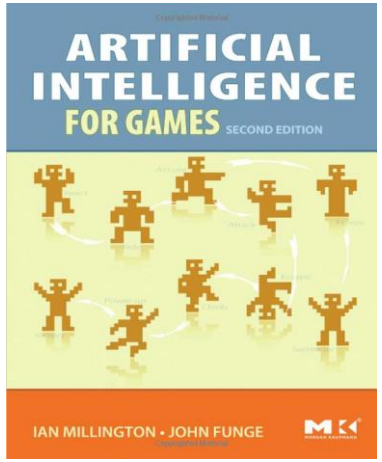
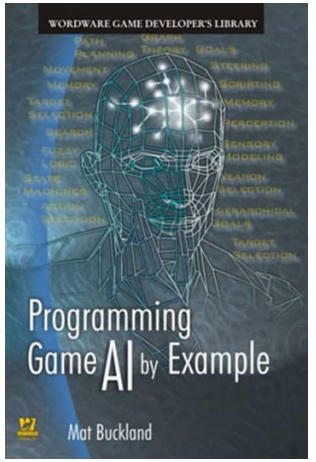


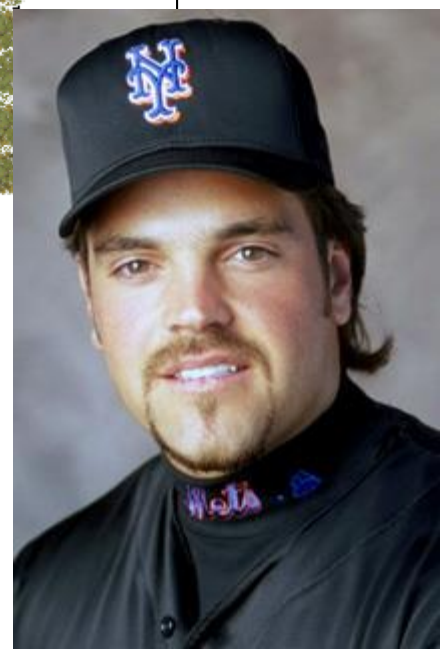


"the question of whether computers can think is like the question of whether submarines can swim" -- Dijkstra

Game AI: The set of algorithms, representations, tools, and tricks that support the creation and management of real-time digital experiences

Erratum

- Wolfenstein 3D was 3rd Wolfenstein, but 1st FPS in series (‘92)
- The first-person shooter genre has been traced as far back as *Maze War*, development of which began in 1973, and 1974's *Spasim*. Later, and after more playful titles like *MIDI Maze* in 1987, the genre coalesced into a more wantonly violent form with 1992's *Wolfenstein 3D*, which has been credited with creating the genre's basic archetype, that subsequent titles were based upon.
  - https://en.wikipedia.org/wiki/First-person_shooter
- *Wolfenstein 3D* is important for popularizing the first person shooter and inventing many of the tropes that became standard in the genre.
  - https://en.wikipedia.org/wiki/Wolfenstein



PREVIOUSLY ON…

Class N-1: What is…

- GAI?
  - Set of tricks and techniques to bring about a particular game design.
  - Goals include:
    - enhancing the player's engagement, enjoyment, and experience
      - End behavior is the target
      - Do better than random
    - doing things the player or designer cannot do or don't want to do
      - replace real people when they are unwilling or unavailable to play
      - aid for designers and developers
    - making the entities, opponents, agents, companions, etc. in games appear intelligent
    - believable characters / looking convincing
- A Game?
  - A system of rules and a goal and agency.

N-1: How/Why distinct from “academic AI”

- Supporting the player experience
- Good game AI == matching right behaviors to right algorithms
- Product is the target, not clever coding – ends justify means. FUN
- Illusion of intelligence
- “Magic Circle” (Rules of play: game design fundamentals)
- Elegance in simplicity & the complexity fallacy
- Quality control & resource limits
- Fun vs smart: goal is not always to beat the player
- Optimal/rational is rarely the right thing to do

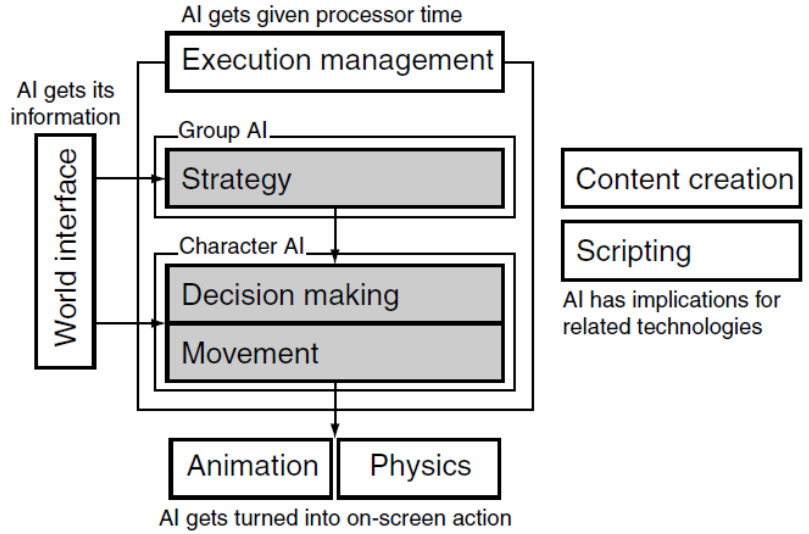
N-1: Common (game) “AI” Tricks?

- Move before firing – no cheap shots
- Be visible
- Have horrible aim (being Rambo is fun)
- Miss the first time
- Warn the player
- Attack “kung fu” style (Fist of Fury; BL vs School)
- Tell the player what you are doing (especially companions)
- React to own mistakes
- Pull back at the last minute
- Intentional vulnerabilities or predictable patterns

Liden, “Artificial Stupidity: The Art of Intentional Mistakes”. *AI Game Programming Wisdom*.

N-1: Major ways GameAI is used…

- In game
  - Movement
  - Decision making
  - Strategy
  - Tailoring/adapting to player individual differences
  - Drama Management
- Out of game
  - PCG
  - Quality control / testing



M&F Fig 1.1

N-1: Why AI is important for games

- Why is it essential to the modeled world?
  - NPC's of all types: opponents, helpers, extras, …
- How can it hurt?
  - Unrealistic characters → reduced immersion
  - Stupid, lame behaviors → reduced fun
  - Superhuman behaviors → reduced fun
- Until recently, given short shrift by developers. Why?
  - Graphics ate almost all the resources
  - Can't start developing the AI until the modeled world was ready to run
    - AI development always late in development cycle
- Situation rapidly changing / changed. How?
  - AI now viewed as helpful in selling the product
  - Still one of the key constraints on game design

Credit: Dr. Ken Forbus

Intelligent vs. random

Graphs, Search, & Path Planning

2018-01-11

Graphs

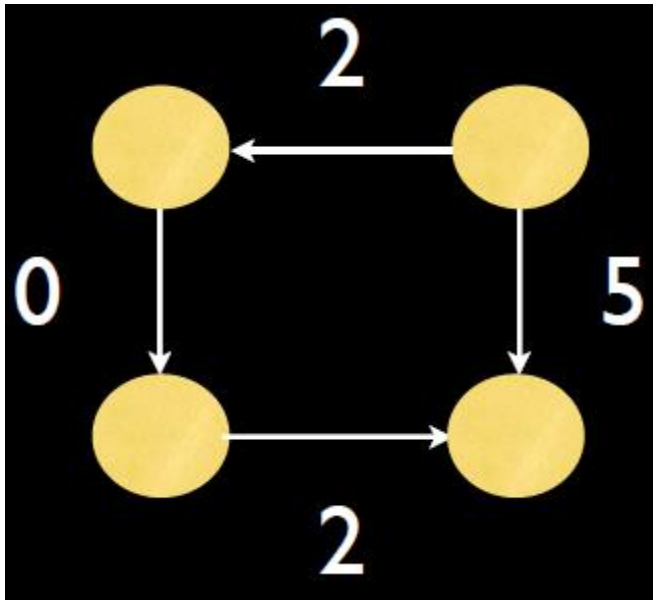
- What is a graph?
- What defines a graph?
- How can we represent them?
- How does representation effect search?

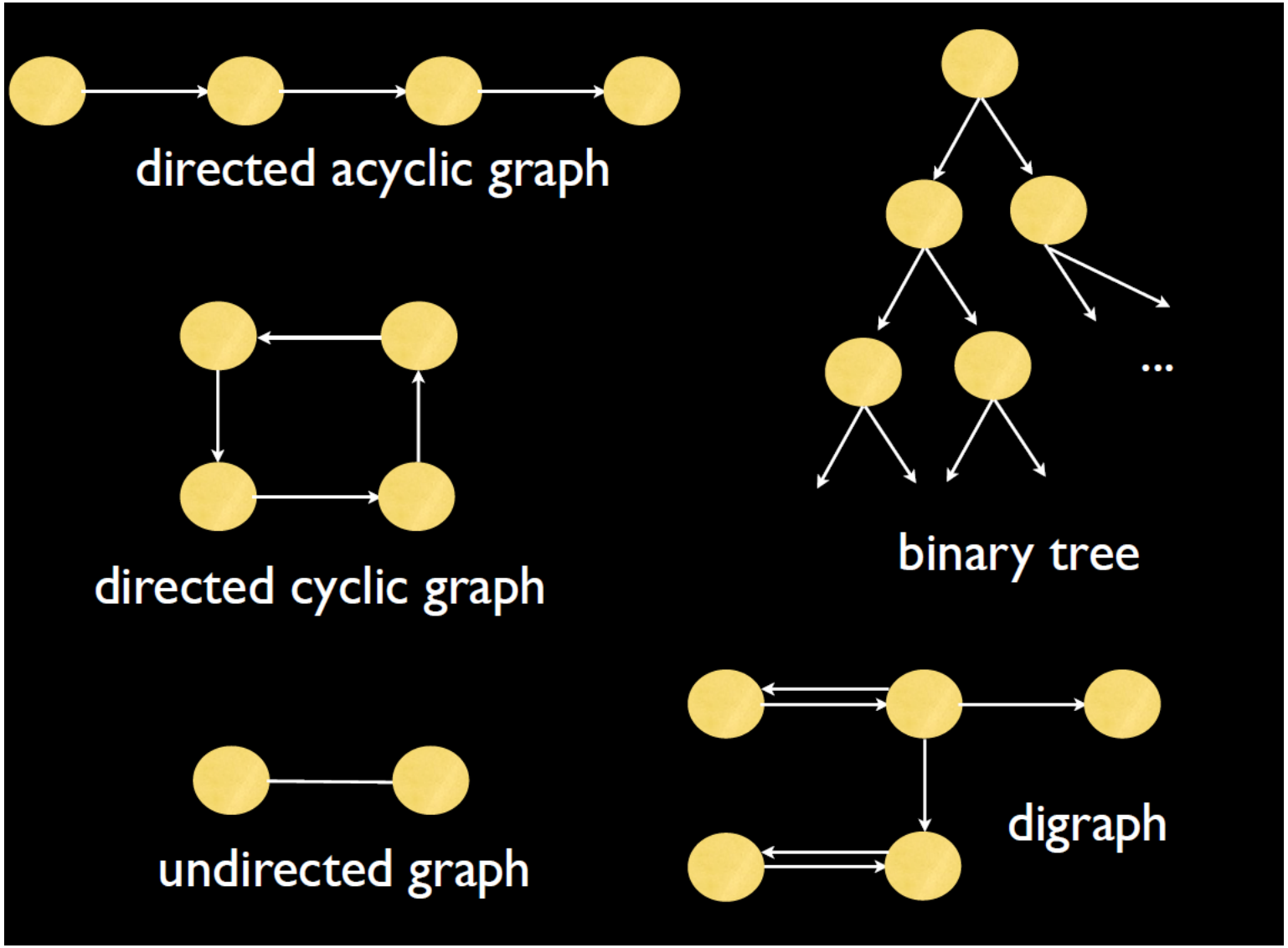
- **Applications to GAI?**

- See Buckland CH 5 for a refresher

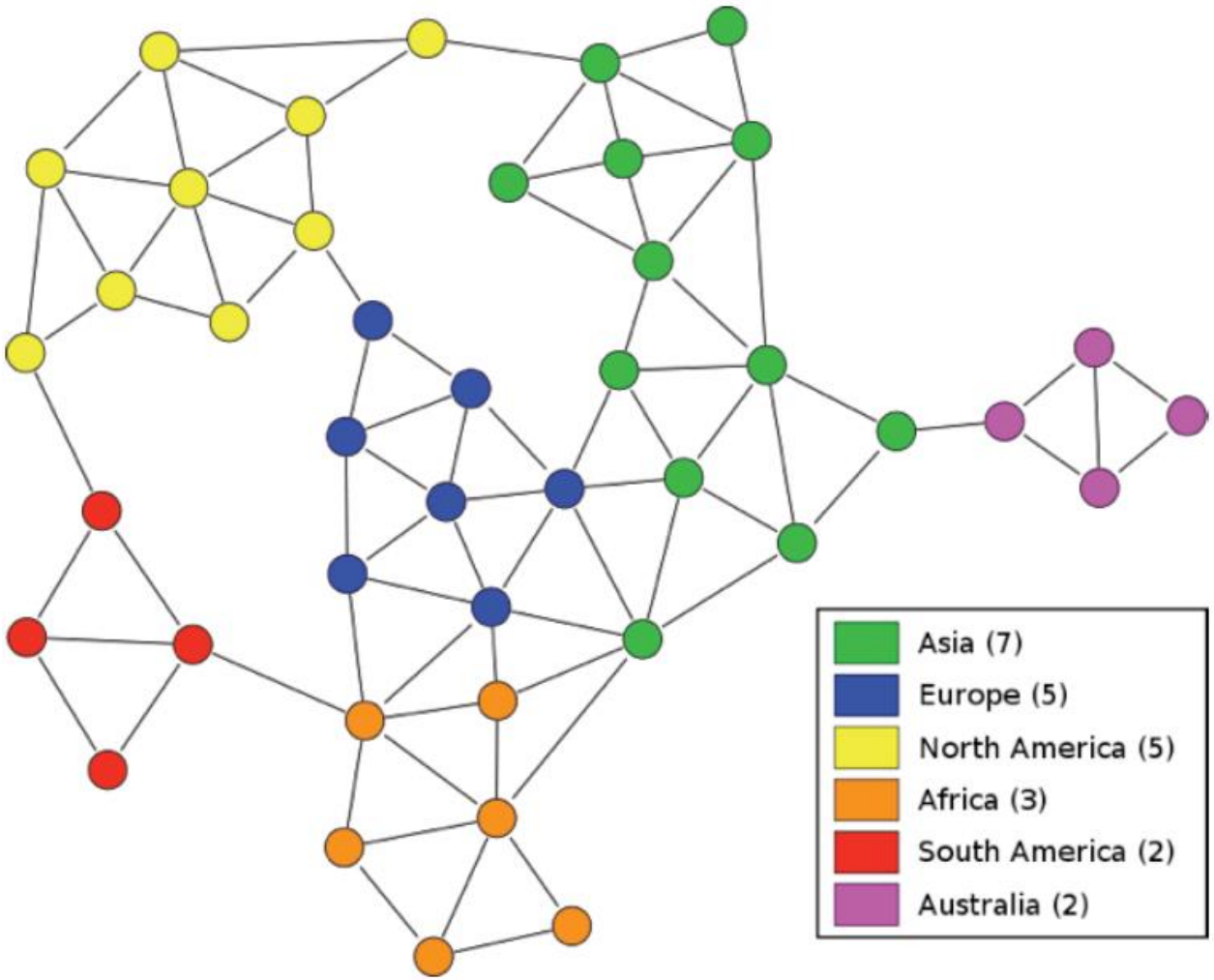
Graphs (2)

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

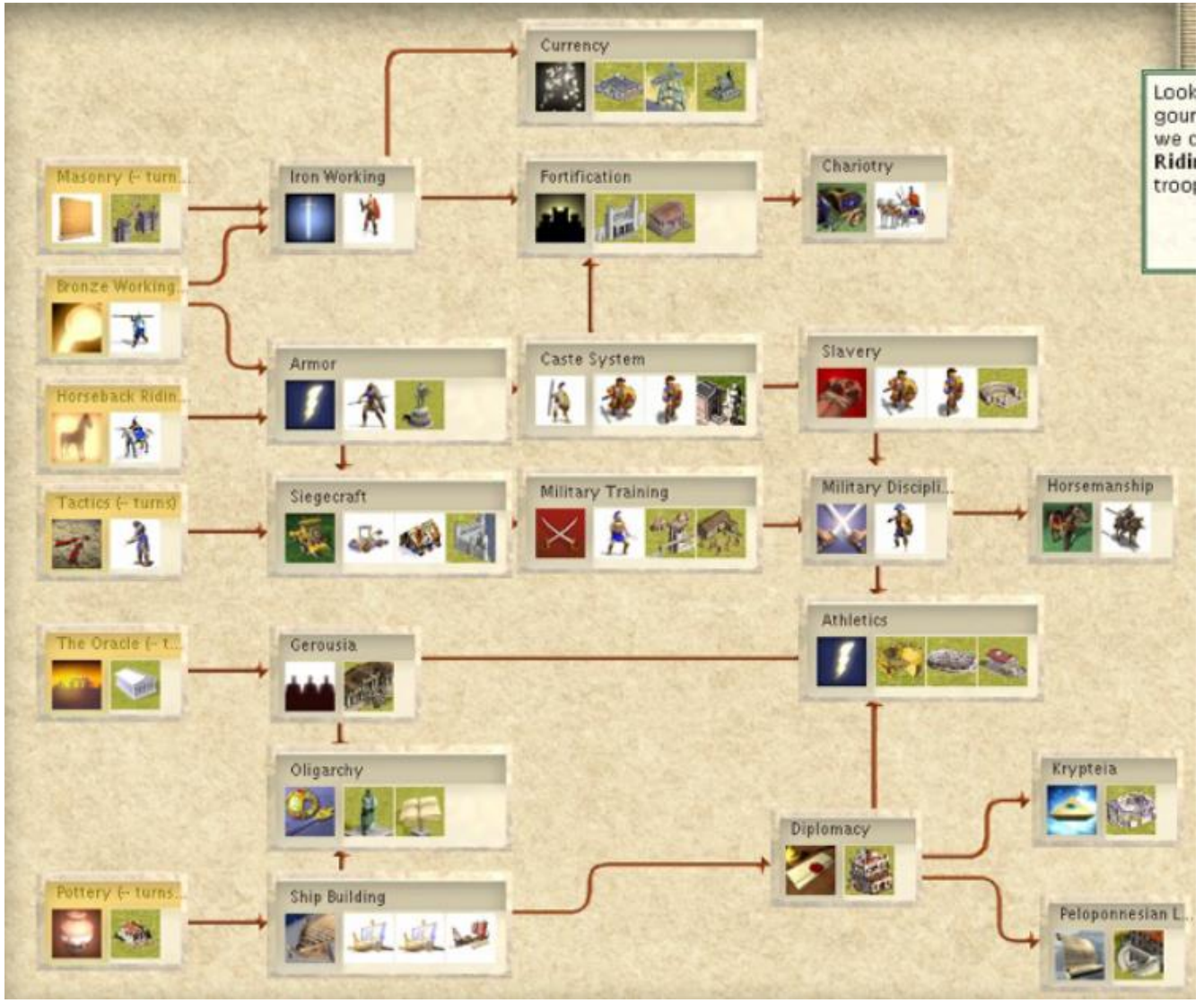



directed acyclic graph
directed cyclic graph
binary tree
...
undirected graph
digraph

Risk


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

RTS Dependency Tree



Graphs: Killer App in GAI

- Navigation / Pathfinding
- Navgraph: abstraction of all locations and their connections
- Cost / weight can represent terrain features (water, mud, hill), stealth (sound to traverse), etc
- What to do when …
  - Map features move
  - Map is continuous, or 100K+ nodes?
  - 3D spaces?

Graph Search

- Uninformed (all nodes are same)
  - DFS (stack – lifo), BFS (queue – fifo)
  - Iterative-deepening (Depth-limited)
- Informed (pick order of node expansion)
  - Dijkstra – guarantee shortest path ($E\log_2 N$)
  - A* (IDA*)…. Dijkstra + heuristic
  - D*

http://en.wikipedia.org/wiki/A*_search_algorithm

Heuristics

- [dictionary]*"A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood."*
- *h(n)* = estimated cost of cheapest path from n to goal (with goal == 0)

Path finding problem solved, right?

- Hall of shame:
  - Compilation
    - http://www.youtube.com/watch?v=lw9G-8gL5o0
  - Sim City (1, 2 … 5)
    - https://www.youtube.com/watch?v=zHdyzx_ecbQ
  - Half-Life 2
    - http://www.youtube.com/watch?v=WzYEZVI46Uw
  - Fable III
  - DOTA (Defense of the ancients) 1+2
  - WoW (World of Warcraft)
- DARPA robotics challenge:
  - https://www.youtube.com/watch?v=g0TaYhjpOfo

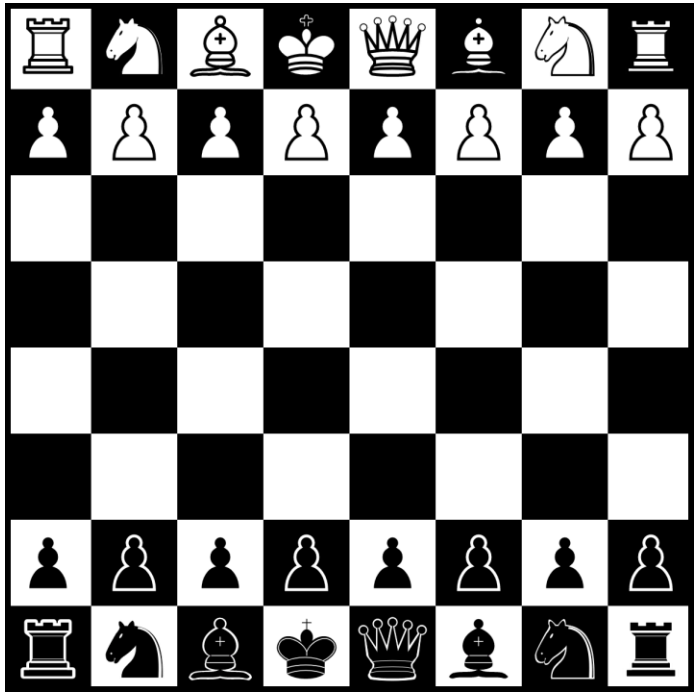
Path finding models

1. Tile-based graph – “grid navigation”
2. Path Networks / Points of Visibility NavGraph
3. Expanded Geometry
4. NavMesh

Model 1: Grid Navigation

- 2D tile representation mapped to floor/level
  - Squares, hex; 8 or 6 neighbors / connectivity
- Mainly RTS games
- One entity/unit per cell
- Each cell can be assigned terrain type
- Bit mask for non-traversable areas
- Navigation: A* (or perhaps greedy), Dijkstra
  - http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Grids

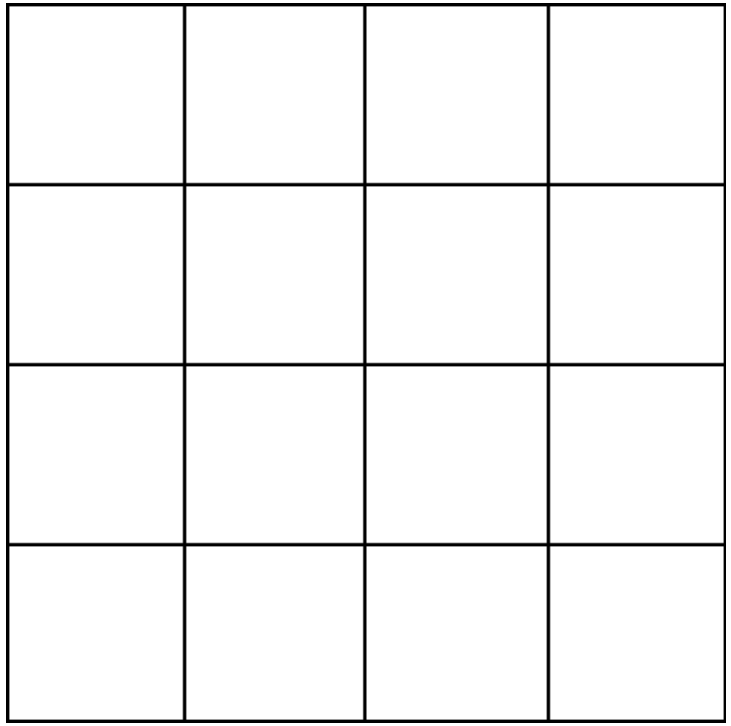


Also Grids


8h
Demkina
37 / 37
43 / 43
0
10 / 11
Depth 18
Centering Tool
Water Shortage Reported

Grids

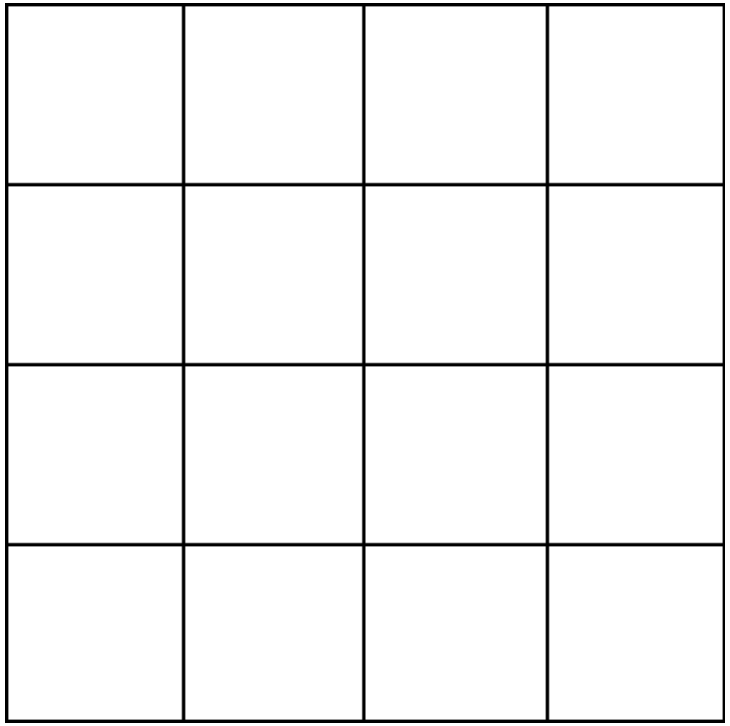
- 2D tile representation mapped to floor/level
  - Squares/hex cells
  - 8 or 4 neighbors / connectivity
  - Simplify the space
- At most one entity/unit per cell



Movement through Grids

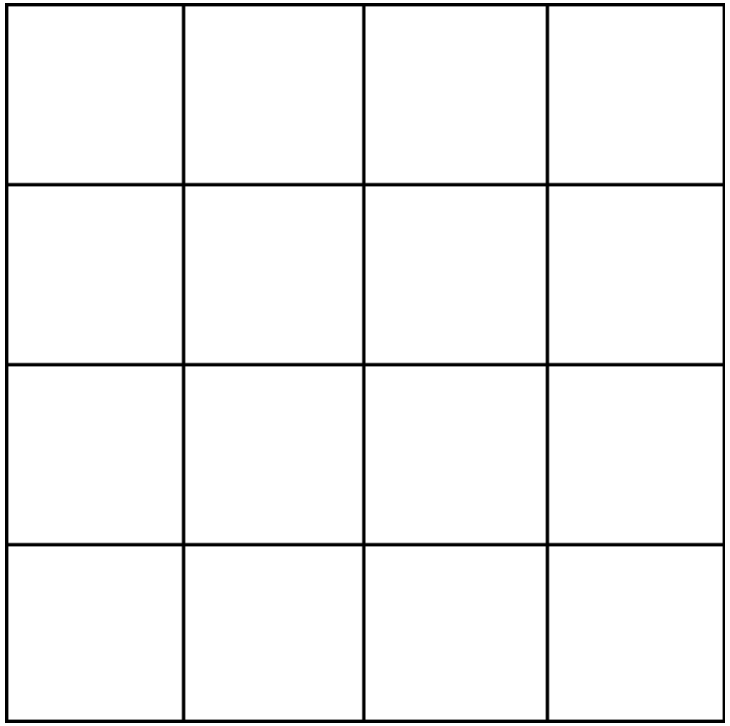
- Continuous or Discrete?

- If continuous, we need a path of cells from a current cell to a goal cell
  - Navigate from one cell center to the next

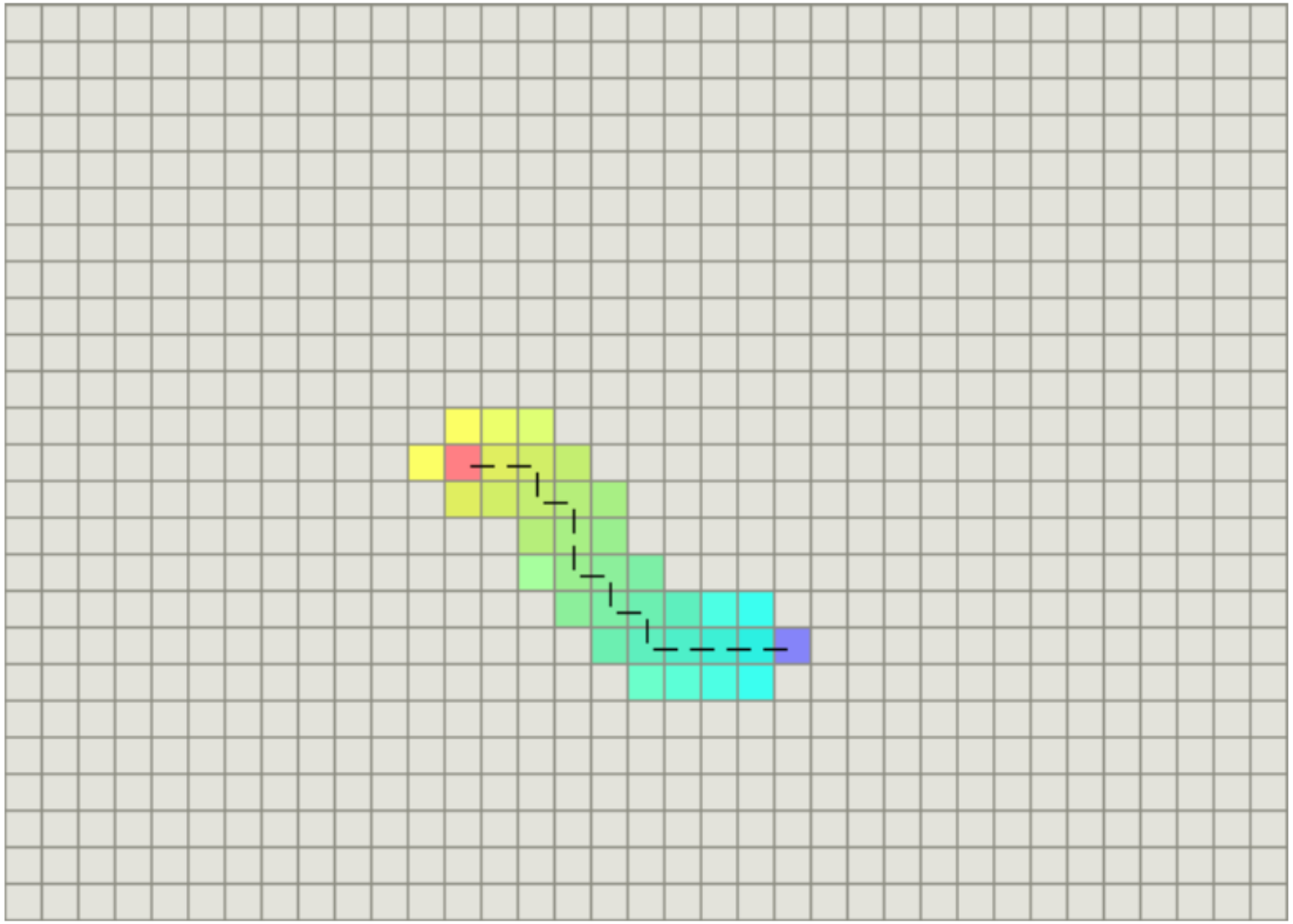


Greedy Path Planning

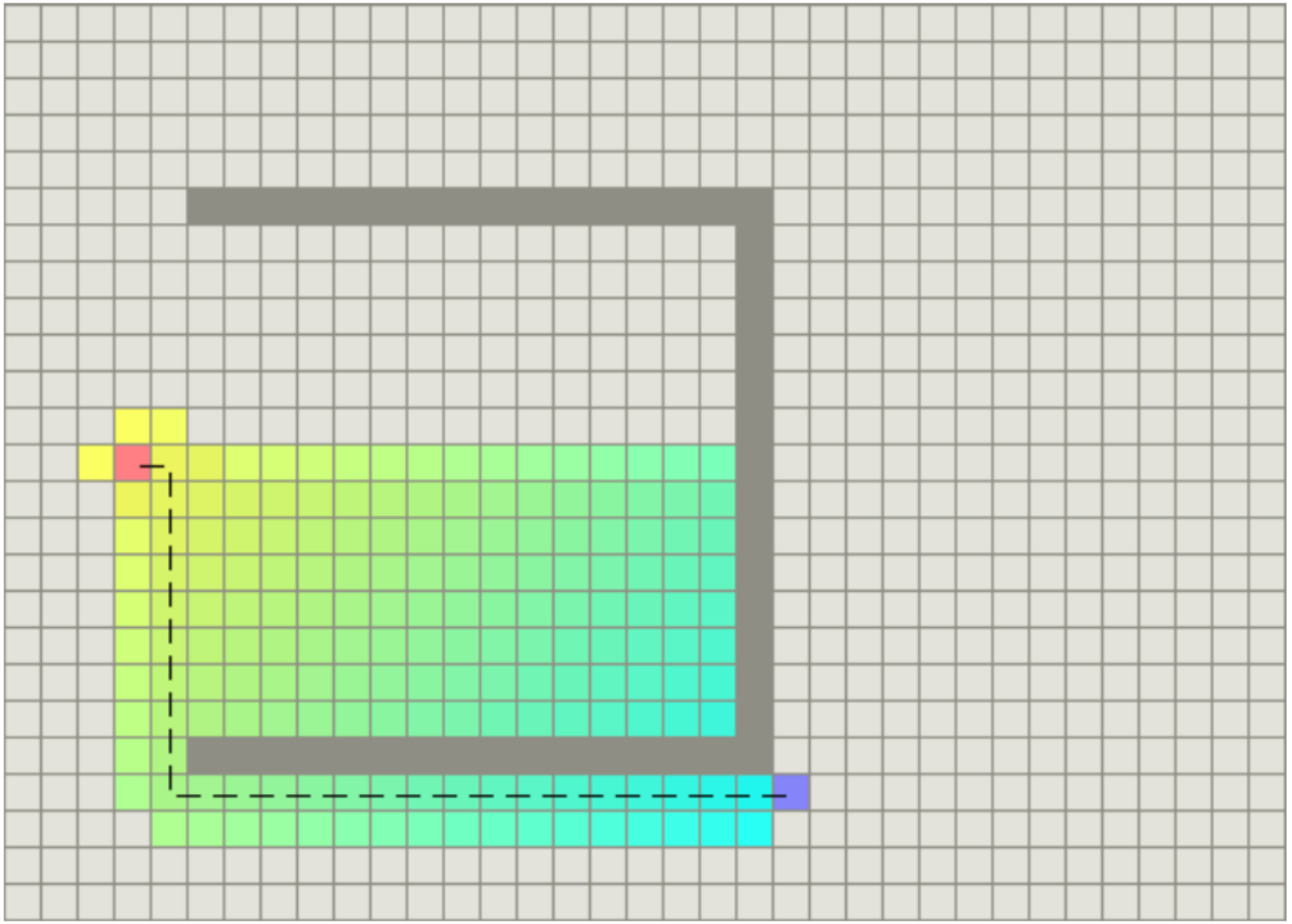
- Given current cell/node, pick the next cell that is closest to the goal cell according to some heuristic
- Once goal cell is reached, backtrack to the initial cell



Grid Path planning can be fast



Grid path planning can be very slow



Path Planner

- Initial state (cell), Goal state (cell)
- Each cell is a state agent can occupy
- Sort successors, try one at a time (backtrack)
- Heuristic: Manhattan or straight-line distance
- Each successor stores who generated it

Question

What are pros and cons of a grid representation of space in terms of character movement?

Grid navigation: pros

- Discrete space is simple
- Can be generated algorithmically at runtime (Hw1)
- Good for large number of units
- A*/greedy search works really well on grids (uniform action cost, not many tricky spots)

Grid navigation: cons

- Discretization “wastes” space
- Agent movement is jagged/awkward/blocky, though can be smoothed
- Some genres need continuous spaces
- Partial-blocking hurts validity
- Search must visit a lot of nodes (cells)
- Search spaces can quickly become huge
  - E.g. 100x100 map == 10k nodes and ~78k edges

New Problems

- Generation
- Validity
- Quantization
  - Converting an in-game position (for yourself or an object) into a graph node
- Localization
  - Convert nodes back into game world locations (for interaction and movement)
- Awkward agent movement
- Long search times

Validity