Pathfinding Algorithms: From Basics to Game-Ready Optimizations

Why Pathfinding Matters

Pathfinding Powers Game Movement

- Finds routes around obstacles
- Drives NPC + enemy movement
- Must run fast in real time

Three Core Algorithms

The Classic Trio

- BFS $\rightarrow$ uniform-cost shortest steps
- Dijkstra $\rightarrow$ weighted optimal paths
- A* $\rightarrow$ optimal + heuristic

Breadth-First Search

BFS in a Nutshell

- Expands by distance layers
- Guarantees fewest steps
- Good for simple grids

Dijkstra's Algorithm

Costs Matter

- Uses priority queue
- Handles terrain/weights
- Guarantees optimal path

A* Core Formula

A = *Dijkstra + Heuristic**

- f(n) = g(n) + h(n)
- g = cost-so-far
- h = estimated distance

Heuristics

Choosing Good Heuristics

- Manhattan $\rightarrow$ grids
- Euclidean $\rightarrow$ free movement
- Must not overestimate

Pause & Reflect

Pick a Heuristic Prompt: Which heuristic fits a grid with diagonal movement—and why?

Why Games Need More

Raw A *Isn't Enough**

- Many agents = heavy load
- Real-time constraints
- Need shortcuts

Hierarchical Pathfinding

Think Regions First

- Divide map into clusters
- Global then local path
- Speeds up large maps

Navigation Meshes

NavMesh Basics

- Convex polygons
- Low node count
- Natural movement

Jump Point Search

JPS for Uniform Grids

- Skips straight segments
- Finds forced neighbors
- 10–20$\times$ faster on open maps

Goal Bounding

Early Rejection

- Detect unreachable goals
- Precompute reachability
- Avoid costly searches

Cached & Incremental

Reuse Paths

- Cache recent routes
- Update with D* Lite
- Great for moving goals

Theta* Family

Smoother Paths

- Line-of-sight shortcuts
- Natural movement
- Fewer waypoints

Flow Fields

For Many Units

- Shared destination
- Precomputed field
- Ideal for RTS

Time-Slicing

Distribute Work

- Avoid frame spikes
- Multi-frame compute
- Background threads

Steering & Avoidance

Pathfinding ≠ Movement

- Global path = route
- Steering avoids collisions
- RVO for multi-agent cases

Precomputed Data

For Static Maps

- Precompute distances
- Speed strategic decisions
- Memory tradeoff

Path Simplification

Clean the Path

- Remove unnecessary nodes
- Line-of-sight tests
- Smoother animation

Pause & Apply

Choose Optimizations Prompt: For 200-unit RTS on a large map, which two optimizations would you combine—and why?

Full Pipeline

A Practical Game Navigation Stack

- NavMesh core
- A*/JPS for requests
- Steering for local fixes