring Random Trees: A New Tool for Path Planning.” Technical Report, 1998.