A* Path Finding

Matthew Guzdial

Announcements

- HW1 due tonight at 11:55pm
  - Yes, 23-hour grace period, but far reduced TA/instructor response times on HW1 questions.
- Updated office hour Google Meet on Syllabus (v3) meet.google.com/vks-rpjv-qvw
- HW2 has been released, due 11:55pm Oct 4th
- ~90% completed Monday's participation questions.
- Quiz 1 Space Representations + Path Planning next Friday
  - Covering all lecture material through next Wednesday (but focused on material before next Monday)

Grids in the News: Deltarune Ch. 2 Releases Today

Quizzes

- Quiz Format
  - Fill-in-the-blanks (MadLibs where you create your own question)
  - Combination of multiple choice and short answer
  - Questions will depend on each other.
  - **No lecture on quiz days**
  - Quizzes designed to take ~1 hour, but you'll have 48 hours to complete it when it is released at 11am next Friday

- Quiz 2 will be 2 weeks later covering both Pathing and Decision making.

Real Quick

Polygons, Vectors, and mor:
https://datacarpentry.org/organization-geospatial/02-intro-vector-data/

C# primer:

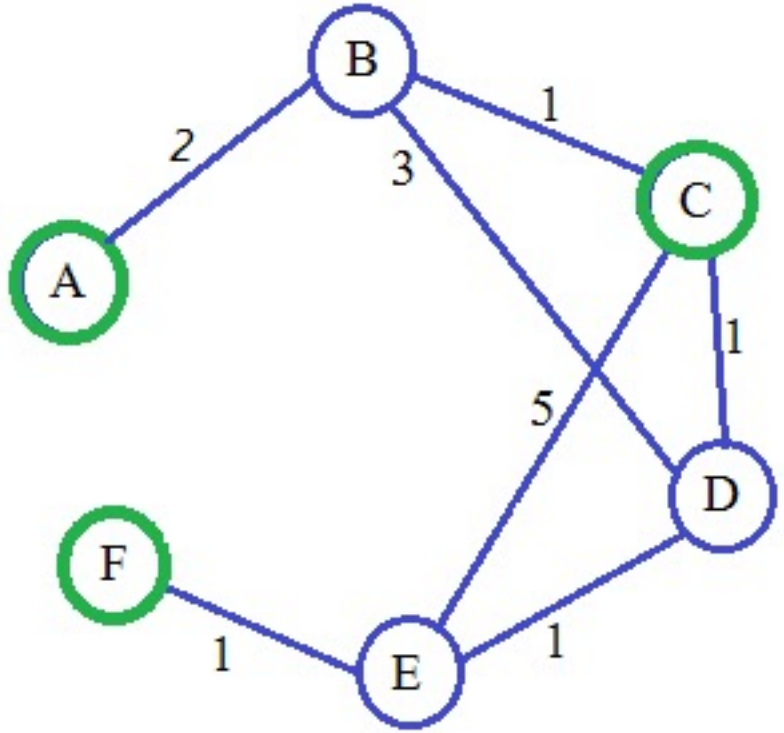
https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/tutorials/

Quick Review

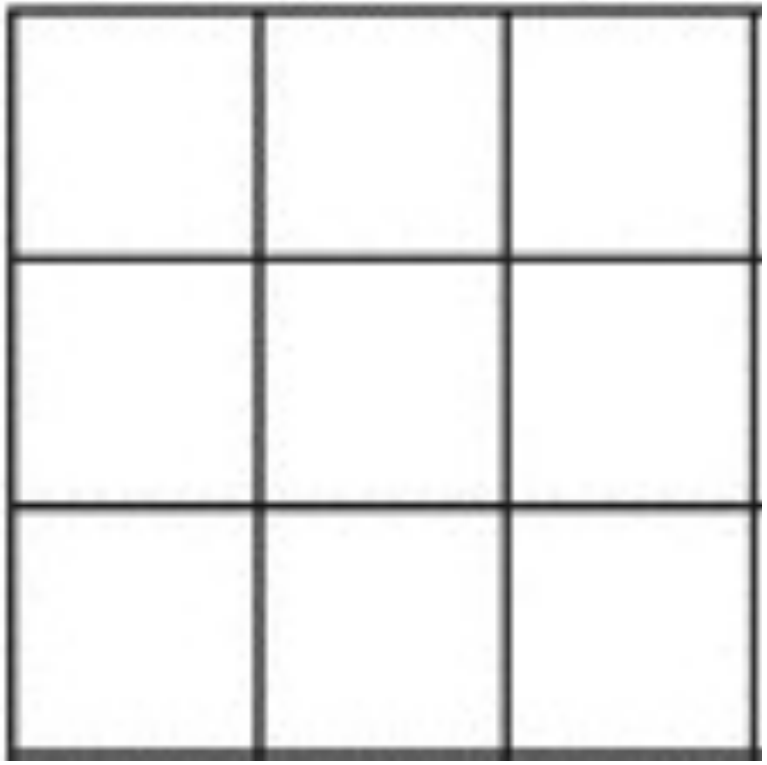
- Three ways to represent space
  - Grids
  - Path networks
  - Navigation Mesh
- Greedy Path Planning
- Three ways to post-process path movement
  - Path Smoothing
  - Steering (Blending, Flocking, etc.)
  - (Given multiple agents) Formations

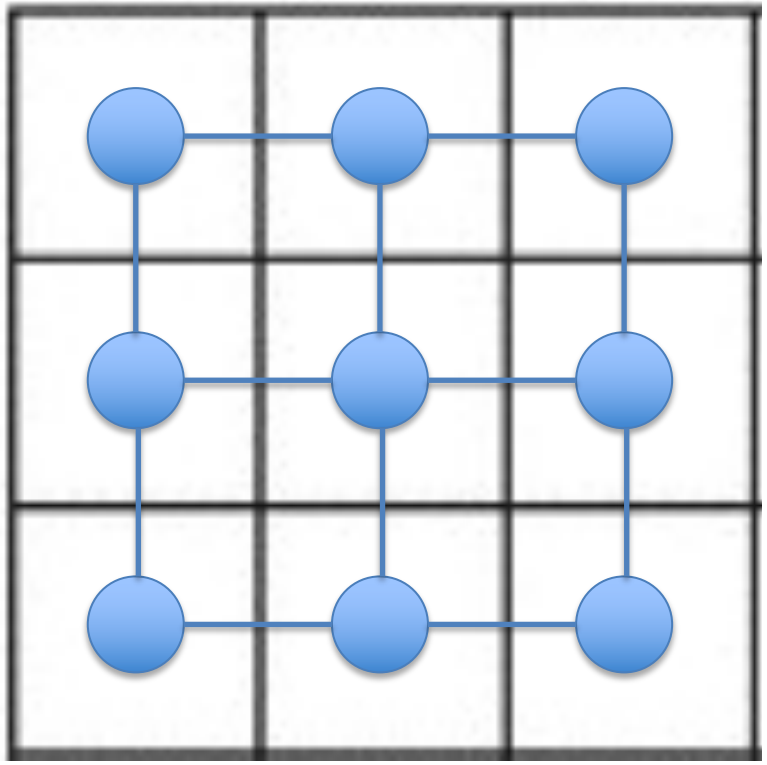
Graphs

- Nodes: Basic unit of the graph representation
  - Example: A point in space
  - Sometimes called vertex/vertices
- Edges: Links between nodes
  - Can have weights on the edge to represent cost to travel along it (e.g. distance)

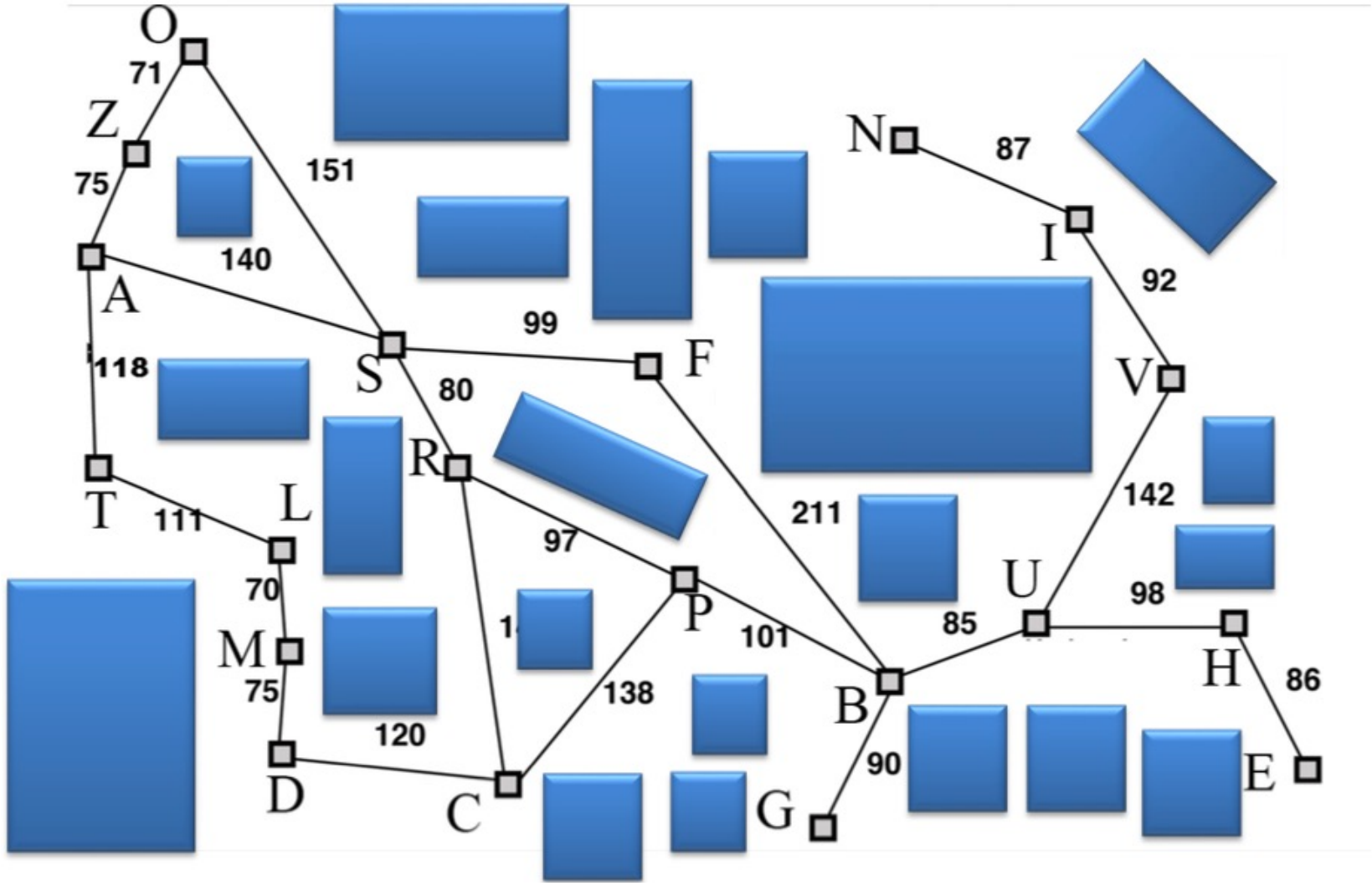


Grid as Graph



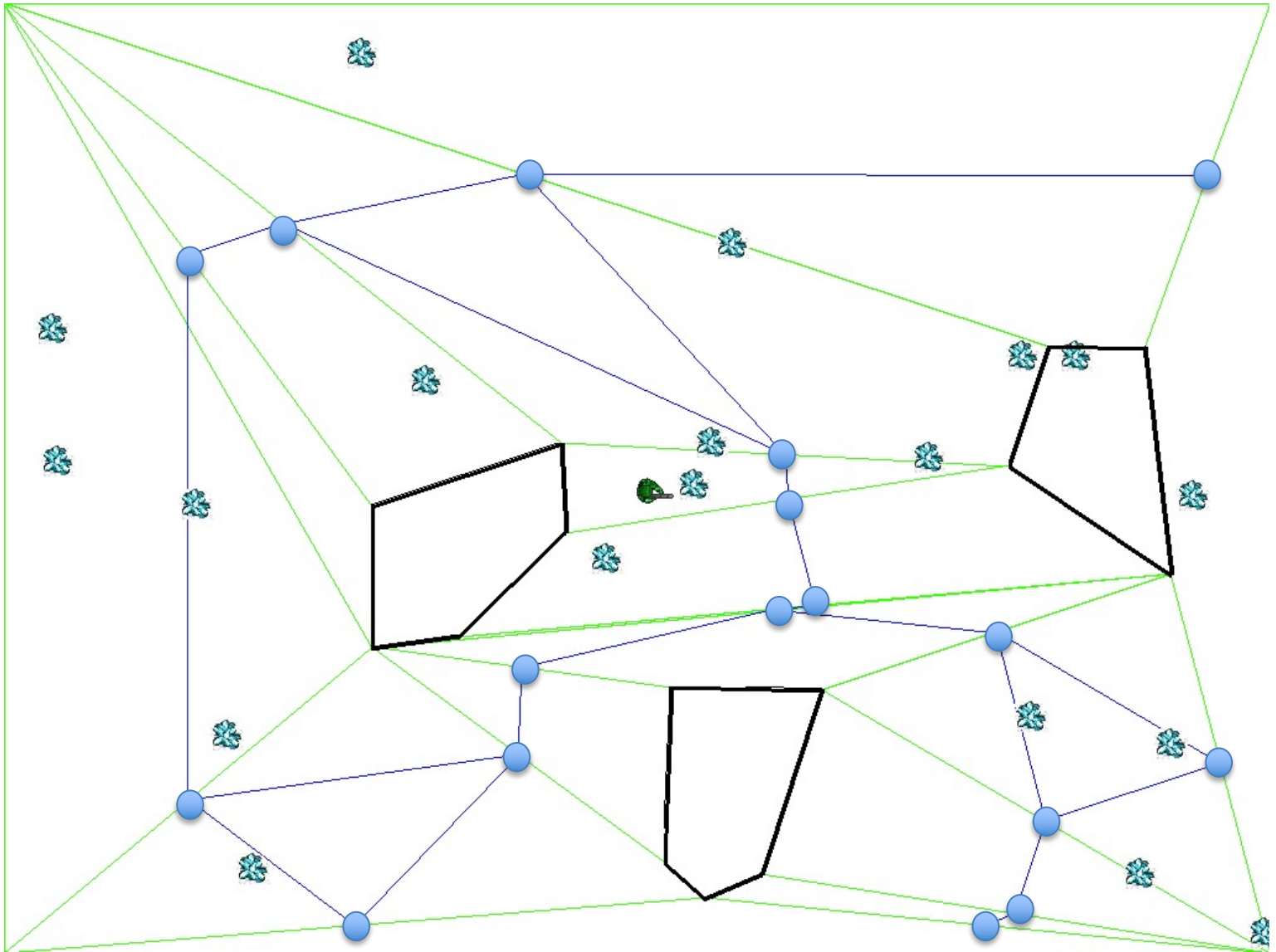


Path Network as Graph



Nav Mesh as Graph

(well actually path network again)



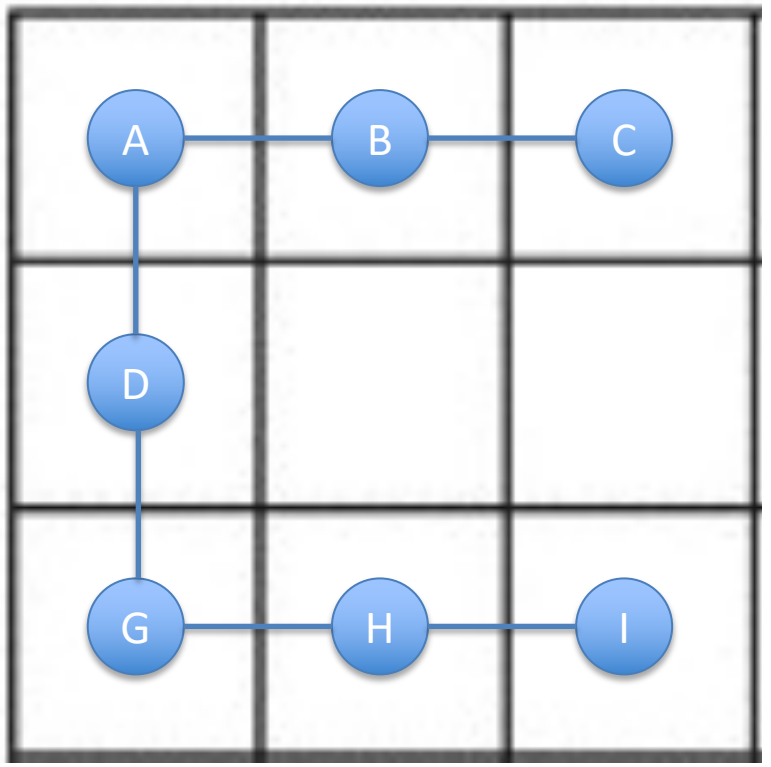
Why talk about these as graphs?

- Standard, abstract way to discuss different spatial representations
- Allows for quantifiable comparison between different spatial representations (e.g. number of edges/nodes)
- Allows us to discuss different search approaches without worrying about the exact spatial representation

Greedy Algorithm Review

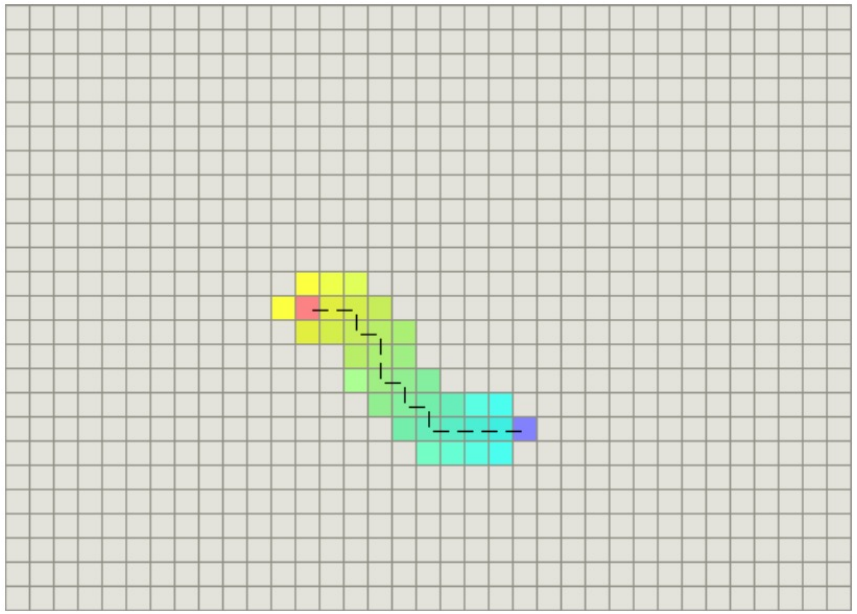
Find a path from start to goal node

1. Add the neighbors of the current node to some open set list
   - We can get here!
2. Pick next current node from open set
3. If next node is goal, backtrack to start of path



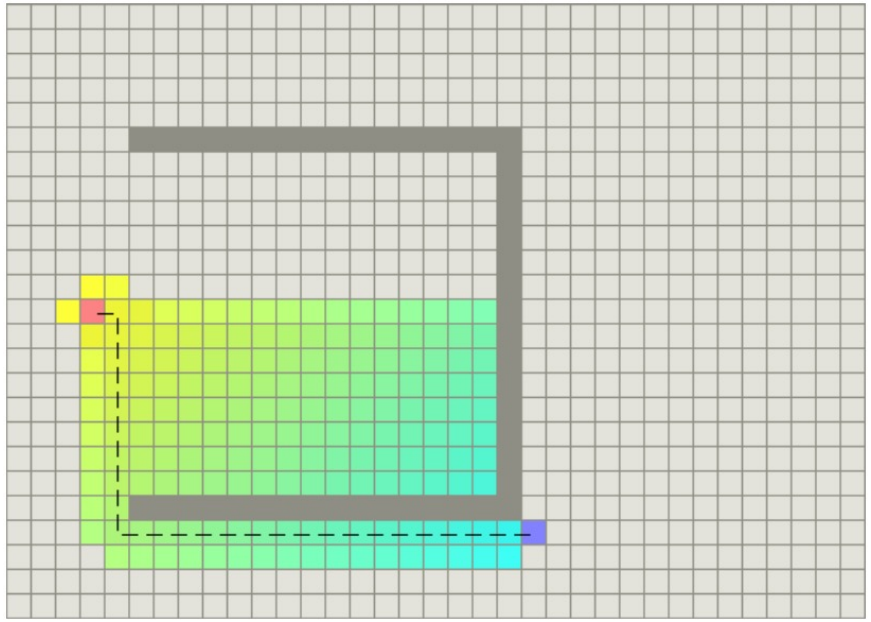
Improvement over Greedy

- Beyond improving the heuristic, how can we improve the greedy pathing algorithm?
- When does it fail?



A*

- Nodes will have two costs:
  - G score: Cost from getting from start to here
  - H score: Estimated cost of getting from here to goal
  - F score: G+H
- We will pick which node to choose next based on both of these scores



A*

- Won’t just have an open set, but also a closed set (nodes already evaluated)
- Open set will be a priority queue, so if we discover a better node (in terms of F score) we can immediately pick it

- Priority Queue: A queue that automatically sorts itself so minimum cost is at the top

A*

```
add start to openSet
while openSet is not empty:
    current = openSet.pop()
    if current == goal:
        return reconstruct_path(current)
    closedSet.Add(current)
    for each neighbor of current:
        if neighbor in closedSet:
            continue
        gScore = current.gScore + dist(current, neighbor)
        if neighbor not in openSet:
            openSet.add(neighbor)
        else if gScore< openSet.get(neighbor).gScore
            openSet.replace(openSet.get(neighbor), neighbor)
```

Heuristic Features

Admissible: For every position (x,y), Heuristic(x,y) <= the actual cost of getting to the goal.

Consistent: For every two positions (x,y) and (x,y) abs(Heuristic (x,y)-Heuristic(x,y)) must be <= the actual cost of traversing between the two positions.

In Game AI we only care about results, but having these two features can still be helpful. **A* will always find the optimal path if both are true.**

A* Weaknesses

- At worst you can expand/open n^2 nodes where n is the number of locations in the world. This is especially bad in grids!

- There are stronger forms of pathfinding (there's a B and B* search!) but they're much less common in games.
  - Professors Nathan Sturtevant and Michael Buro are world experts at pathfinding.

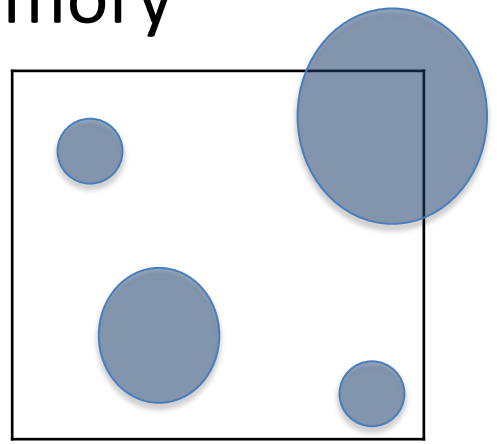
Assignment 2 Review

Practice Quiz 2

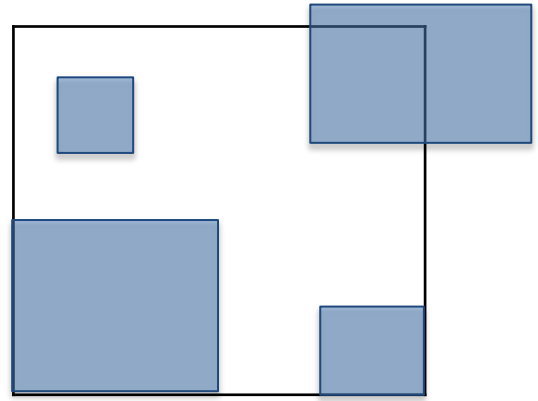
https://forms.gle/hFv86ititVJe1Vyj8

https://tinyurl.com/guz-pq6

1. Which of the following has the largest runtime memory cost (typically)?
   a) Grid (generated)
   b) Nav mesh (generated
   c) Path network (authored)
   d) These are all roughly equivalent



2. Let's say you wanted to use one spatial representation for both environment A and B (on the right). Which one would you pick? Why?



3. What particular implementation details would be important for your answer to (2)? (e.g. what would you have to do to make sure it'd work for both?)
4. What steering approaches would you use for a gang of chaotic motorcyclist enemies in a video game?

Answers

1. A
2. Hand-authored path network or generated Nav Mesh would both be fine answers, you could even get away with grids. Depends on the reasoning of the “why”
3. Many answers. Path network example: rounded edges. Nav Mesh example: obstacles given polygonal bounding boxes.
4. Something like boids for group dynamics (full marks for mentioning separation, cohesion, and alignment). But you don’t just want boids as it wouldn’t look chaotic, so throw in some intense kinematic wander as well.