Game AI: Path Planning

Jeff Wilson

Graphs

- G = {N,E}
- N: Nodes
- E: Edges
  - cost

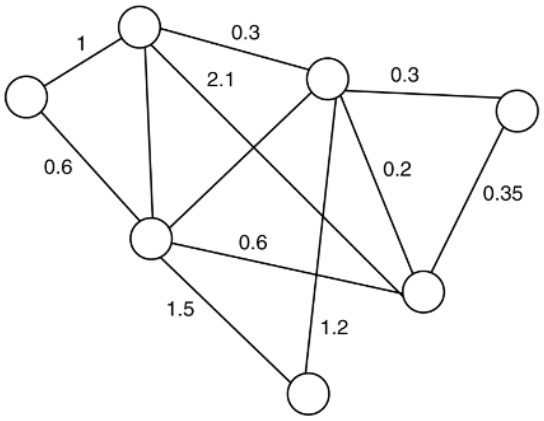
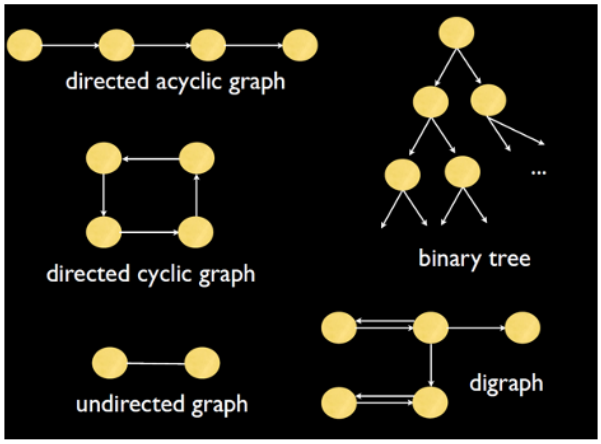
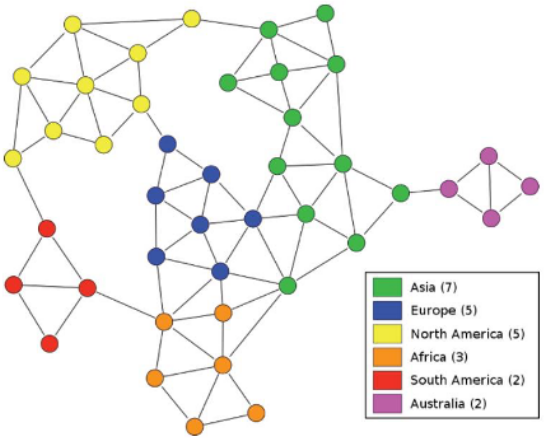


Figure 4.3: A weighted graph

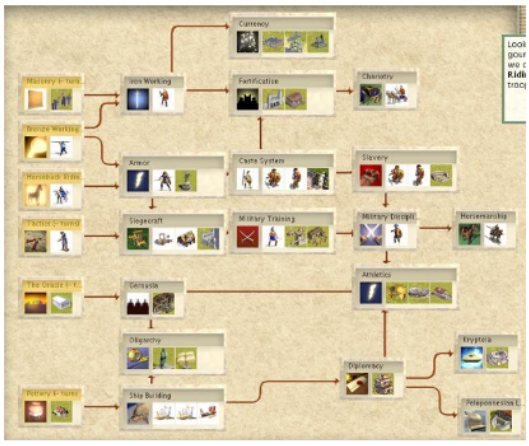
Graphs


directed acyclic graph
directed cyclic graph
undirected graph
binary tree
...
digraph

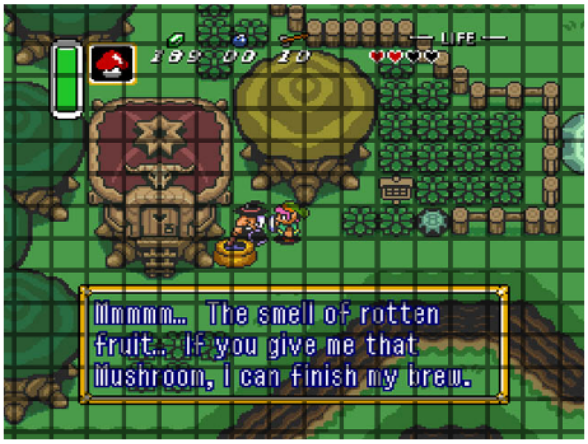
Risk


Asia (7)
Europe (5)
North America (5)
Africa (3)
South America (2)
Australia (2)

RTS Dependency Tree



Graph from Uniform Structure (Grid)



Zelda

What about other game types with unique geometry?

Is this graph sufficient for representing the rooms? To be revisited…

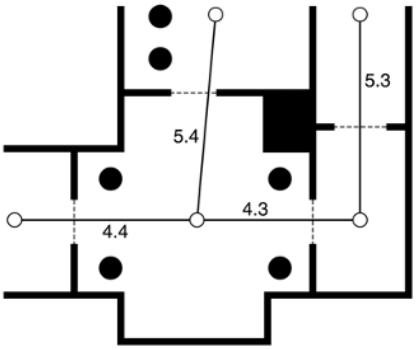


Figure 4.5: Weighted graph overlaid onto level geometry

Planning

- Part of intelligence is the ability to plan
- Move to a goal
  - A Goal State
- Represent the world as a set of **States**
  - Each configuration is a separate state
- Change state by applying **Operators**
  - An Operator changes configuration from one **state** to another **state**

Path Planning

- States:
  - Location of Agent/NPC in space
  - Discretized space
    - Tiles in a tile-based game
    - Turn 3D surface to pixels (grid lattice)
    - Floor locations in 3D
    - Voxels
    - Waypoints/"breadcrumbs"
    - Navmesh
- Operator
  - Move from one discrete location to next
  - Modifies state of modeled world, hopefully moving towards goal state
- **Graphs facilitate formal planning**

9

Path Planning Algorithms

- Must **Search** the state space to move NPC to goal state
- Computational Issues:
  - Completeness
    - Will it find an answer if one exists?
  - Time complexity
  - Space complexity
  - Optimality
    - Will it find the best solution

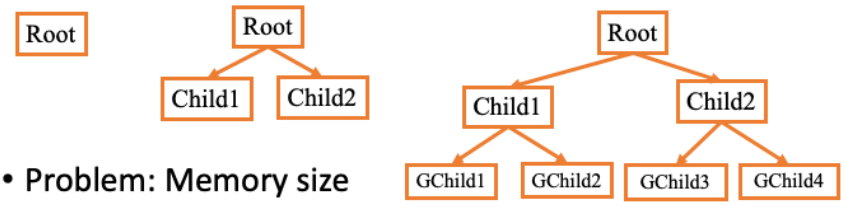
10

Search Strategies

- Blind search
  - No domain knowledge.
  - Only goal state is known
- **Heuristic search**
  - Domain knowledge represented by **heuristic rules**
  - Heuristics drive low-level decisions
  - Video games provide domain knowledge that can be leveraged!

Breadth-First Search

- Expand **Root node**
  - Expand all **Root node's children**
    - Expand all **Root node's grandchildren**

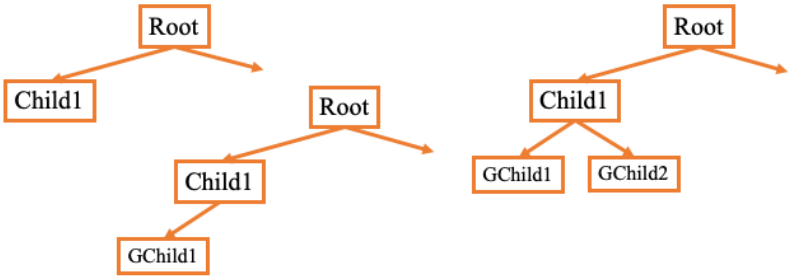


- Problem: Memory size

12

Depth First Search

- Always expand the node that is deepest in the tree
- Not best solution (if you stop at first found)



13

Path Finding Solved?

http://www.youtube.com/watch?v=lw9G-8gL5o0

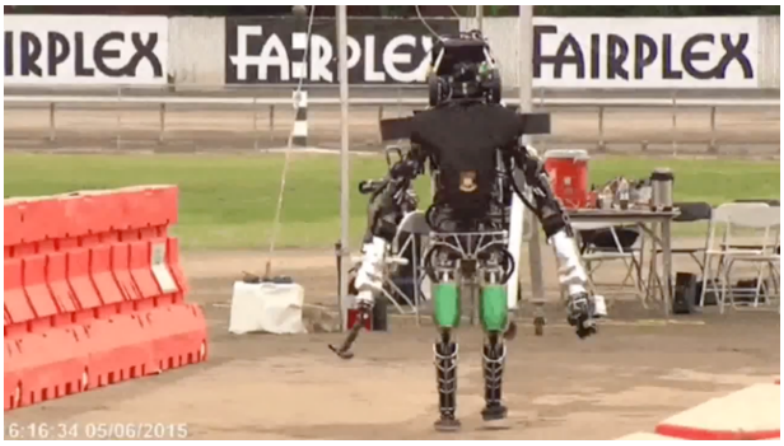
Pathfinding in Sim City

https://www.youtube.com/watch?v=zHdyzx_ecbQ

Pathfinding in Half-Life 2

Pathfinding waypoint structure doesn't align with the simulated space
http://www.youtube.com/watch?v=WzYEZVI46Uw&feature=player_embedded

Pathfinding DARPA Robotics Challenge



More of an inability to even begin to follow a path
https://www.youtube.com/watch?v=g0TaYhjpOfo

Issues

- Failures when graph representation or intended path is not coordinated with:
  - The AI Entity (vehicle)
  - Simulated space (the environment)

Tile-based Games

- (Usually) 1 to 1 coordination of path-planning graph and simulated space
- AI exist in discrete locations

Xcom Enemy Unknown

Grid Navigation

- 2D tile representation
  - Squares (4 or 8 way connectivity)
  - Hex (6 way connectivity)
- Often enforce one unit/entity per cell
- Each cell has terrain type
- Traversable or not

- 2.5D variants possible

Jagged Alliance
Mechwarrior Tactics

http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Discretization of Continuous Space

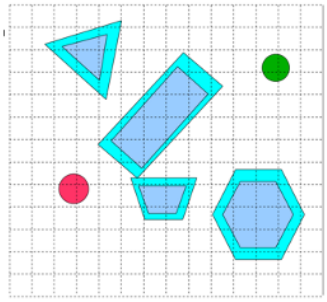
- How to generate?
- Validity
- Quantization
- Localization
- Agent Movement
- Search Efficiency

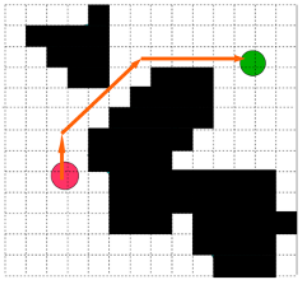
Quake 3 Arena

Generation

- Determine boundaries and terrain type regions
- …but often a transition across types of terrain

Discretized Space – Grid Lattice

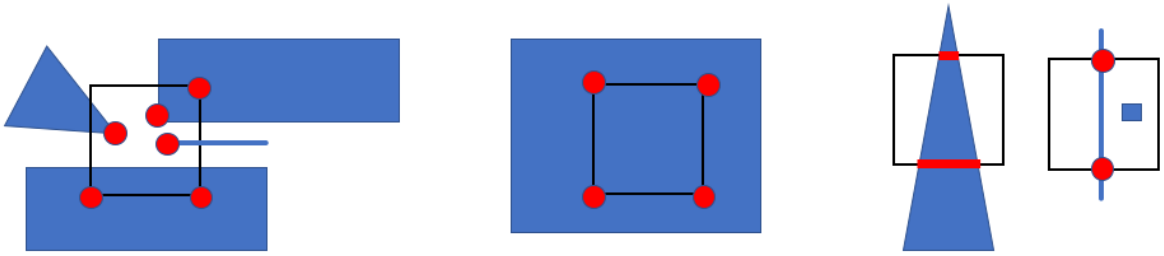




23

Generation

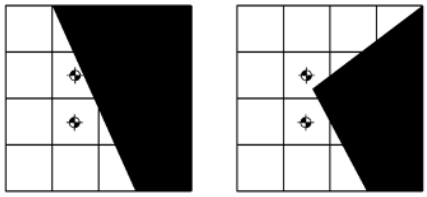
- Verify world boundaries don't go through grid lines
- Verify no obstacle point within a grid cell, or vice versa Verify obstacle edge does not intersect grid cell edge
- Sometimes useful to mark both edges and cells as traversable (can help NPC get back on navigable grid if on a partially blocked cell)



Purple/blue are static obstacles, black square is grid cell, red denotes a positive intersection test of points (circles) or line segments

Validity

- Validity: Consider A, B are nodes in a discretized space. From any point in node A, can an agent travel in a straight line to any point in adjacent node B?
- (For grid lattice, only applicable with continuous space)
- **NOTE: Validity considerations not necessary for Homework 1!**



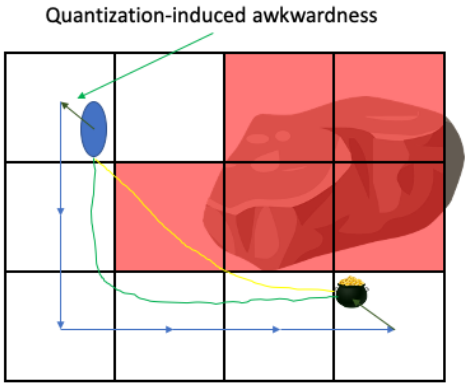
Valid partial blockage Invalid partial blockage

Off Grid?

- What to do if an AI entity is off a navigable grid cell?
- Local steering behaviors (raycasting+greedy)
- Cells marked as not traversable can still have meta data denoting edges that are fully/partially passable
- Random movement and physics engine to slide/bump around and hope to eventually get to a valid position
- Just cheat a teleport (especially if the game player isn't looking)

Continuous Position/Movement versus Path from Discrete Graph

- Awkward/blocky movement
- Post process path
  - "string pulling"
- Steering behavior at runtime (greedy)
  - May need to augment the environment (special colliders for pits, etc.)



Post Process (String Pulling)

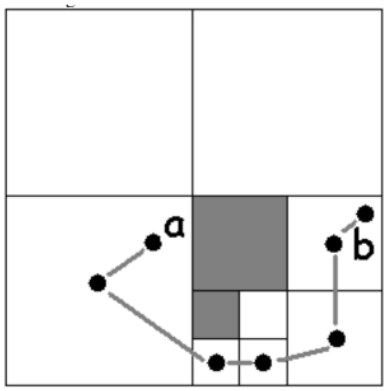


Figure 3. A not entirely good path

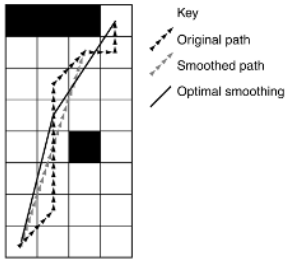


Figure 4.36: Smoothed path with optimal smoothing indicated

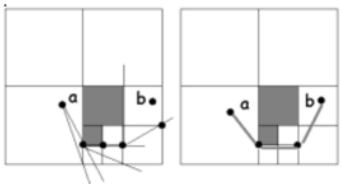
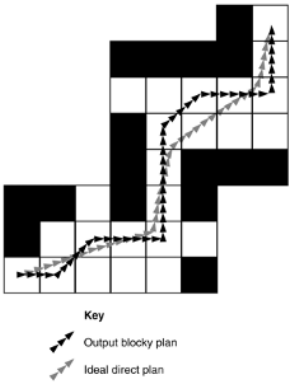


Figure 5. Visibility regions (left) and the final path (right)



[1]
I. L. Davis, “Warp speed: Path planning for star trek: Armada,” presented at the AAAI Spring Symposium (AIIDE), 2000, pp. 18–21.

Search Efficiency for Grid

- Search algorithm must visit a lot of nodes
- Search spaces can quickly become huge
  – E.g. 100x100 map == 10k nodes and ~78k edges

A Simple Greedy Search

- Uses a heuristic!
- Expand the node that yields the minimum cost
  - Consider a current node (starting with start node)
  - Expand the node that is closest to target
  - New current node
  - Terminate if new current node isn't closer to goal than previous
    - Avoids infinite loop if not storing "visited" metadata to consider
  - Repeat until goal found or no options at current node
- Not Complete!
- "Greedy" implies that working solution is not revised
- Local Minima/Maxima (dead end)