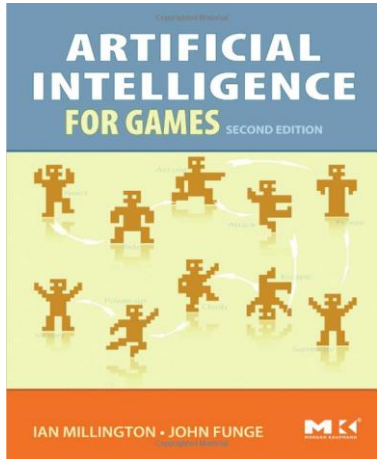
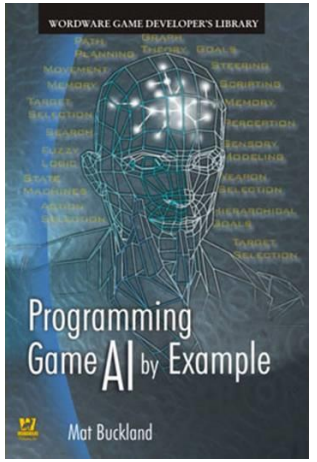


Disclaimer: I use these notes as a guide rather than a comprehensive coverage of the topic. They are neither a substitute for attending the lectures nor for reading the assigned material.



"I may not have gone where I intended to go, but I think I have ended up where I needed to be." –Douglas Adams

"All you need is the plan, the road map, and the courage to press on to your destination." –Earl Nightingale

Announcements

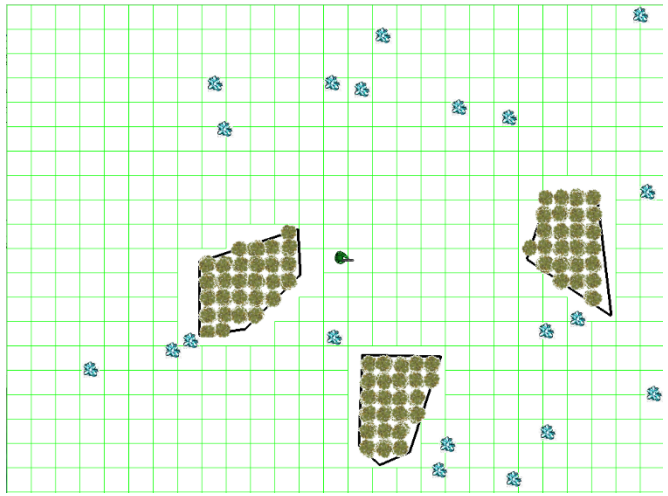
- HW1 due Sunday night, Sept 1 @ 11:55pm
- HW2 is much more challenging than HW1. Start early!
- Game engine & HWs – piazza only; please no posting on any public forum (public git, stackoverflow, etc)
- Office hours
  - https://calendar.google.com/calendar?cid=dGozaWc2ZGh1cTg0OG44aWQ3cGo5bDdlaG9AZ3JvdXAuY2FsZW5kYXIuZ29vZ2xlLmNvbQ
- Special lectures
- Labor day.

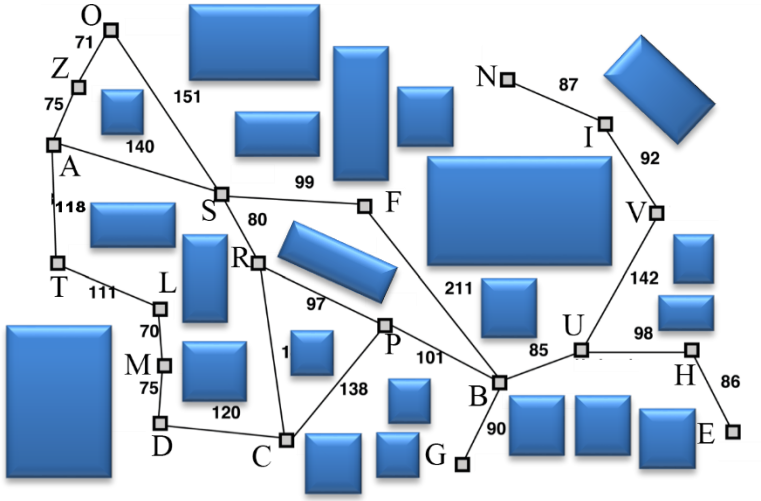
Grid Generation Hints

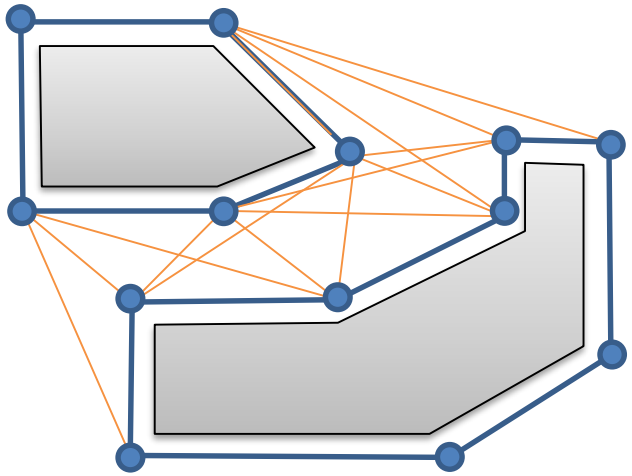
- Verify no world line goes through grid lines (rayTraceWorld)
- Verify no obstacle point within grid cell? Grid corner in obstacle?
  - e.g. pointInside
- Please check the following sections
  - “Miscellaneous utility functions”
  - “Hints”

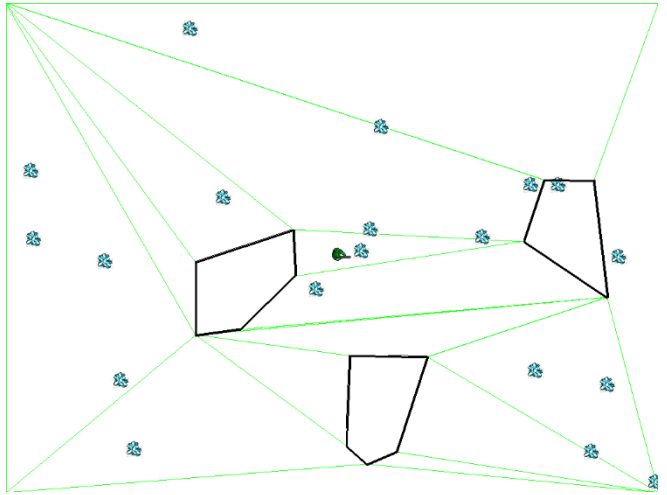
PREVIOUSLY ON…

Modelling and Navigating the Game World









N-1: Grids, Path Networks

1. What's the intuition behind iterative deepening?
2. What are some pros/cons of grid navigation?
3. What are some benefits of path networks?
4. Cons of path networks?
5. What is the flood fill algorithm?
6. What is a simple approach to using path navigation nodes?
7. What is a navigation table?
8. How does the expanded geometry model work? Does it work with map gen features?
9. What are pros and cons of expanded geometry?

Graphs, Search, & Path Planning
Continued: Models of world for path planning

2019-08-28;

See also: Buckland Ch 5 & 8,

Millington & Funge Ch 4

Path finding models

1. **Tile-based graph – “grid navigation”**
   - Simplest topography
   - assume static obstacles
   - imaginary latice of cells superimposed over an environment such that an agent can be in one cell at a time.
   - Moving in a grid is relatively straightforward: from any cell, an agent can traverse to any of its four (or eight) neighboring cells
2. Path Networks / Points of Visibility NavGraph
3. Expanded Geometry
4. NavMesh

Path finding models

1. Tile-based graph – “grid navigation”
2. **Path Networks / Points of Visibility NavGraph**
   - does not require the agent to be at one of the path nodes at all times. The agent can be at any point in the terrain.
   - When the agent needs to move to a different location and an obstacle is in the way, the agent can move to the nearest path node accessible by straight-line movement and then find a path through the edges of the path network to another path node near to the desired destination.
3. Expanded Geometry
4. NavMesh

Path finding models

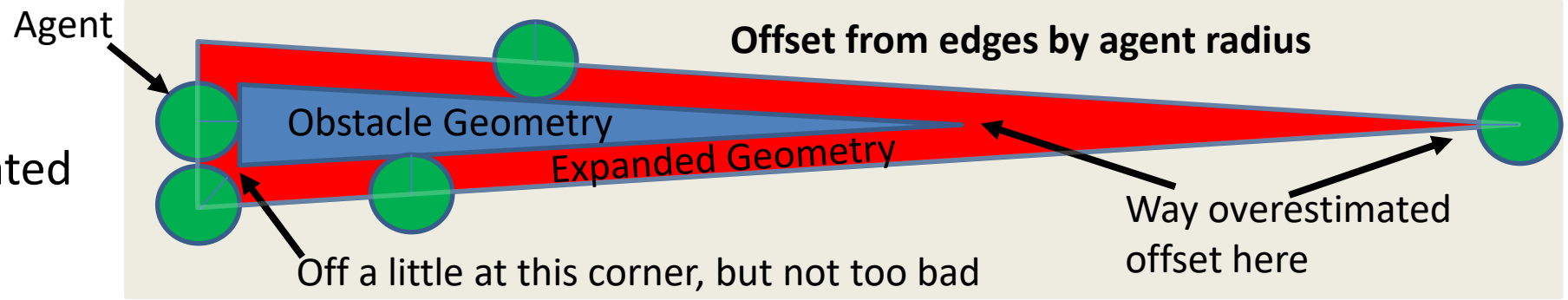
1. Tile-based graph – “grid navigation”
2. Path Networks / Points of Visibility NavGraph
3. **Expanded Geometry**
   - Discretization of space can be smaller
   - 2 tier nav: Continuous, non-grid movement in local area
   - Can work with auto map generation
   - Can plan nicely with “steering behaviors”
4. NavMesh

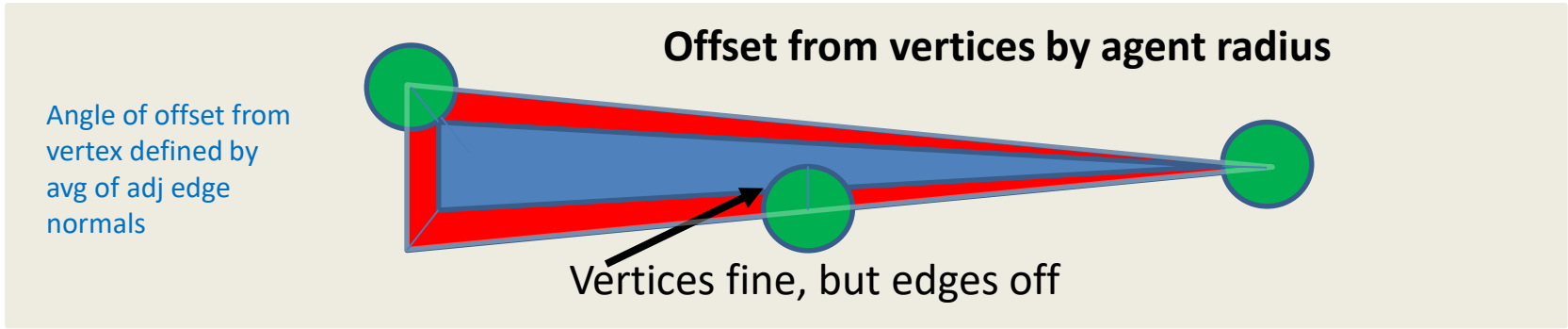
Model 3: Expanded Geometry

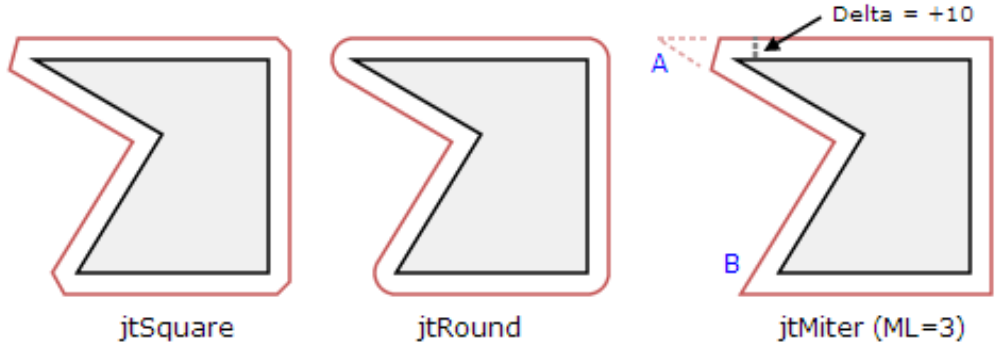
- Automatic, and no wall bumping.
- Also a two-tiered navigation system
  - Local, continuous
  - Remote
- Automatically expand boundaries of obstacles ($\Delta \geq$ agent_radius)
- Add vertices as nodes
- Test line of sight for all vertices ($O(n^2)$)
- Add edges where (v_1, v_2) == true

Expanded Geometry: Corner “Gotchas”

- Expanding edges
  - can result in overestimated offsets
- Expanding vertices
  - can result in underestimated offsets
- Equidistant expansion
  - introduces non linear curvature (curved at corner offsets)
- Squaring off/selective mitering is compromise to avoid curves







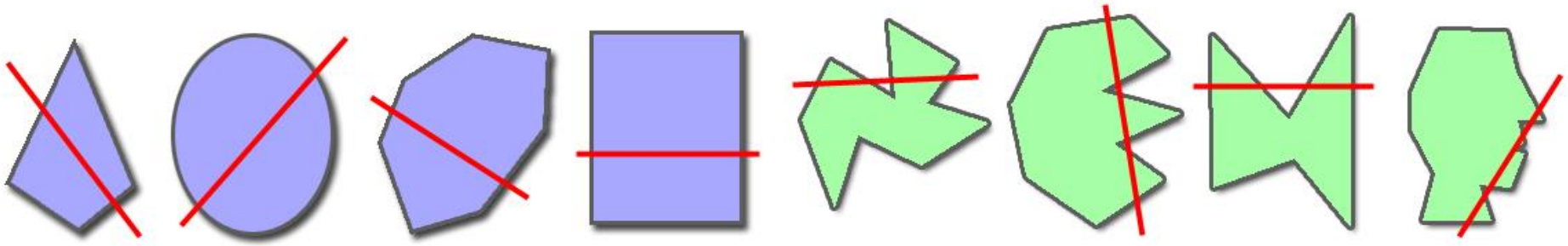
Credit: Jeffrey Wilson

Path finding models

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

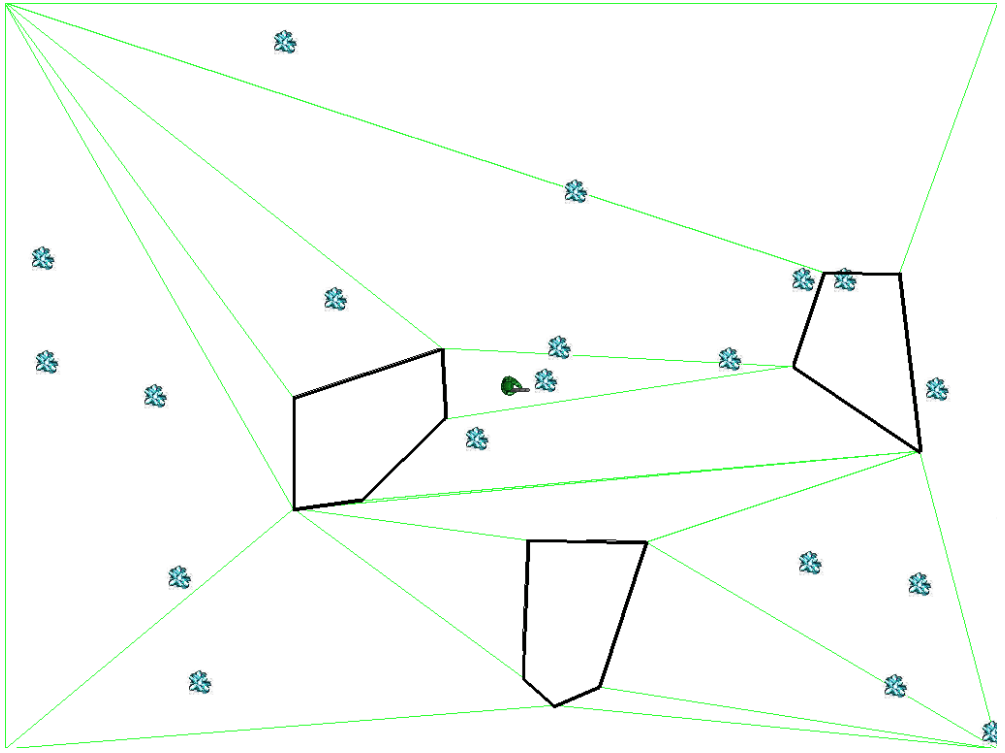
M4: NavMesh

- Win: compact rep, fast search, auto create
- Each node (list of edges) is a convex polygon
- Convex = Any point within the polygon is unobstructed from any other
- Can be generated from the polygons used to define a map



Generating the Mesh

- Lots of algorithms
- Optimal:
  - Fewest polygons, smallest discretization possible
  - NP-complete
- Greedy:
  - Find triangles guarantees convex
  - Merge triangles



Generating the Mesh: Greedy/Simple Approach

For point a in world points:

    For point b in world points:

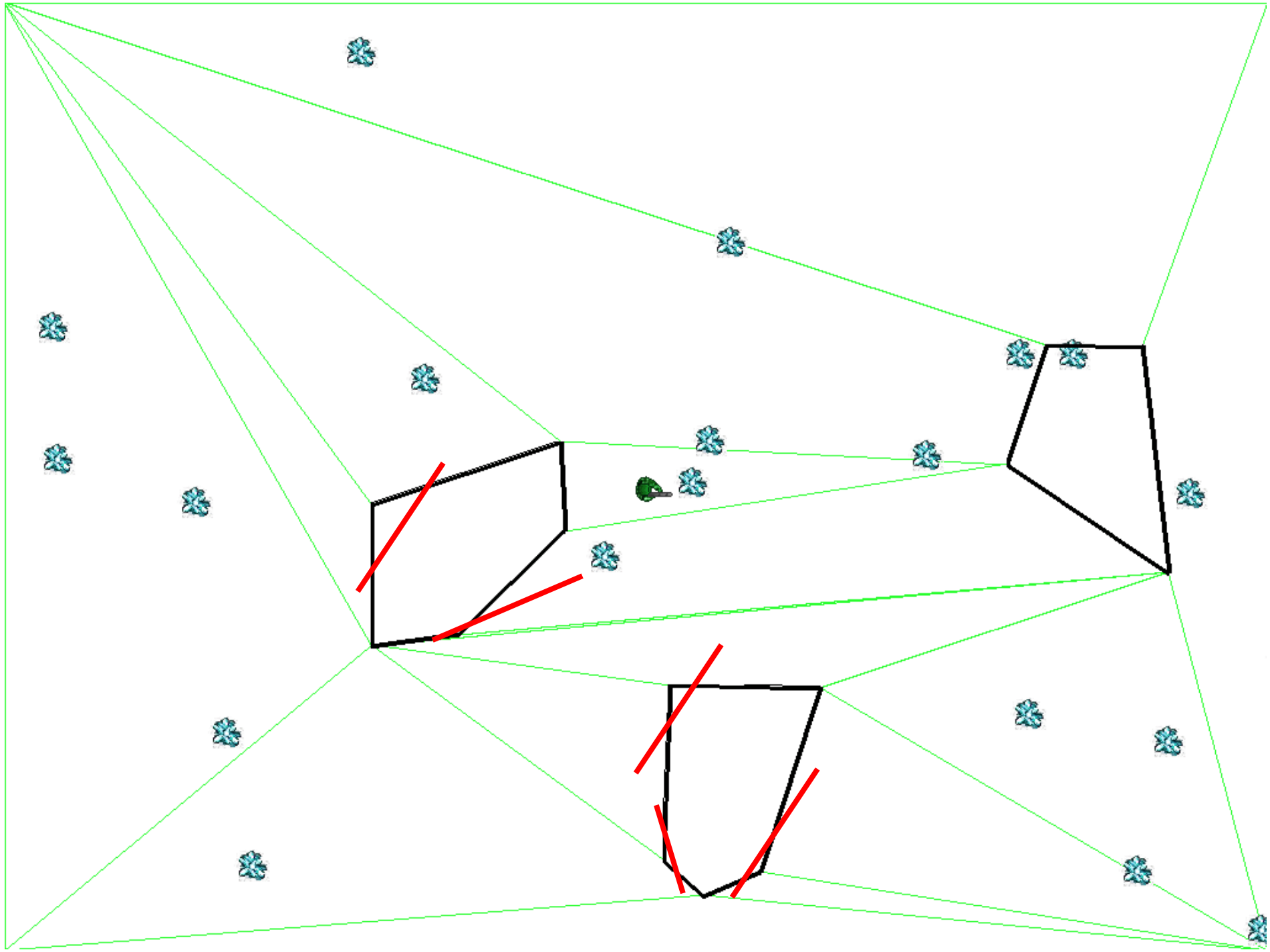
        For point c in world points:

            if (it is a valid triangle) and !exists:

                add triangle to mesh

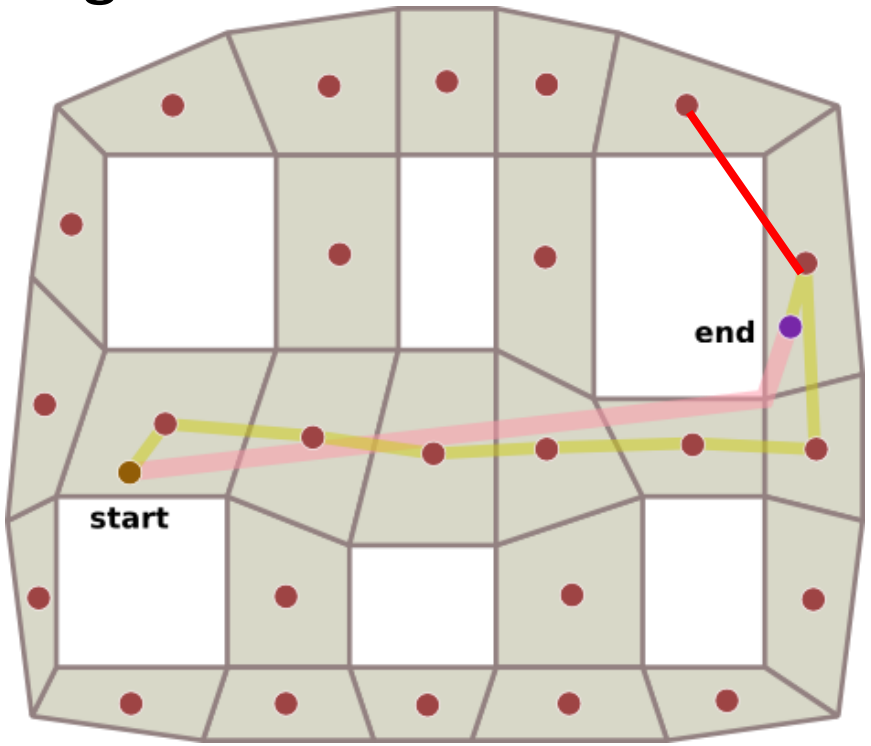
Iterate through triangles to merge to quads

Iterate through quads to merge to 5-sided shapes...



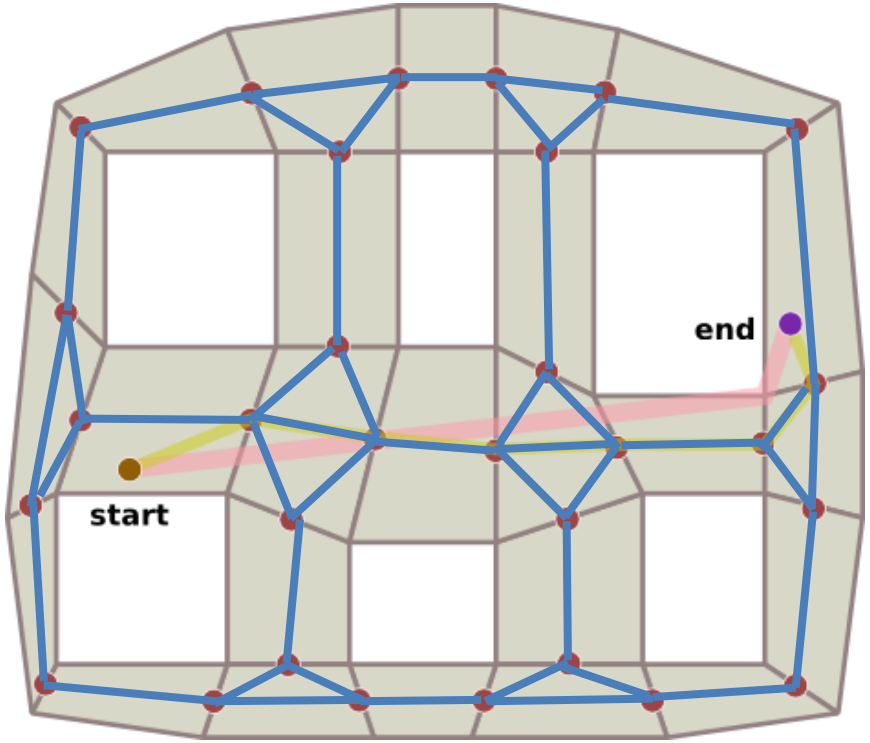
Nav Meshes + Waypoints

- Put a waypoint in center of each nav mesh
  - It's important to get a good set of nav meshes



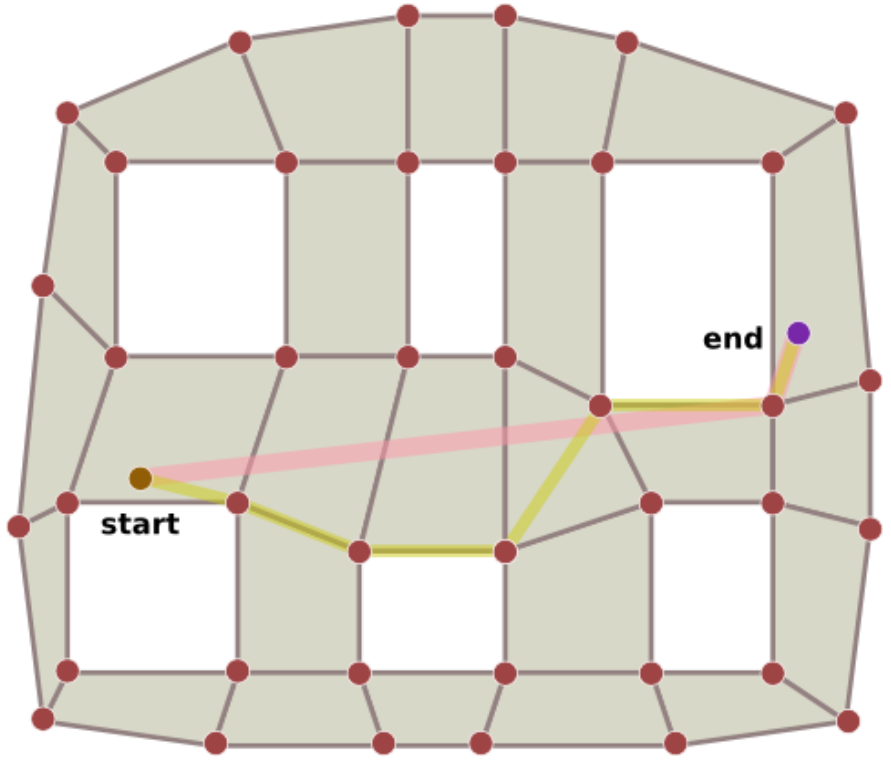
Nav Meshes + Waypoints

- Put a waypoint at adjoining edges



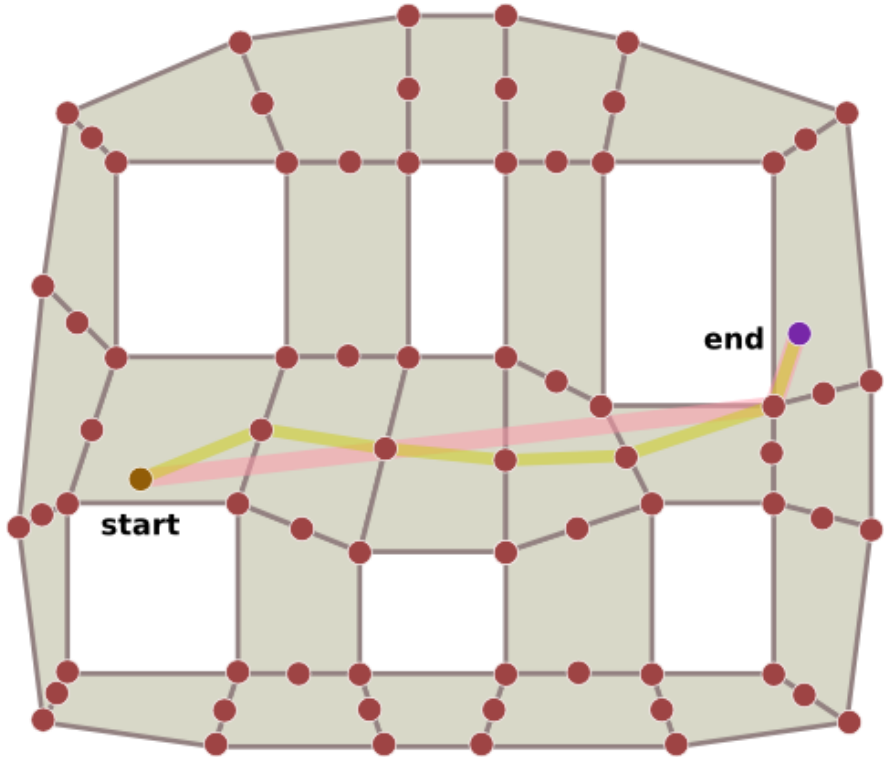
Nav Meshes + Waypoints

- Put a waypoint at corners of obstacles



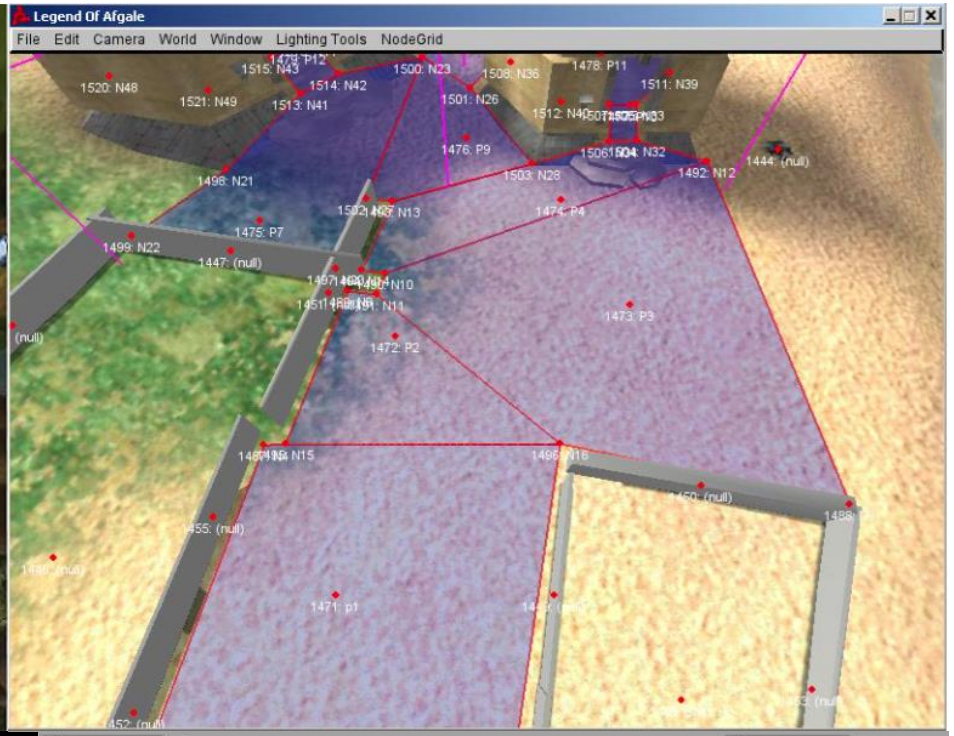
Nav Meshes + Waypoints

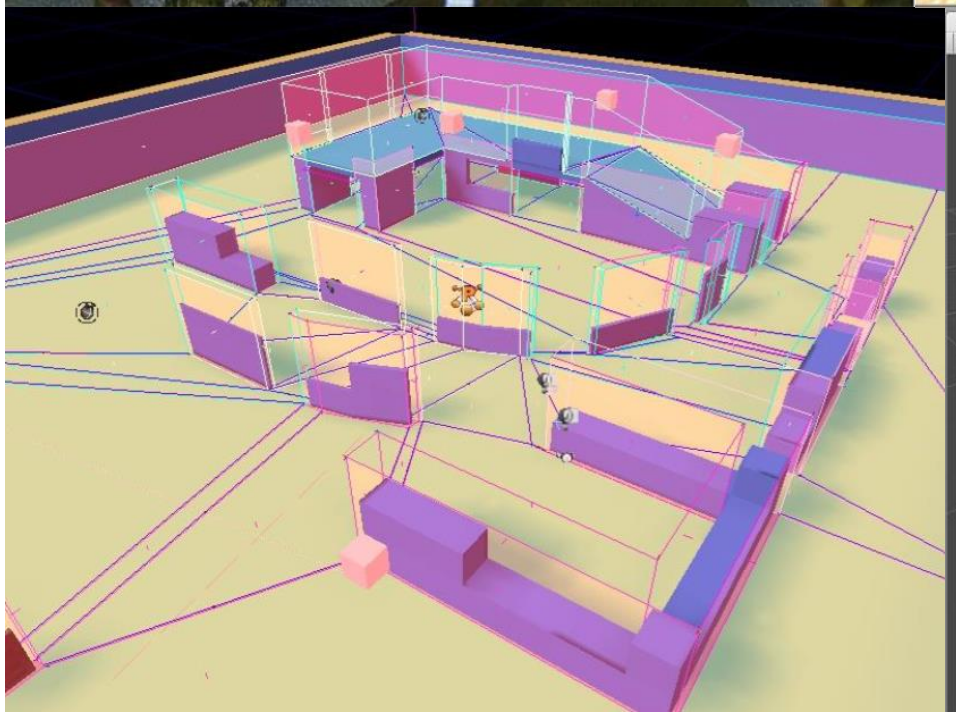
- Put a waypoint at edges and corners

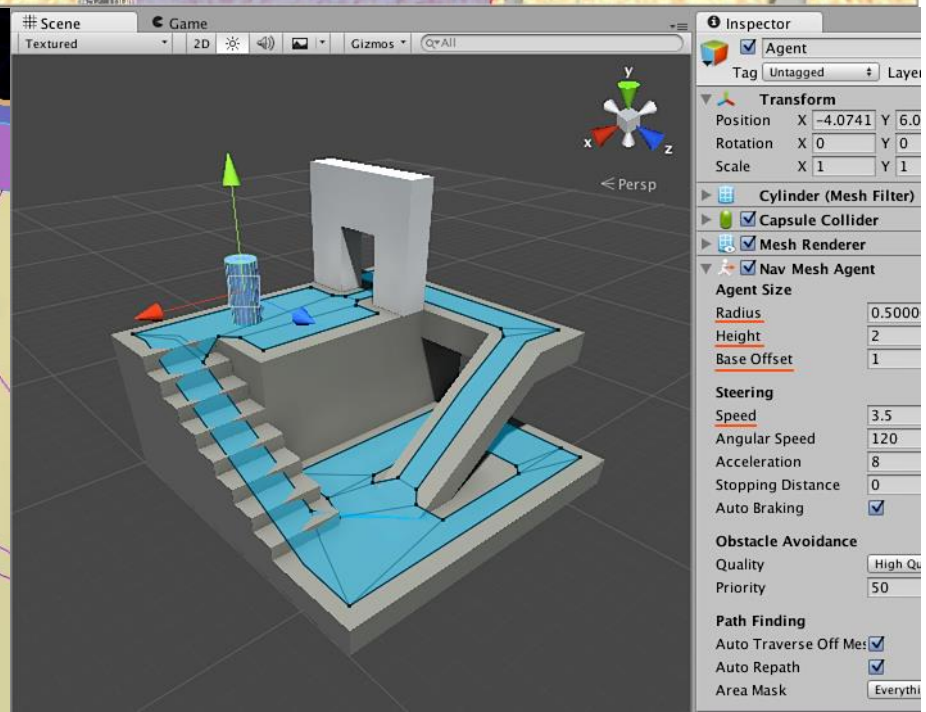


See

- ~12:00 https://www.gdcvault.com/play/1024912/Beyond-Killzone-Creating-New-AI
  - Navmesh, waypoints, string pulling, a*, Bezier path smoothing, steering behaviors, polygon vs point paths
  - http://digestingduck.blogspot.com/2010/03/simple-stupid-funnel-algorithm.html
    - https://www.gamedev.net/forums/topic/669843-the-simple-funnel-algorithm-pre-visited/
  - http://jeffe.cs.illinois.edu/teaching/comptop/2009/notes/shortest-homotopic-paths.pdf
- ~5:00, ~20:30: Flood fill, navmesh, blackboards, hash, cheating, crowdsource/breadcrumb/clustering/filtering: https://youtu.be/iVBCBcEANBc

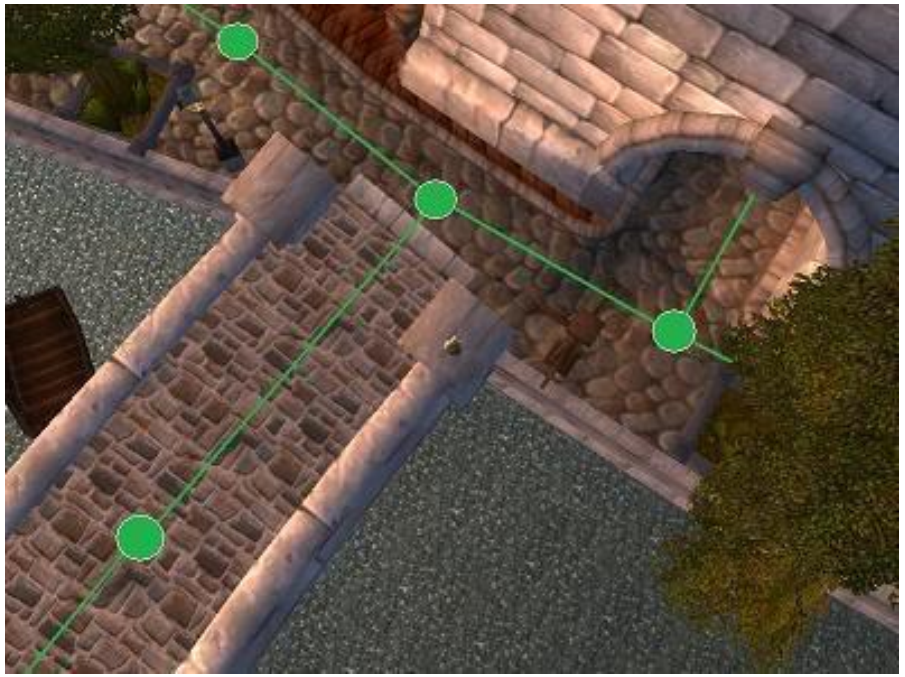

Legend Of Afgale
File Edit Camera World Window Lighting Tools NodeGrid




Scene
Game
Textured
2D
Gizmos
Persp
Inspector
Agent
Tag
Untagged
Transform
Position
Rotation
Scale
Cylinder (Mesh Filter)
Capsule Collider
Mesh Renderer
Nav Mesh Agent
Agent Size
Radius
Height
Base Offset
Steering
Speed
Angular Speed
Acceleration
Stopping Distance
Auto Braking
Obstacle Avoidance
Quality
Priority
Path Finding
Auto Traverse Off Me
Auto Repath
Area Mask

Waypoints vs. NavMesh

5 Reasons why waypoints fall short

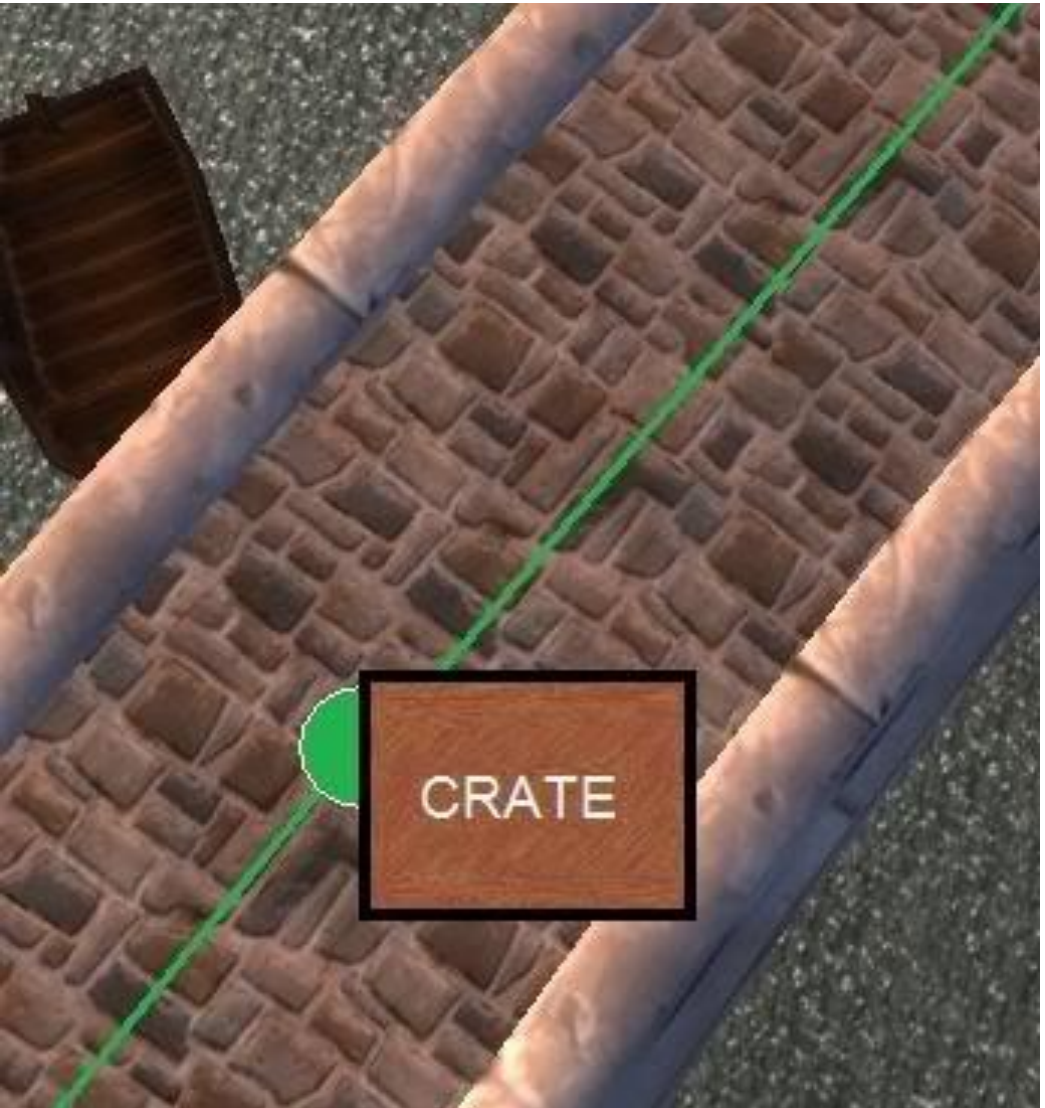


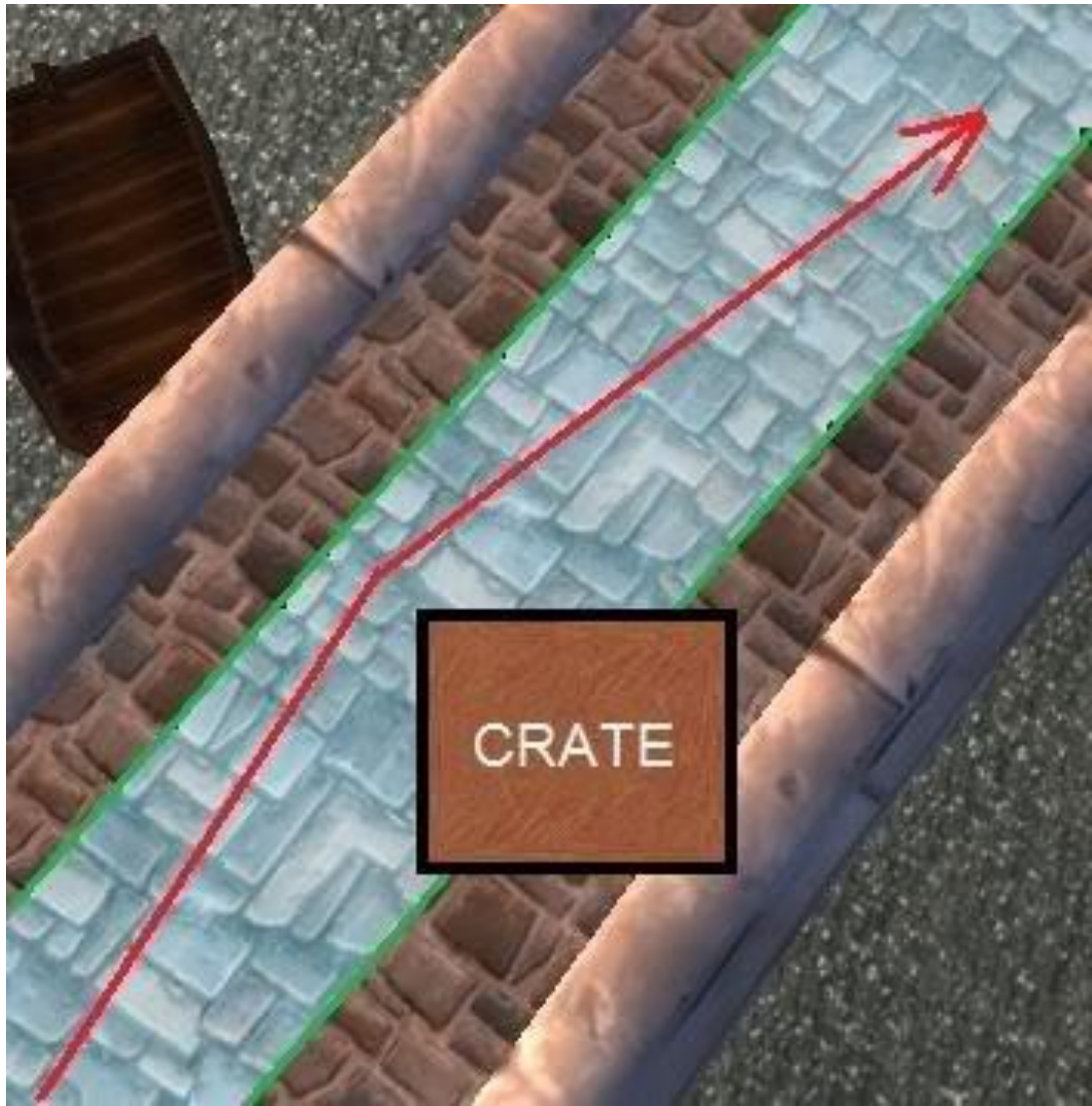
1) Some worlds need
WAY too many to match freedom of nav mesh

2) Waypoints make NPCs zig-zag

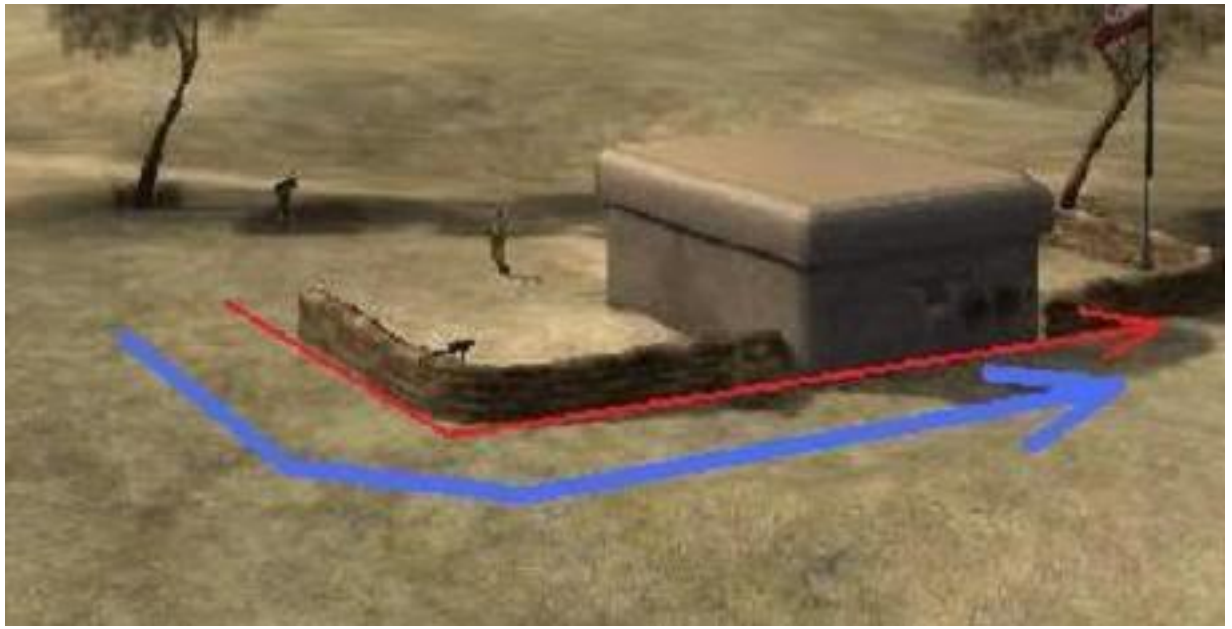
NavMesh Smoothing

3) Waypoints don’t allow for path correction: Alterable/Generated Content

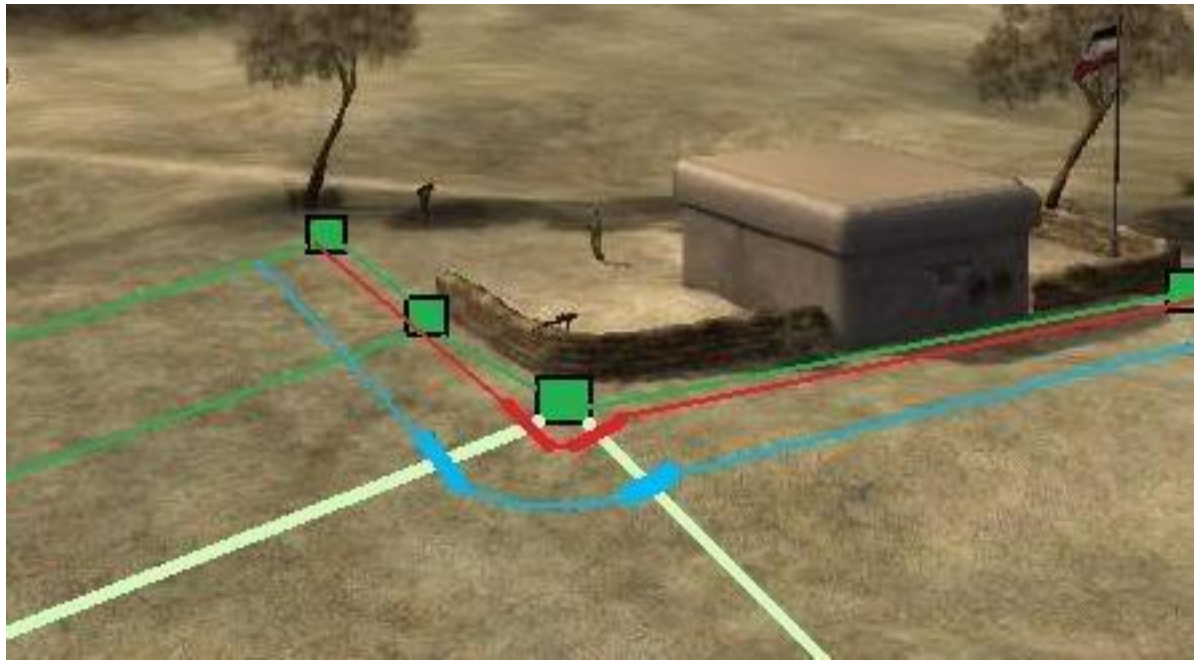




4) Waypoints don’t work well for different characters

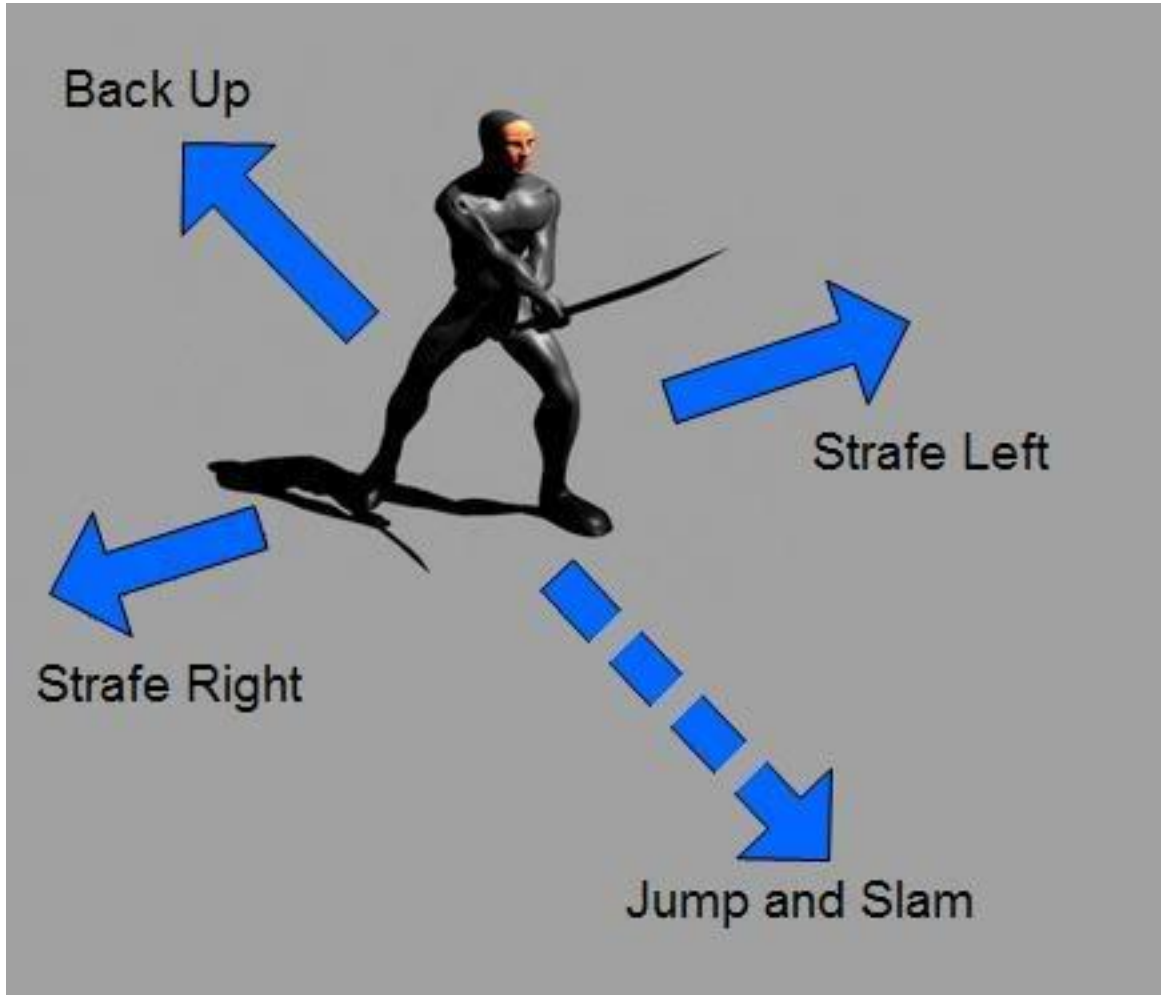


NavMesh solution



5) They don’t hold enough data

- Game character frequently queries the pathfinding system
- Can **test the predicted end position of each of these animations against the navigation mesh**
- Raycasting is possible, but expensive, and… can't tell me if the swordsman will land in a position that the level designers actually want characters to walk in



Designers need to be able to add info…

Nav Meshes allow for many agents

NavMesh

- Isn’t pathfinding on a NavMesh slower?
  - No

- Graph usually has fewer nodes
- Movement not restricted within the mesh (**convex poly assumption**)
- Only need to path in between individual sections of the mesh

NavMesh

- Don’t they take up a lot of memory?
  - No

- Can be smaller than dense waypoint graphs
- Smaller than collision mesh (ignores walls, etc.)
  - https://developer.valvesoftware.com/wiki/Collision_mesh
- Fairly compact representation
- May be generated automatically

Question 2:
Memory

Rank these four space representations according to the memory they would use for the same simple scene (empty space and obstacles):

1. Grid
2. Path network (designed)
3. Path network (flood fill)
4. Nav Mesh + Path network

Why?

Game design can cover Game AI

- Cheating / hiding the problem
  - Most AIs don't live long enough to let you spot the flaws in their pathfinding (LOS stop, shoot)
  - Many 1P FPS, AIs don't move very much, shoot from relatively fixed position.
  - FPS games with AI sidekicks kill the enemy AIs so quickly they don't have time to move very far.
  - AI agents can “give up” and return to a safe default
    - ~0:50 https://www.youtube.com/watch?v=gXjUzHhNjIA

FPS implications

- What if we force characters to use melee weapons (e.g. Covenant soldiers in Halo, the Icarus stealth assassins in F.E.A.R., or the Metroids in a Metroid Prime game)? Which world rep technique did they use?

How do you handle walking under bridges?

Is this good for all games?

- Not necessarily
- Find the right solution for your problem