Lecture 20

Pathfinding
Take Away for Today

• What are the primary goals for pathfinding?

• Identify advantages/disadvantages of A*  
  • In what situations does A* fail (or look bad)?  
  • What can we do to fix these problems?

• Why combine steering and A*?  
  • Is this combination always appropriate?

• What do commercial games use?
Pathfinding

- You are given
  - Starting location $A$
  - Goal location $B$
- Want **valid** path $A$ to $B$
  - Avoid “impassible” terrain
  - Eschew hidden knowledge
- Want **natural** path $A$ to $B$
  - Reasonably short path
  - Avoid unnecessary turns
  - Avoid threats in the way
Abstraction: Grid & Graph

- Break world into grid
  - Roughly size of NPCs
  - Terrain is all-or-nothing
  - Majority terrain of square
  - Terrain covering “center”

- Gives us a weighted graph
  - Nodes are grid centers
  - Each node has 8 neighbors
  - Weight = distance/terrain

- Search for shortest path

- Real distance not required
  - 14:10 ratio for diagonals
  - Allows us to use integers
Breadth-First Search (Lab 2)

### Intuition

- **Search maintains**
  - Current node, initially **start**
  - List of nodes to visit

- **Basic Steps**
  - Have we reached the **goal**?
  - Add neighbors to **end** of list
  - Work from **first** node in list
  - Process “first-in first-out”

### Algorithm

```plaintext
n = start; L = { };  
while (n not goal) {
    add n to visited;  
    N(n) = unvisited neighbors
    foreach (m ∈ N(n)) {
        add m to end of L;  
    }
    n = removeFirst(L);  
}  
return path to goal;
```
Pathfinding: Breadth-First
Pathfinding: Breadth-First

The diagram illustrates a grid with obstacles marked by 'X'. The objective is to find the shortest path from point A to point B. The numbers represent the cost of moving through each cell. The green path shows the optimal route taken by the Breadth-First algorithm.
Breadth-First is Slow!

- Searches too many grids
  - Grids far away from goal
  - Works “radially outward”
- What is the problem?
  - Using graph algorithms
  - No spatial knowledge
- **Idea**: Spatial+Graph
  - Measure distance normally
  - Pick neighbor close to goal
Heuristic Search

Intuition

- Modified version of BFS
  - Have a list of candidates
  - Always pick *best* candidate
- Need $f$, **heuristic** function
  - Used to pick next step
  - Avoids stupid choices
- Regularly **update** $f$
  - Recompute on all neighbors
  - Reassign value if smaller

Algorithm

```plaintext
n = start; L = {};  
while (n not goal) {  
  add n to visited;  
  N(n) = unvisited neighbors  
  foreach (m ∈ N(n)) {  
    add m to L;  
    update f(m);  
  }  
  pick n ∈ L with f least;  
}  
return path to goal;
```
# Heuristic Search

## Intuition

- Modified version of BFS
  - Have a list of candidates
  - Always pick best candidate
  - Need $f$, heuristic function
  - Used to pick next step
  - Avoids stupid choices
  - Regularly update $f$
  - Recompute on all neighbors
  - Reassign value if smaller

## Algorithm

```plaintext
n = start; L = { };
while (n not goal) {
  add n to visited;
  N(n) = unvisited neighbors
  foreach (m ∈ N(n)) {
    add m to L;
    update f(m);
  }
  pick n ∈ L with f least;
  } return path to goal;
```

## Examples:

- **Dijkstra’s Algorithm**
  - $f = \text{dist. from source}$

- **Greedy Algorithm**
  - $f = \text{estimated dist. to goal}$

### Pathfinding
A* Algorithm

- **Idea**: Dijkstra + Greedy
  - \( g \): distance on current path
    - An “exact calculation”
    - Distance along graph
  - \( h \): estimated dist. to goal
    - *Spatial* distance
    - Ignores all obstacles
  - Final heuristic \( f = g + h \)
- Many variations for \( h \)
  - Regular distance
  - “Manhattan Metric”

Manhattan distance = 30 + 20 = 50
Pathfinding: A* Algorithm

Pathfinding

Game Design Initiative at Cornell University
Pathfinding: A* Algorithm
Pathfinding: A* Algorithm

A* Algorithm:
- \( f = g + h \)
- \( g \) is the cost from start to current node
- \( h \) is the heuristic cost from current node to goal
- \( f \) is the total cost from start to goal via current node

Grid representation:

- \( f \): Total cost
- \( g \): Cost from start
- \( h \): Heuristic cost to goal
Pathfinding: A* Algorithm
Pathfinding: A* Algorithm

In case of tie, use most recently added

The A* Algorithm is a pathfinding algorithm that efficiently searches for the shortest path between two points in a graph. It combines the costs of the path from the start node to a node (g-cost) and an heuristic estimate of the cost from the node to the goal (h-cost) to determine the lowest cost path. The algorithm expands nodes in order of their f-cost, which is calculated as f(n) = g(n) + h(n), where g(n) is the cost to reach node n, and h(n) is the heuristic estimate of the cost to reach the goal from node n.

The diagram illustrates the A* algorithm in action, with nodes represented as squares and the path being explored. The g-cost, h-cost, and f-cost are shown for each node, with the goal being to find the path with the lowest total f-cost.
Pathfinding: A* Algorithm
Issues with A*: Stair Stepping
Stair Stepping

- What is the problem?
  - Move one square at a time
  - All turns are at 45°

- **Idea:** Path smoothing
  - Path is a series of waypoints
  - Straight line between points
  - Remove unnecessary points

- Can combine with A*
  - Get *degenerative* solution
  - Remove to get waypoints

- Choose first q after p where
  - Line pq is valid
  - Point q has successor s
  - Line ps is not valid
Path Smoothing

A

B
Path Smoothing
Path Smoothing
Path Smoothing

A

B
Path Smoothing
Waypoints

Express paths as a sequence of segments

A

waypoint

B
Turning

- **Realistic** turns
  - Smooth paths into line segments
  - Round corners for realistic movement

- **Restricted** turns
  - Limit turns to angles drawn by artist
  - 16 angles standard for 2D top-down

- See online reading for today
  - Pinter, “Toward More Realistic Pathfinding”
  - Requires free registration to Gamasutra
Multiple NPC Sizes

- Grid to largest NPC?
  - Bad for small units
  - Unnecessary blocking

- Grid to smallest NPC!
  - Multiple squares for larger
  - Center fits on grid square

- Pathfinding larger NPCs
  - A* for center-to-center
  - Size to check blocking
  - May alter the path
Fitting NPCs on a Grid

- Assume NPC is square
  - Represents “reach”
  - Simplifies turning
- Requires “odd” sizes
  - Center must be a grid
  - Radius in full grid squares
  - What about even sizes?
- “Tabletop” solution
  - Round down when moving
  - Round up when in place
Waypoints

Express paths as a sequence of segments
Steering

- Alternative to pathfinding
  - Uses forces to move NPCs
  - Great for small paths

- Examples
  - Artificial potential fields
  - Vortex fields
  - Custom steering behaviors

- See Craig Reynold’s page
  - See “Physics & Motion”
  - Library: OpenSteer
Steering and Pathfinding

- Use waypoint as “goal”
  - Attract NPC to waypoint
  - When close, next waypoint
- Great for multiple NPCs
  - Pathfind for largest NPC
  - Steering to move along path
  - Repulsion keeps NPCs apart
- **Drawbacks:**
  - Military formations are hard
  - Get stuck at bottlenecks
Dynamic Obstructions

- Others can get in way
  - Enemies guarding locale
  - Friends waiting in queue
- Correct response?
  - Compute a new path?
  - Wait to be unblocked?
- What would you do?
  - See what is blocking
  - Making an educated guess
  - Character AI solution
Pathfinding in Practice

- **Navigation Meshes**
  - Indicates walkable areas
  - 2D geometric representation
  - Connected convex shapes
  - A* graph: center-to-center

- **Making Nav Meshes**
  - Often done by level editor
  - Can be modified by hand
  - Annotate special movement
  - **Example**: jump points
Easy Pathfinding on Meshes

Center of each Region

Corners of the Mesh
Summary

- **A* algorithm** is primary pathfinding tool
  - Make world into a grid/navigation mesh
  - Search for a path on associated graph
  - Adjust heuristics for terrain, threats

- But there are a lot of “special tricks”
  - Tricks to make movement realistic
  - Tricks to handle coordinated movement
  - Talk to Instructor (or TAs) if need more tricks