NavMeshes

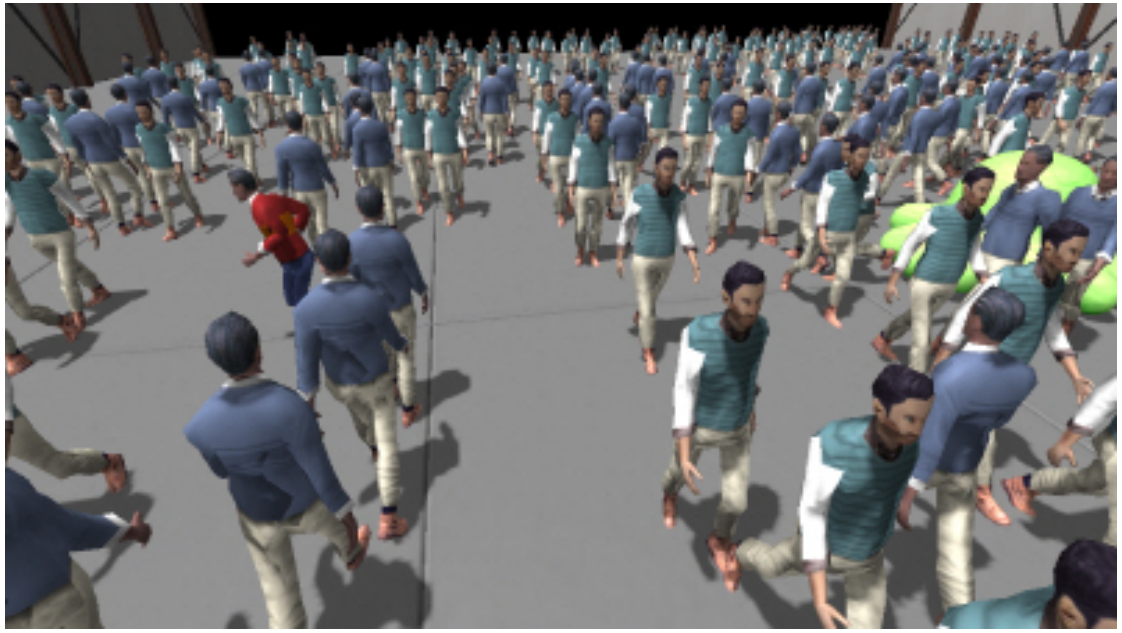
CMSC425.01 Fall 2019

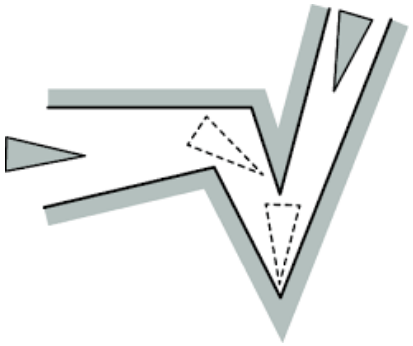
Today’s question

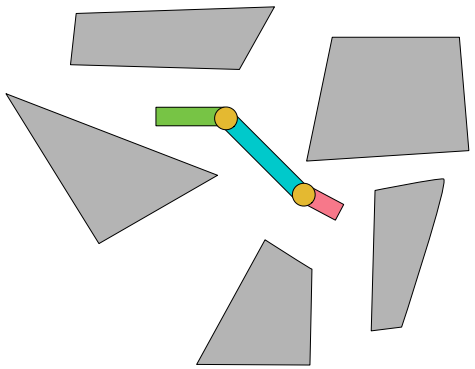
How NPCs move around: NavMeshes

Navigation problems

- Navigating from place to place
- Dense crowd navigation
- Coordinated team movement
- Pursuit
- Moving complex/articulated shape
  - Piano movers problem(rigid)
  - Skeleton (articulated)







@UMD:
Dinesh and Ming

- Dense crowd simulations
- How to move many agents naturally

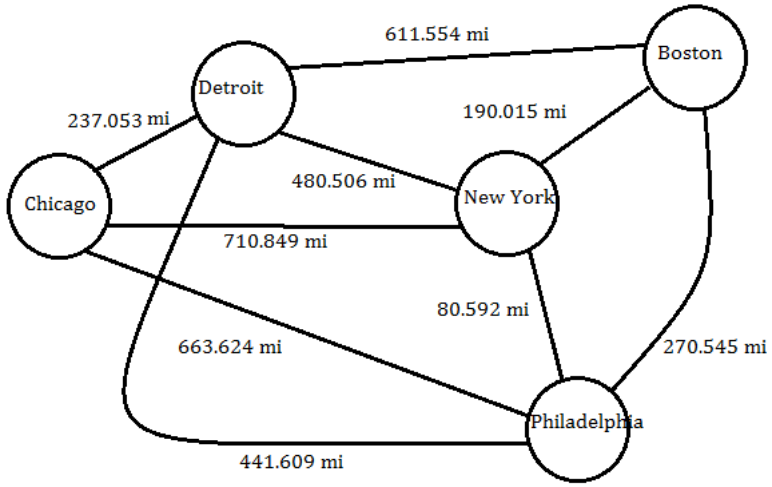
- Video

Navigation Version 1: Use a map

Create graph structure

Use Dijkstra's algorithm

Shortest path



Assumption: World is rigid, limited lanes for movement

What is "shortest" path?

- Distance?

What is "shortest" path?

- Distance?
- Speed?
- Energy cost?
- Exposure to enemy?

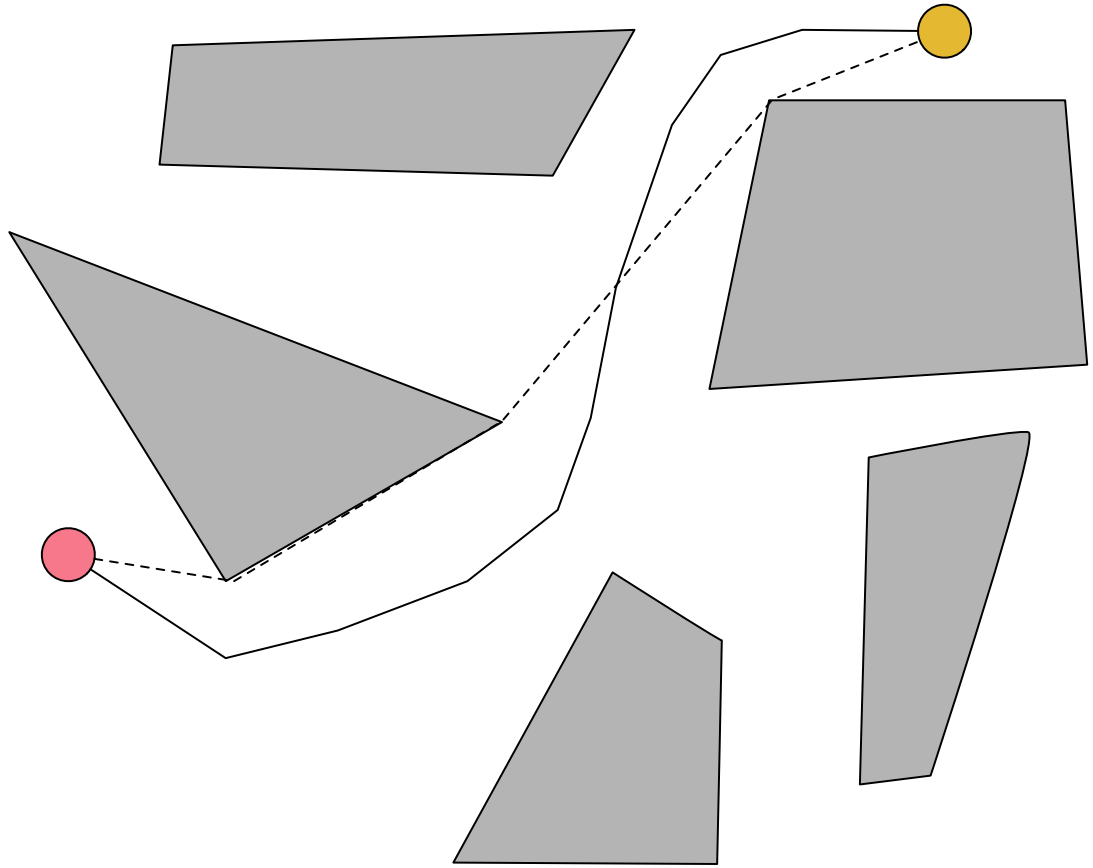
- Hex map
  - 6 directions
  - Terrain types => speed costs

Navigation Version 2: Open Terrain

- Mix of obstacles and open spaces: "free space geometry"
- More options for direction
- How pick path?

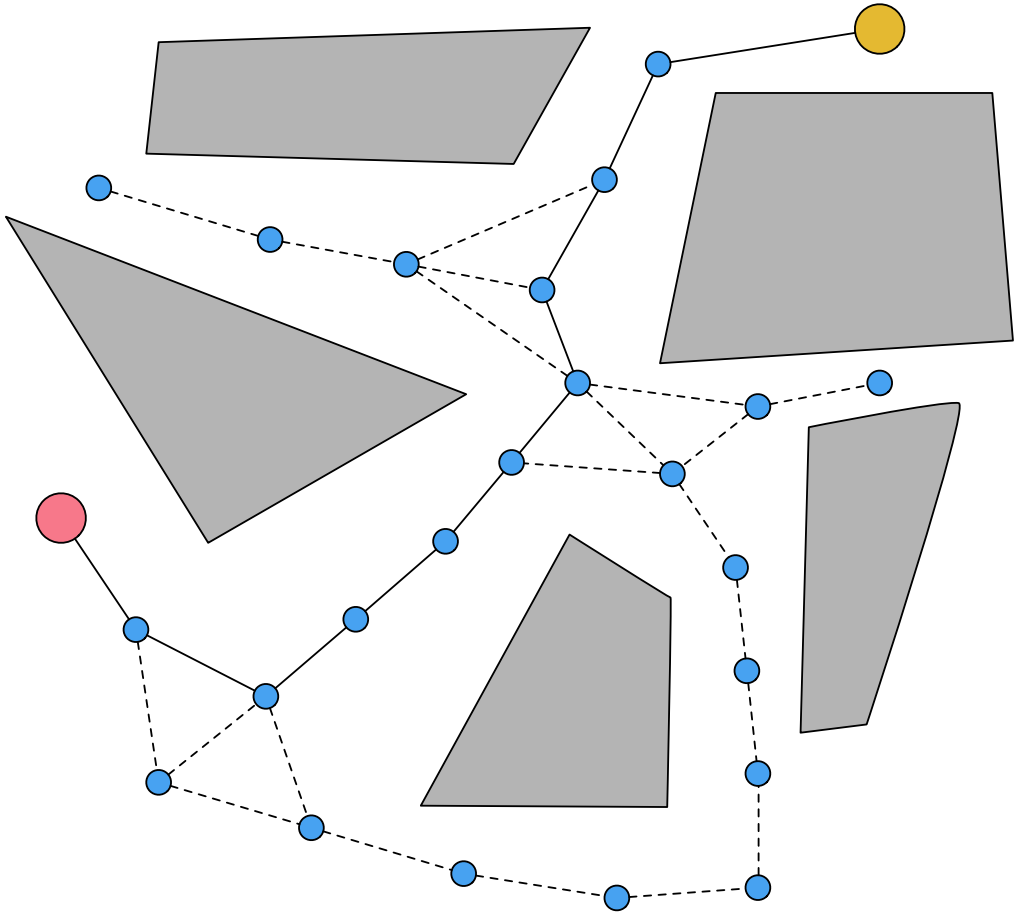
How pick best path?

- Shortest may not be most natural (dashed line)
- Need way to detect natural corridors



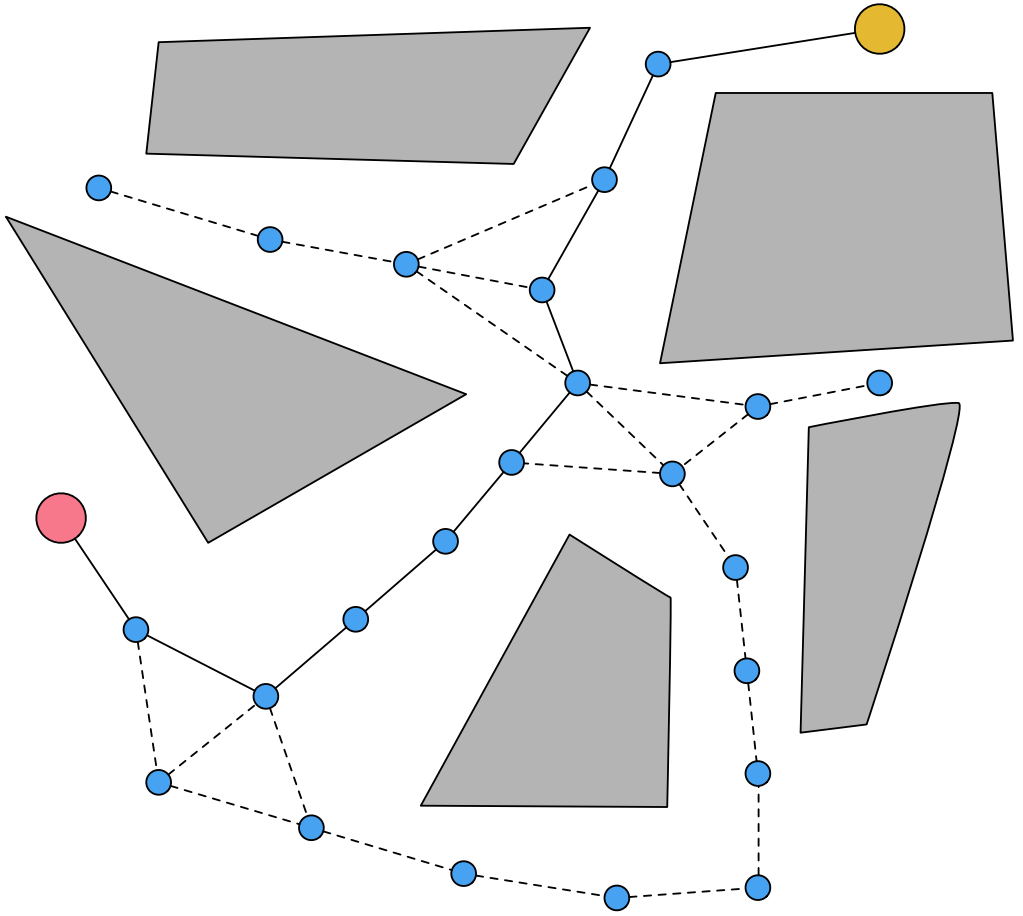
Solution 1: Waypoint roadmap

- Preprocess space into graph of waypoints
- Place waypoints along natural corridors



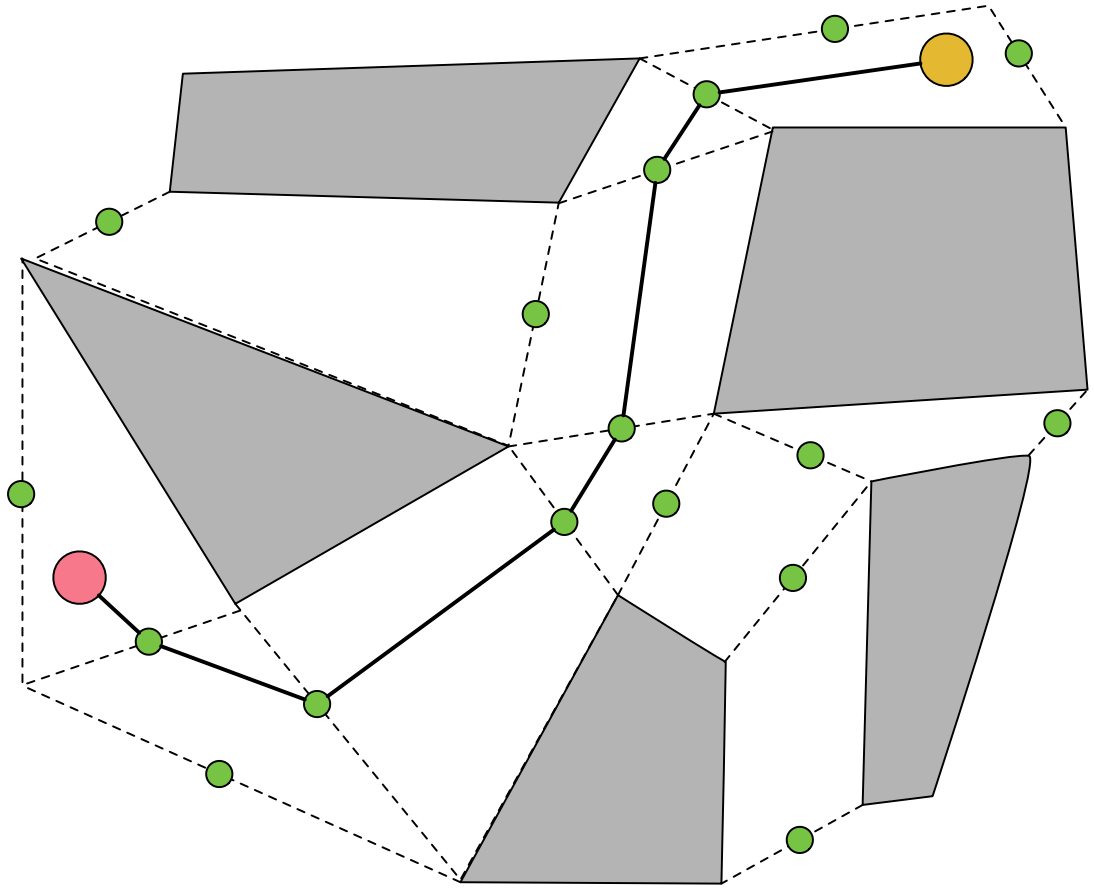
Solution 1: Waypoint roadmap

- Preprocess space into graph of waypoints
- Place waypoints along natural corridors
- Drawbacks
  - Could need lots of way points
  - Harder to plan for coordinated team movement



Solution 2: Navmesh

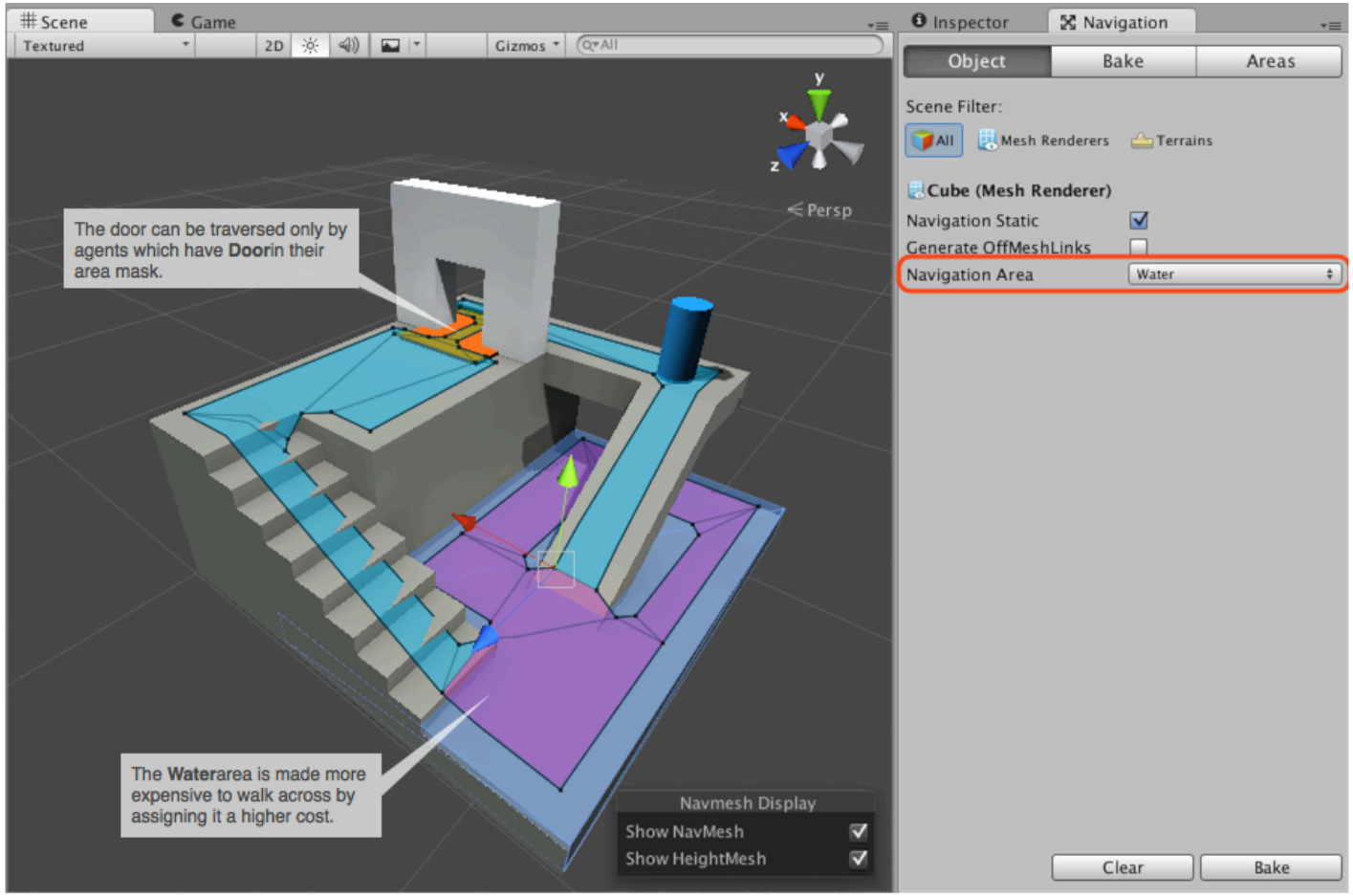
- Preprocess space into mesh of free areas as polygons
- Plan movement between polygons
  - Between edges
  - Between centers
- Multiple level graph search
  - First between regions
  - Then pick entry and exit points



Navmeshes can be

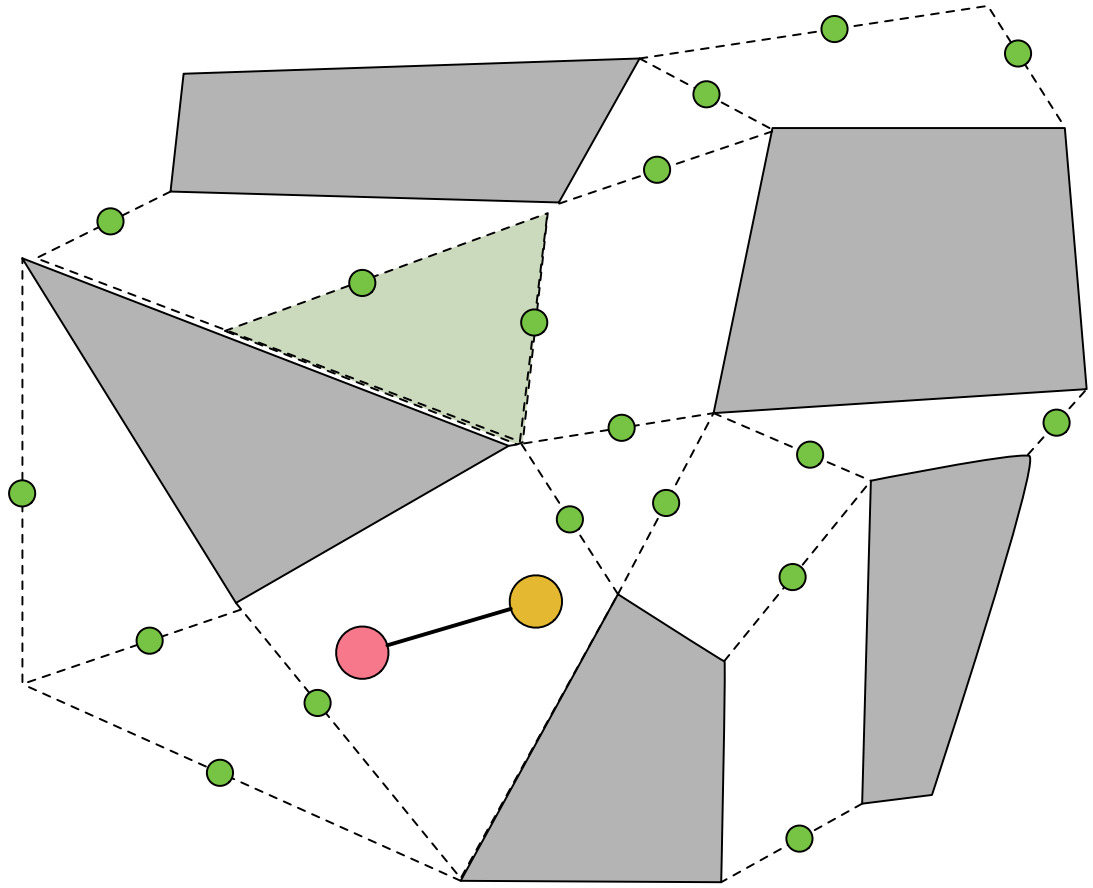
- Labeled with different terrain types
- Set with "gates"

- 2D manifolds which are topologically complex (eg, non-planar)



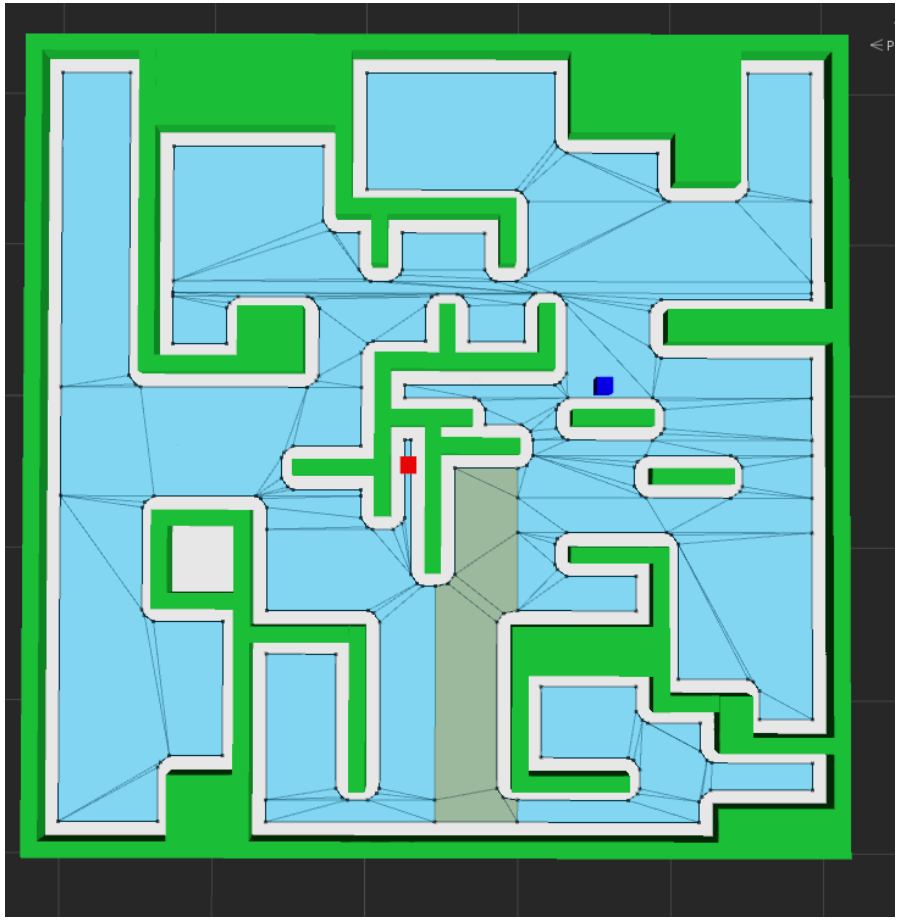
Using Navmesh

- Inside region use direct path
- Regions (and subregions) can be labeled with different terrain types and costs



Creating Navmeshes from terrain

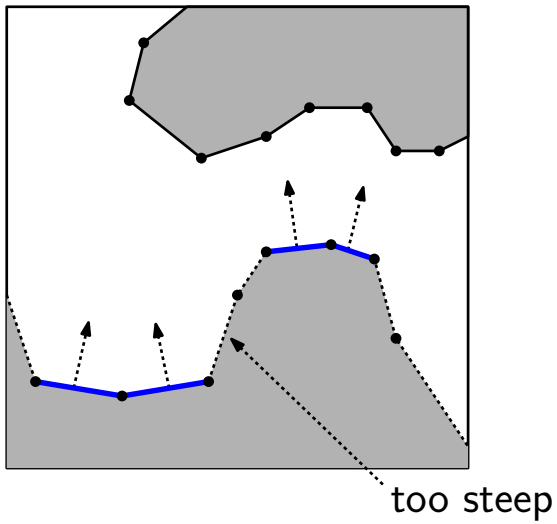
- Step 1: Find walkable surfaces
  - As large polygon "map"
- Step 2: Simplify boundaries
  - Simplify polygon "map"
- Step 3: Triangulate "map"
  - Cover with set of triangles



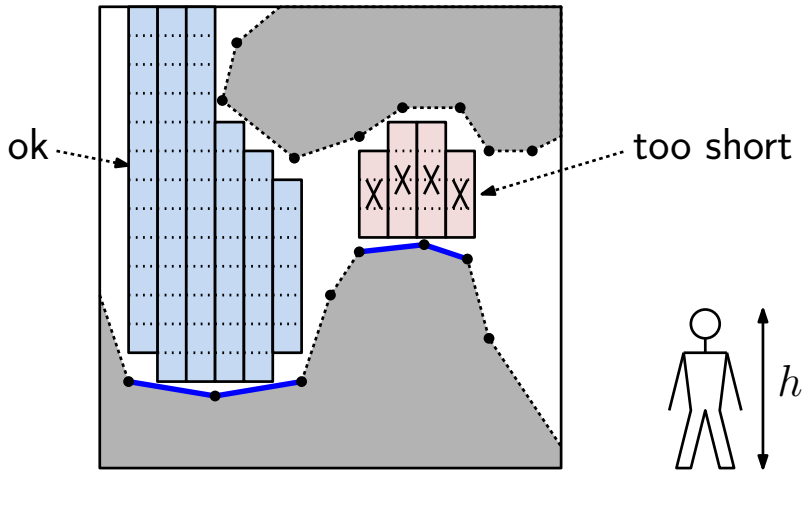
Creating Navmeshes from terrain

- Step 1: Find walkable surfaces based on agent height, width, slope ability

- Variation on piano movers problem



(a)



(b)

Creating Navmeshes from terrain

- Step 2: Simplify boundaries
  - Simplify polygon "map"
- Recursive refinement of straight line

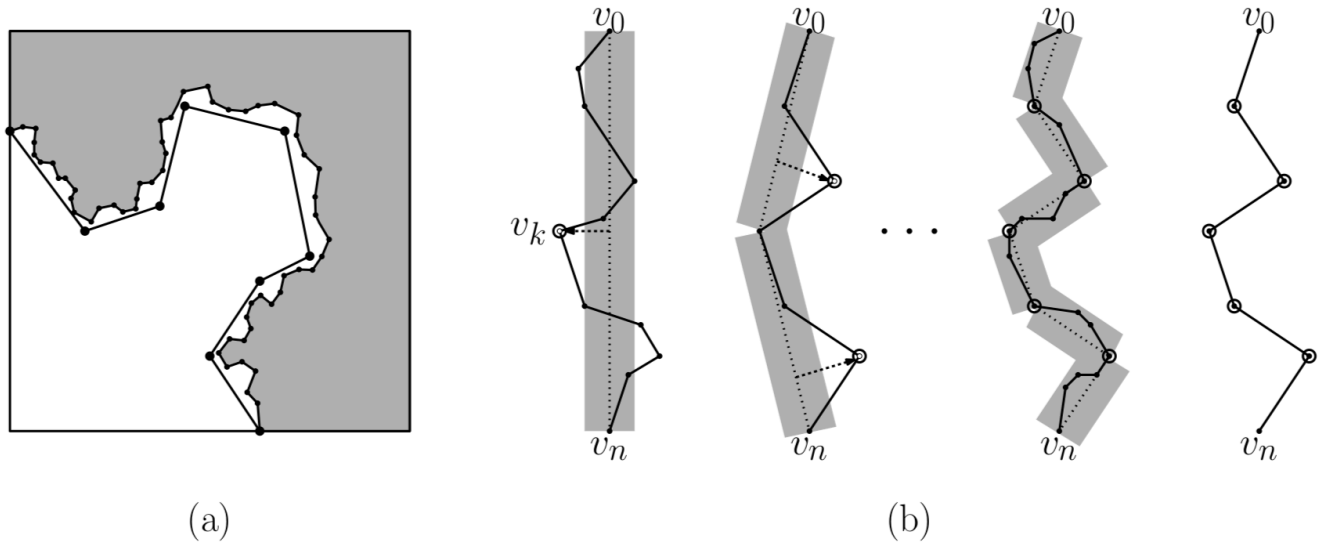
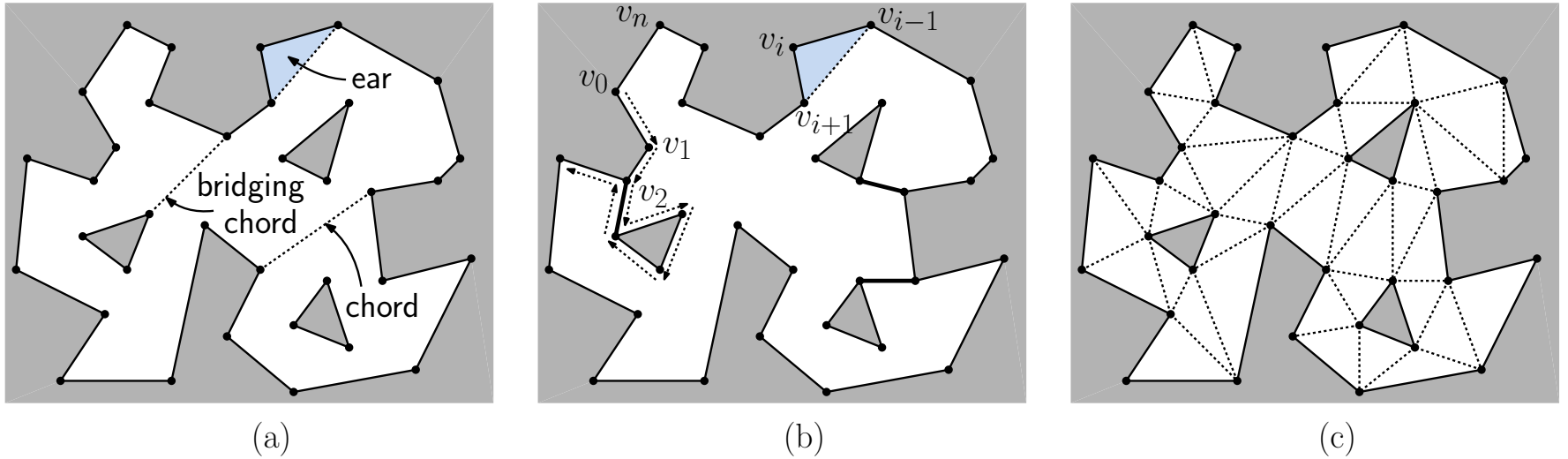


Fig. 3: The Ramer-Douglas-Peucker Algorithm.

Creating Navmeshes from terrain

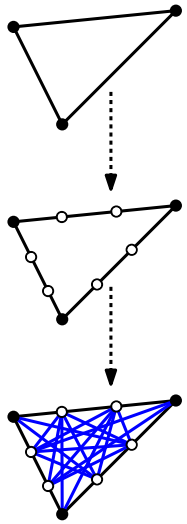
- Step 3: Triangulate "map"
  - Cover with set of triangles

- Bridge holes
- Cut ears (!)

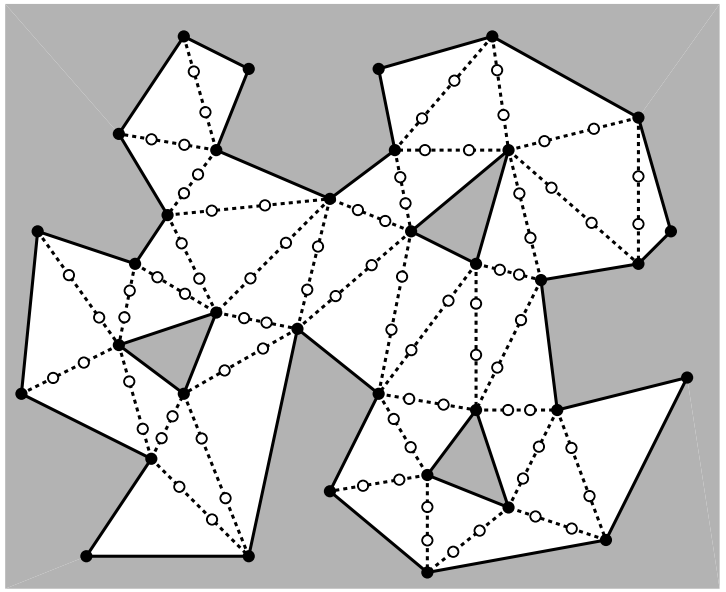


Use Navmesh: find path

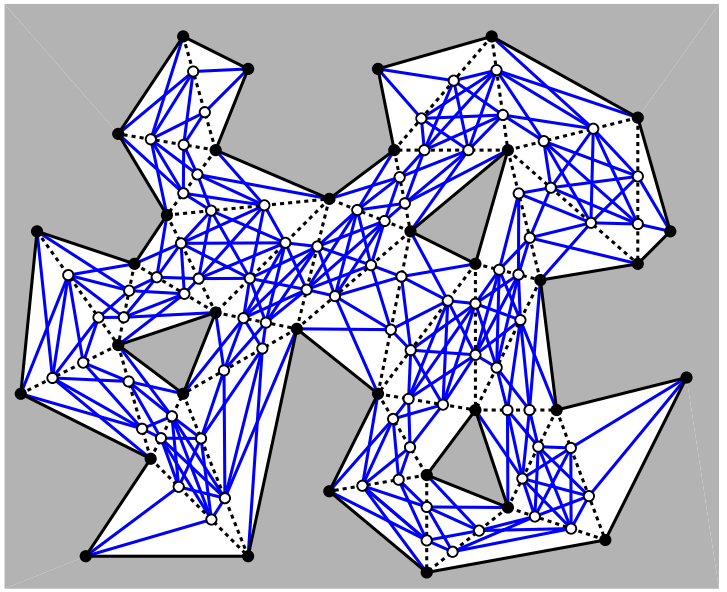
- Discretize by adding points
- Find shortest path



(a)



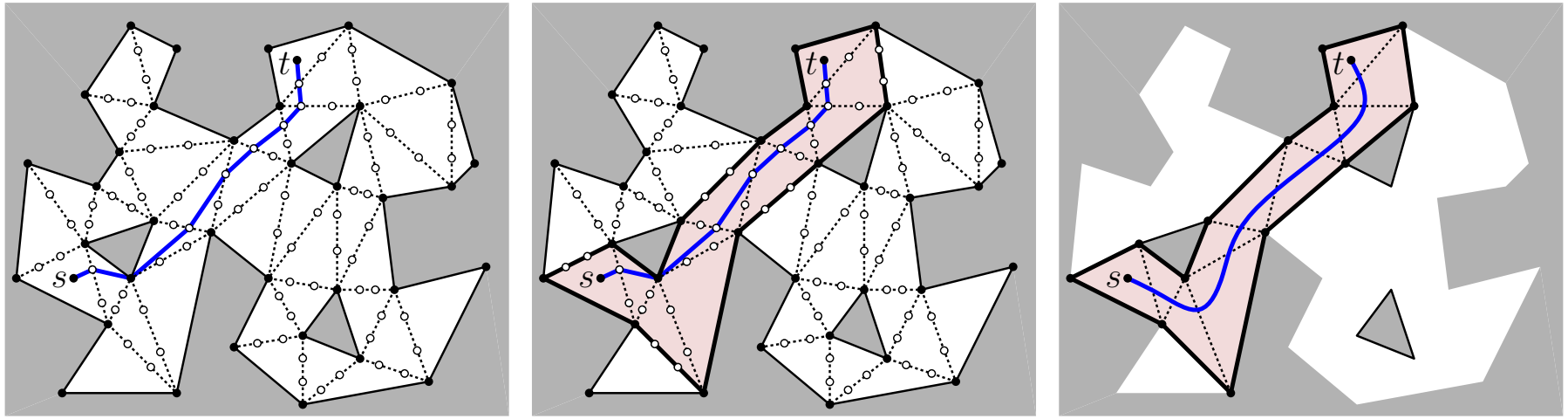
(b)



(c)

Use Navmesh: refine

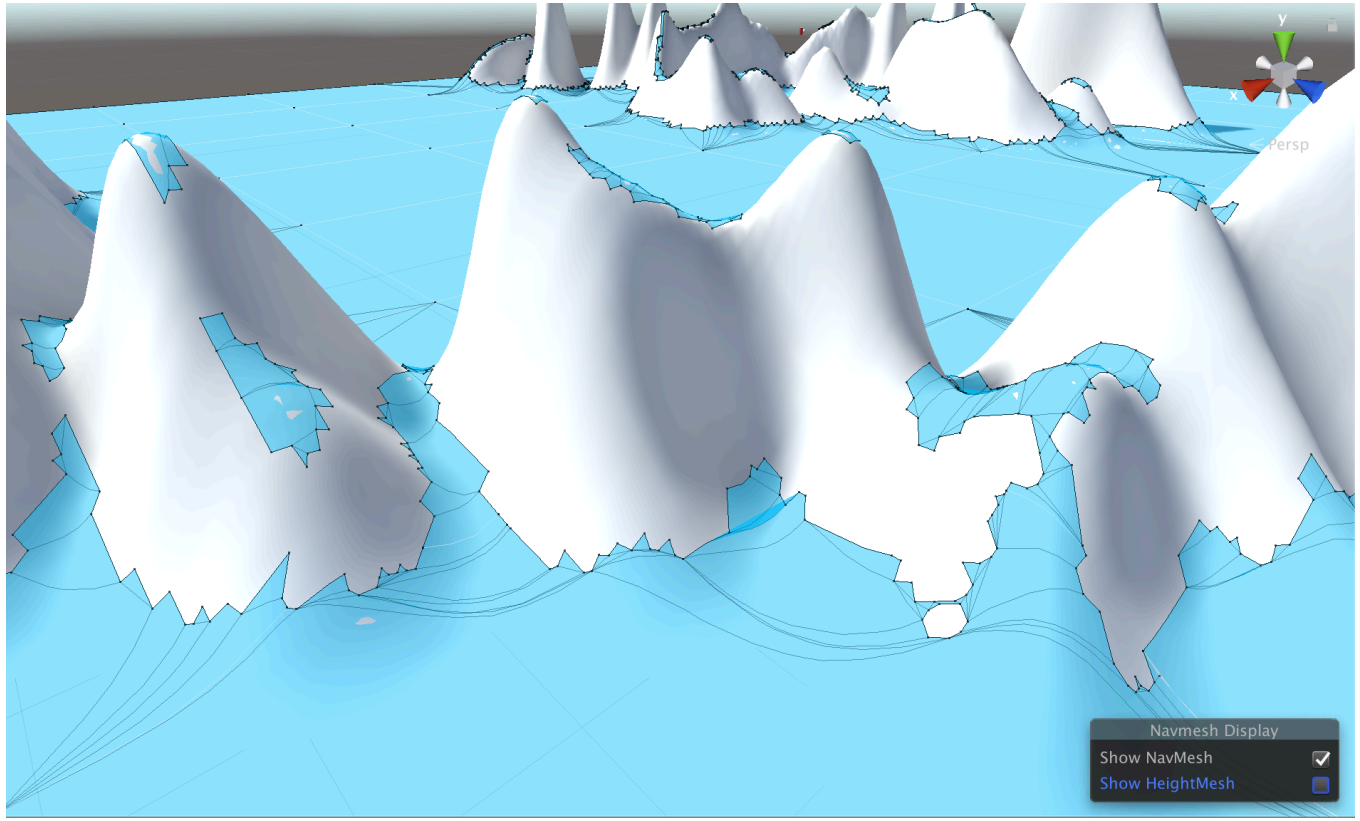
- Smooth and clean path



(a) (b) (c)

Navmeshes in Unity

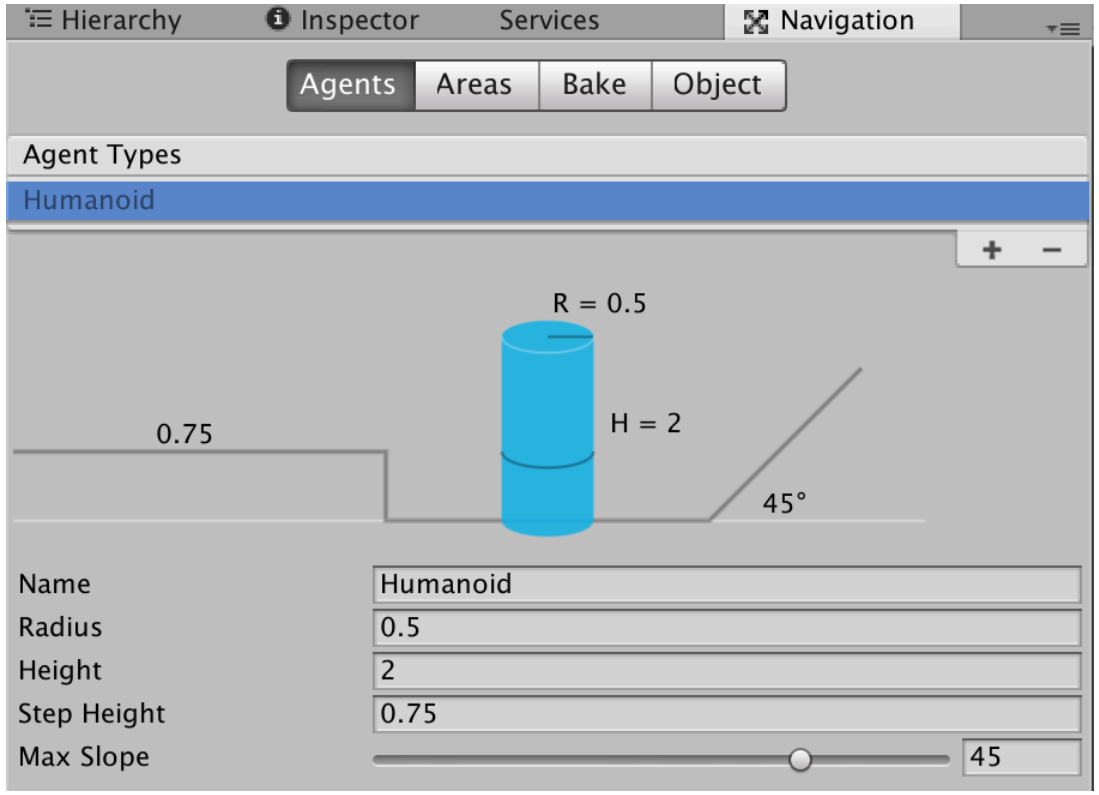
- Create terrain
  - Terrain editor
- Set agent navigation properties
  - Height, width of agent
  - Slope capability
- Bake Navmesh
  - Finds navigable regions
  - Creates mesh



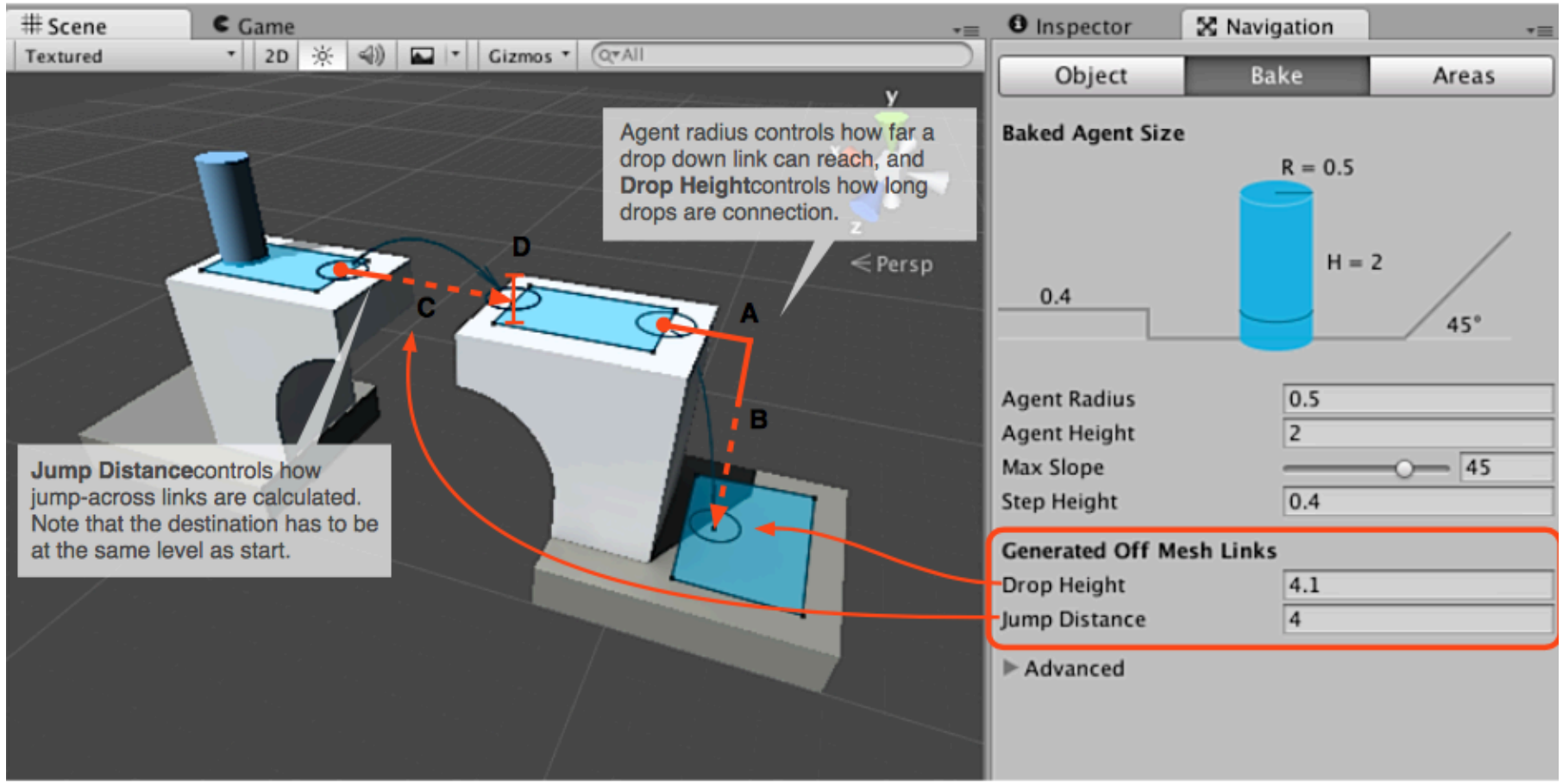
Agent properties

- Radius, Height
- Step height
- Max slope

- Can have multiple agent types



Off mesh links



Question

- How does this all relate to colliders?

Readings

- Look at Unity manual and tutorials
- Terrain
  - https://docs.unity3d.com/Manual/terrain-UsingTerrains.html
- NavMesh
  - https://unity3d.com/learn/tutorials/topics/navigation/navmesh-agent
  - https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html
- Animation
  - https://unity3d.com/learn/tutorials/topics/animation/animate-anything-mecanim
  - https://unity3d.com/learn/tutorials/s/animation