Faculty of Mathematics and Physics
Charles University
October 29th, 2024

Artificial Intelligence for Computer Games

Spatial awareness
Continuous pathfinding

Adam Dingle

Spatial Awareness

Spatial awareness is the capacity to understand and reason about the relations among objects in space.

Spatial Awareness

How do you perceive the world?

Spatial Awareness
How do you perceive the world?

Spatial Awareness

Spatial Awareness is a general term for any code / data structure that provides reasoning about the environment:

- What items are available in my vicinity / in the map?
- How do I get to item X?
- What (safe) routes lead from my base to the enemy base?
- What is a good guarding/cover/ambush spot?
- A rocket is approaching! How can I dodge it? Can I fire back?
- …

Spatial Awareness
How to reason about the virtual world?

- Floors, Walls, Pits, Static obstacles

Spatial Awareness
How to reason about the virtual world?

- Floors, Walls, Pits, Static obstacles
- Doors, Gates, Ladders, Stairs

Spatial Awareness
How to reason about the virtual world?

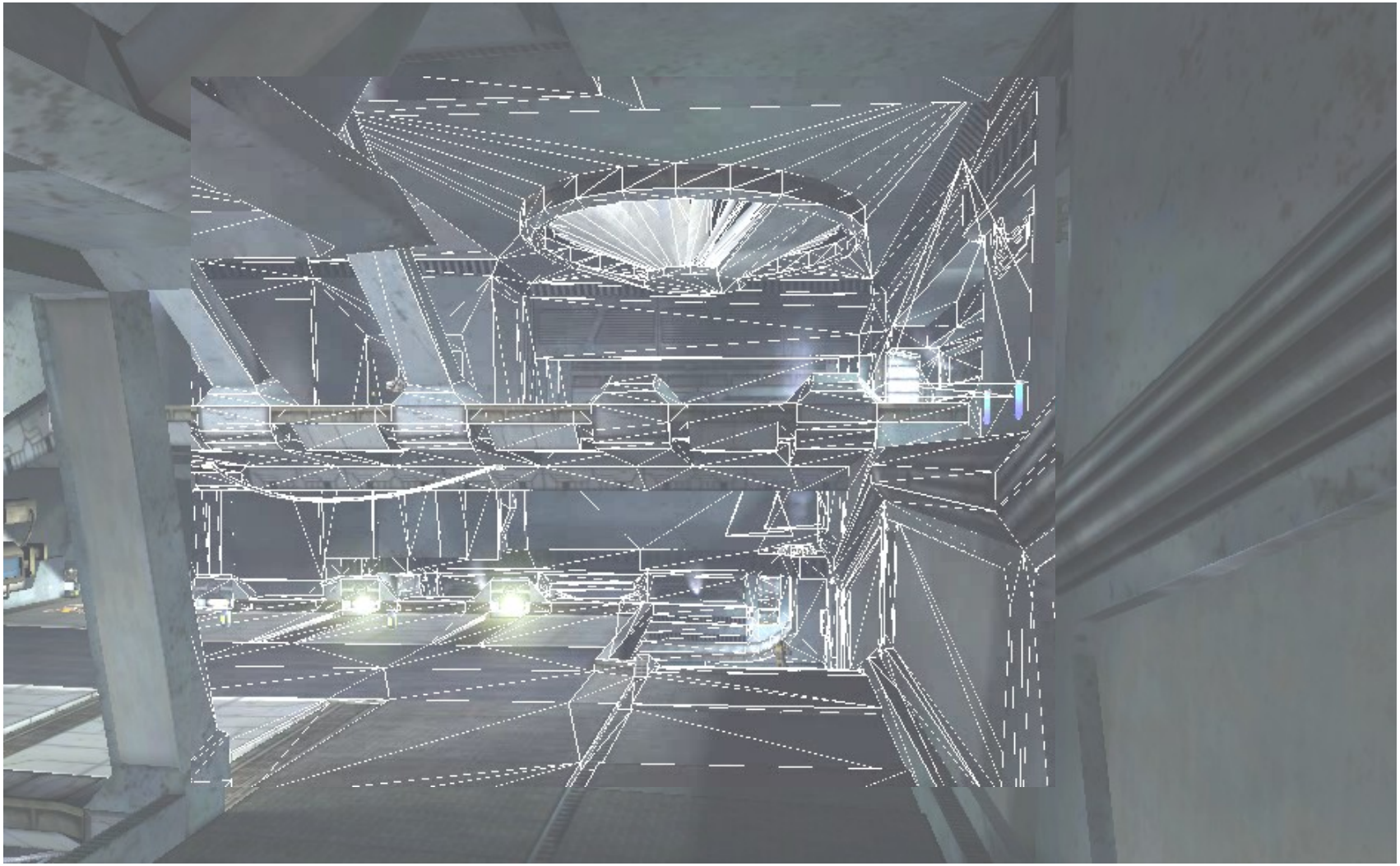
- Floors, Walls, Pits, Static obstacles
- Doors, Gates, Ladders, Stairs
- Items

Spatial Awareness
How to reason about the virtual world?

- Floors, Walls, Pits, Static obstacles
- Doors, Gates, Ladders, Stairs
- Items
- Dynamic objects, Other agents

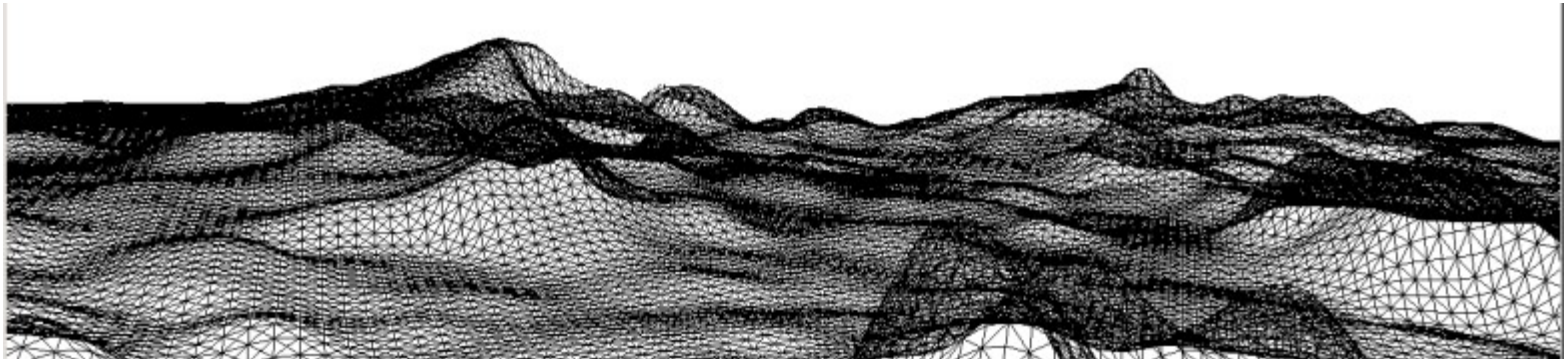
Environment Representation

Environment Representation

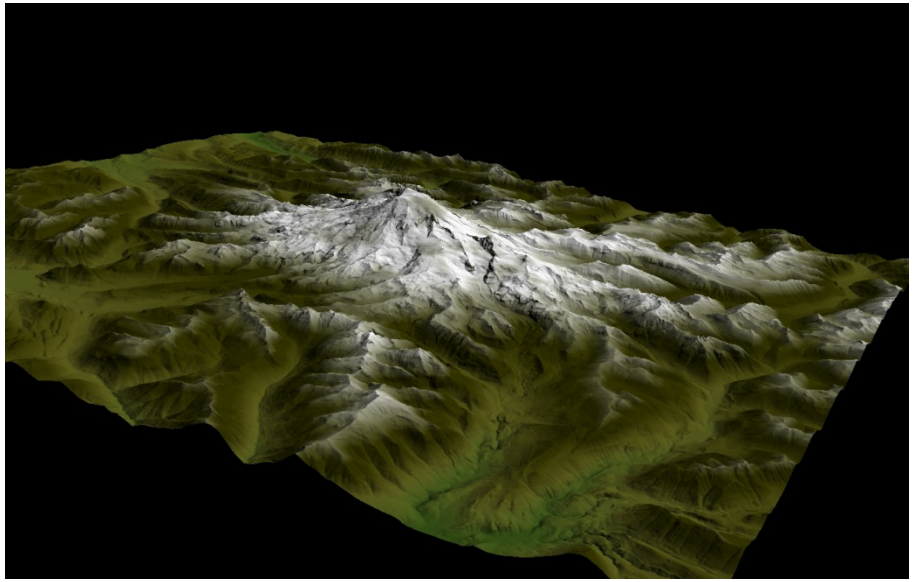


Environment Representation

Height maps



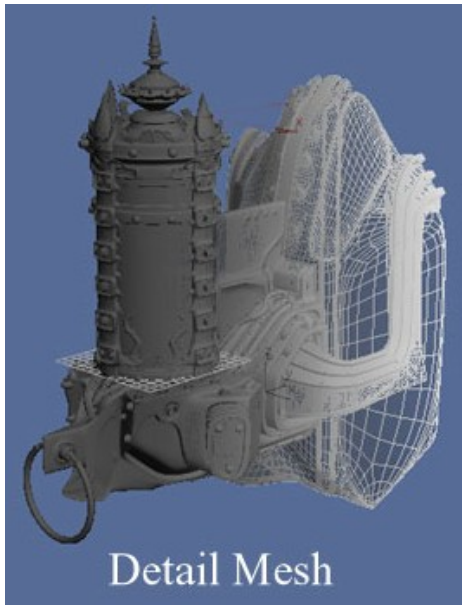
2D matrix of heights transformed into a mesh

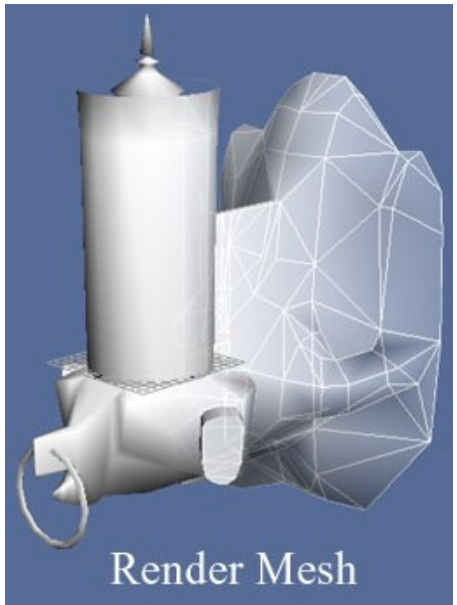


Environment Representation

- Environment
  - Terrain
  - Walls
  - Objects

⇒ It all boils down to triangles

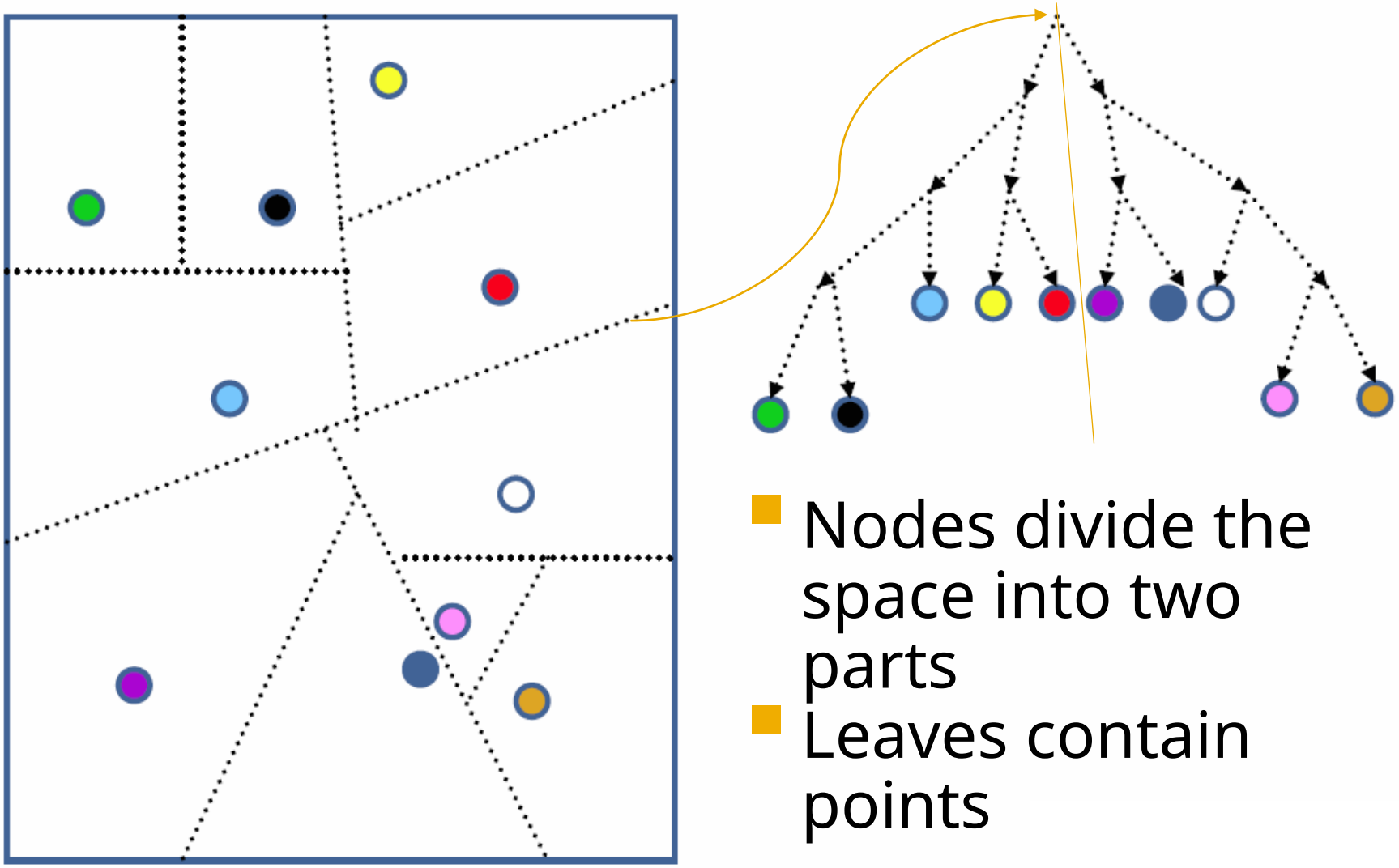




Environment Representation
BSP (Binary Space Partition) Trees

- widely used data structure
  - found in classic games (Doom, Quake)
- work in any number of dimensions
- partition space hierarchically into convex regions
- partitions are hyperplanes: lines (in 2D) or planes (in 3D)

Environment Representation
BSP Trees: storing points



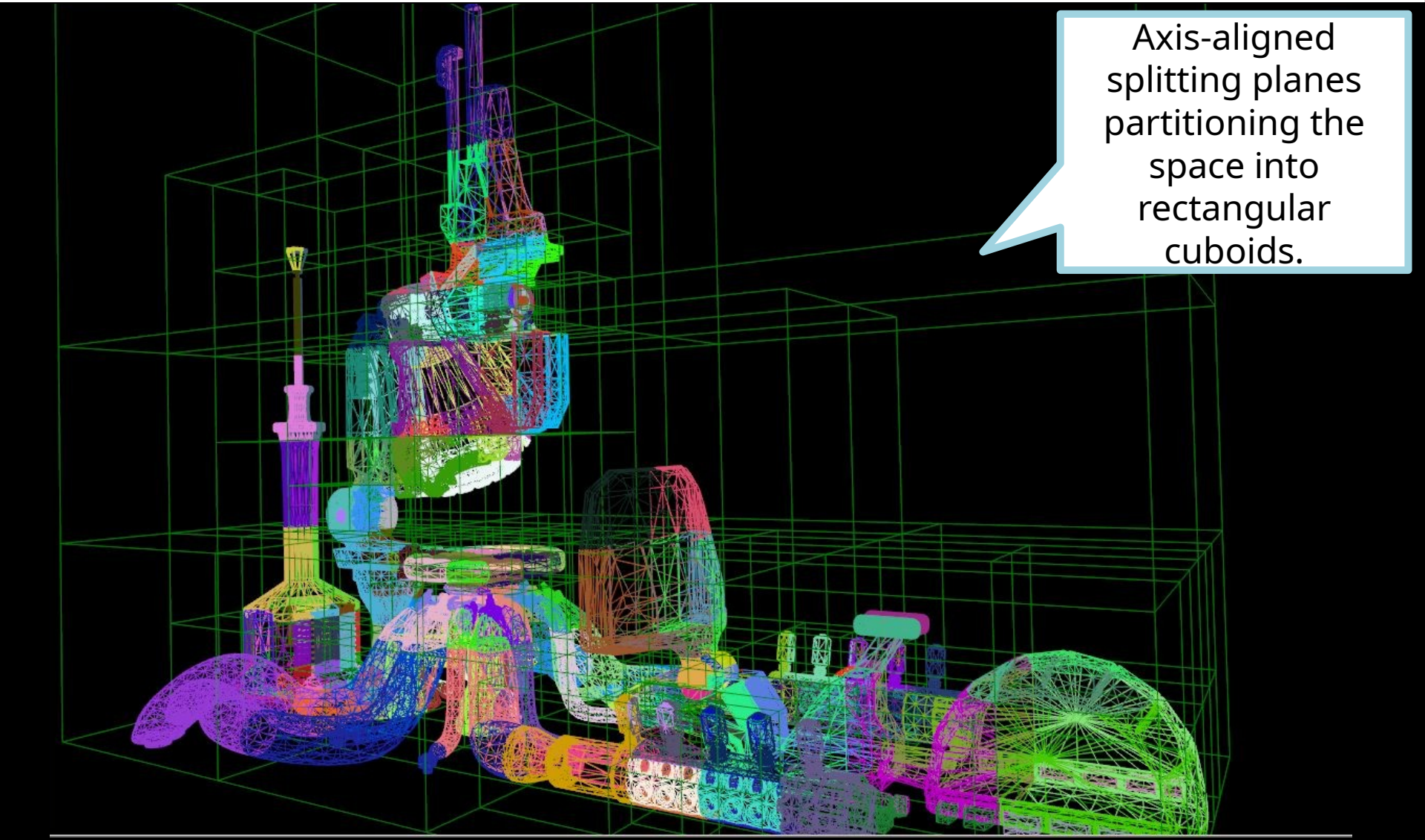
- Nodes divide the space into two parts
- Leaves contain points

Environment Representation
BSP Trees

- We've seen how to store points
- More commonly, we want to store segments (in 2D) or polygons (in 3D)
  - So even interior nodes may contain objects
- How to choose splitting hyperplanes?
  - axis-aligned partitions: kd-trees
  - auto-partition: extend segments/polygons to make hyperplanes

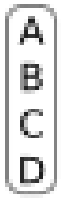
Environment Representation

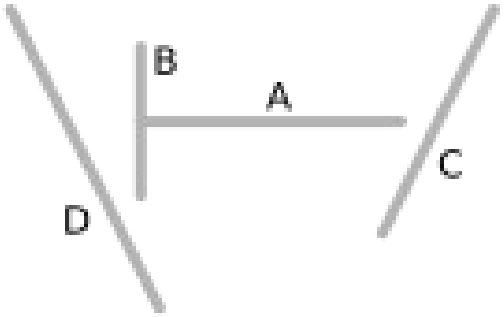
BSP Trees

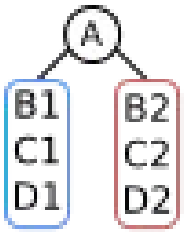


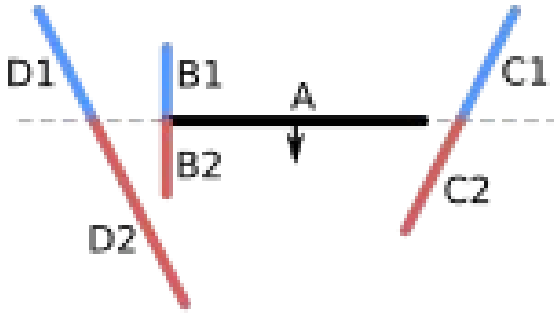
Environment Representation
BSP Trees: Constructing

- Auto-partition: choose an object, extend to a plane, place at root
- Any other objects may need to be split!



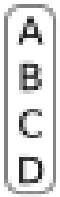


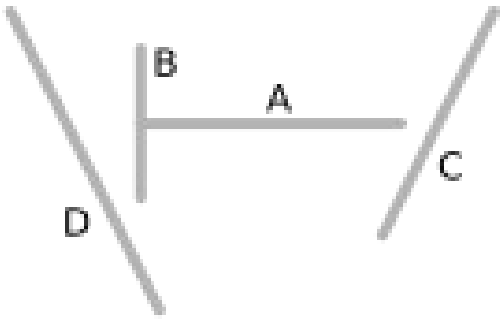


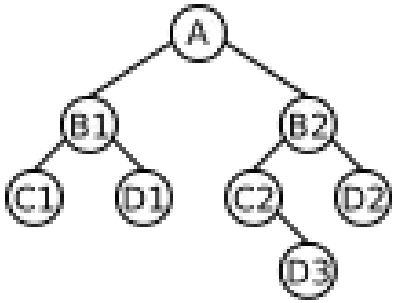


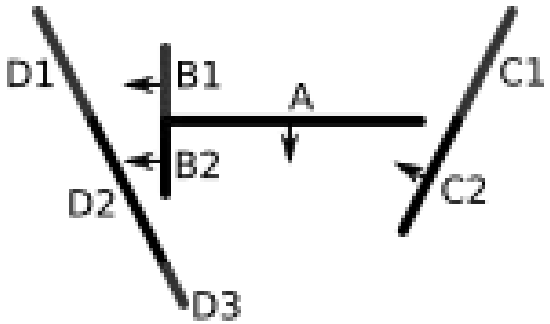
Environment Representation
BSP Trees: Constructing

- Auto-partition: choose an object, extend to a plane, place at root
- Any other objects may need to be split!





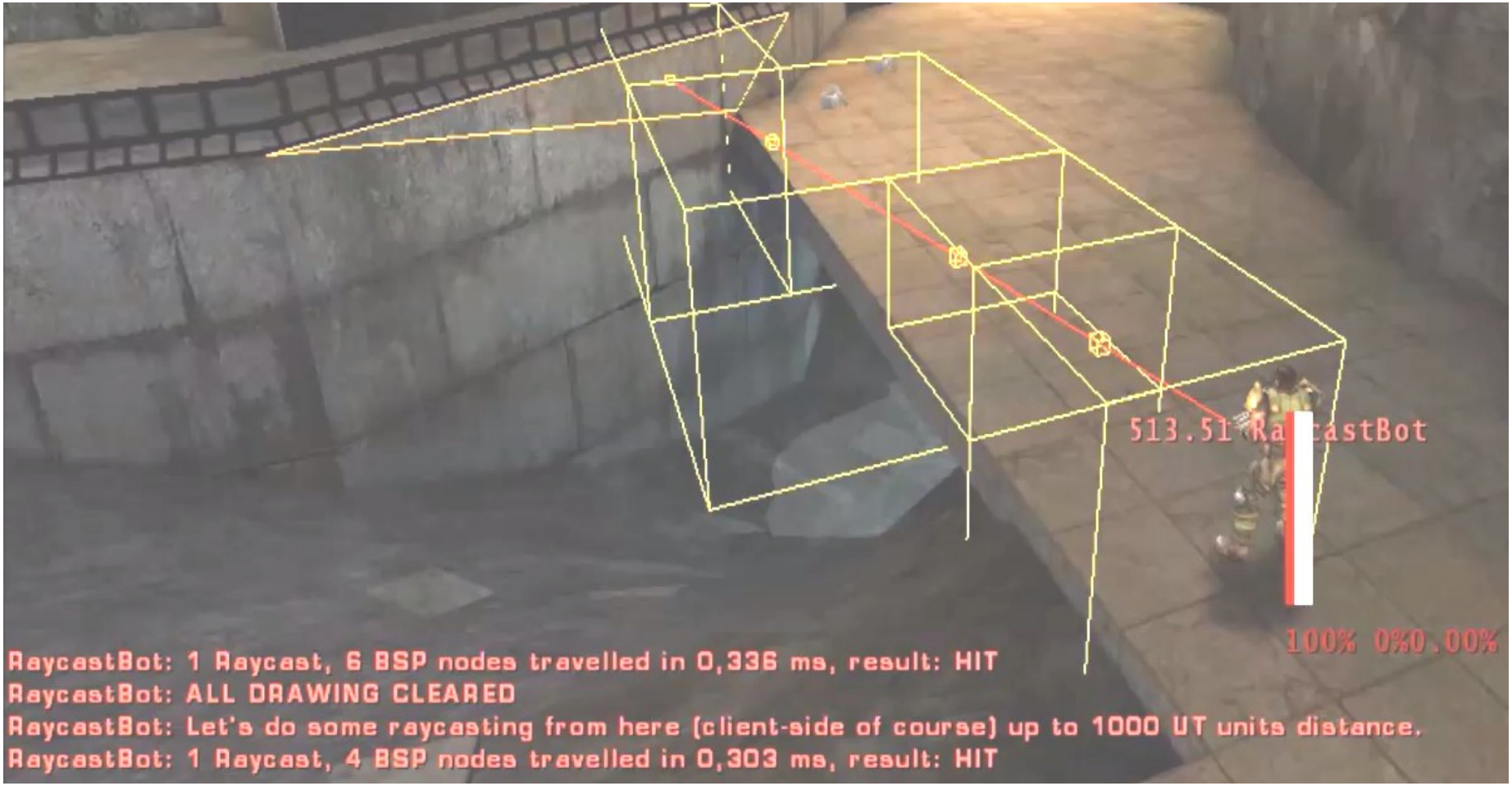




Environment Representation
BSP Trees: Operations

- raycasting
- collision detection
- ordering polygons for rendering
- not very useful for pathfinding

Environment Representation
Raycasting

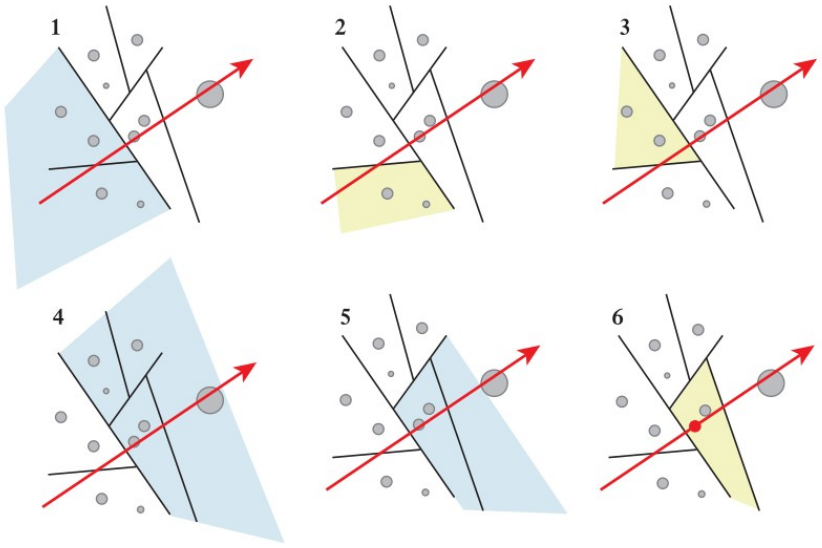


Environment Representation
Raycasting

Can answer questions:

- Can I see (x_1,y_1,z_1) from (x_2,y_2,z_2)?
- What will I see looking in direction v from (x_1,y_1,z_1)?

Environment Representation
Raycasting in a BSP tree



Environment Representation
Visibility testing in a BSP tree

```
function intersects(P, Q, node):
  if PQ intersects a primitive in the node:
    return true

  if node is a leaf:
    return false

  closer, farther = node.positiveChild, node.negativeChild

  if P is in the negative half-space of node:
    swap closer, farther

  if intersects(P, Q, closer):
    return true

  if P and Q are in the same half space of node:
    return false

  return intersects(P, Q, farther)

function visible(P, Q):
  return not intersects(P, Q, root)
```

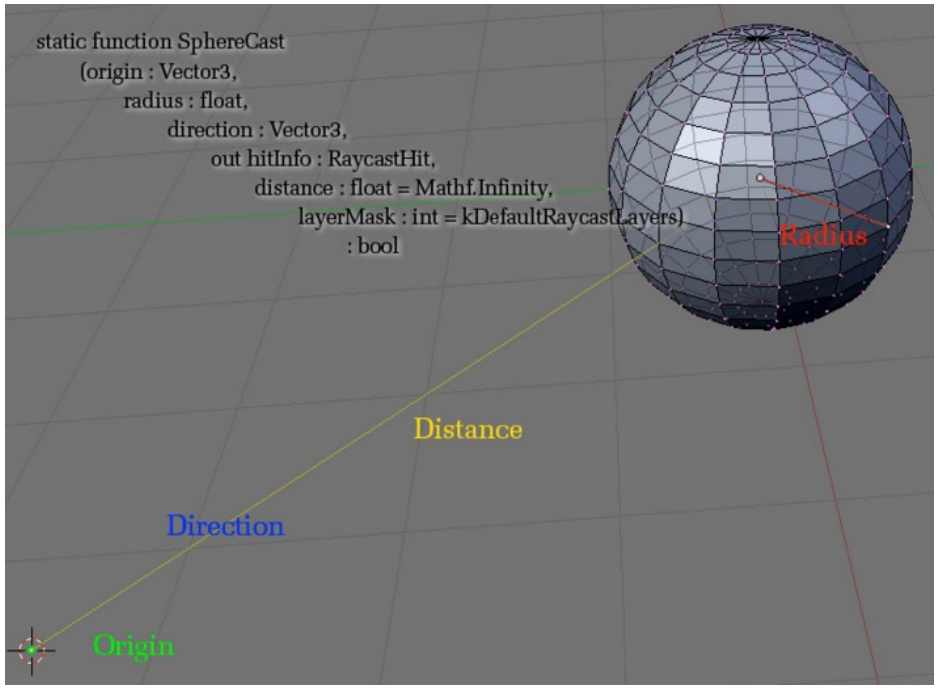
adapted from Hughes et al, *Computer Graphics: Principles and Practice, 3rd ed.*, 36.2.1 "BSP Ray-Primitive Intersection"

Environment Representation
Raycasting in a BSP tree

- To raycast from P in direction v:
  - Let Q be a point at infinity in direction v
  - Check whether Q is visible from P
  - If not, return the first object that was hit

Environment Representation
Detour: Spherecasting

- Volumes can be “raycast” using spherecasting
- Some physics engines (e.g. Unity) can do this

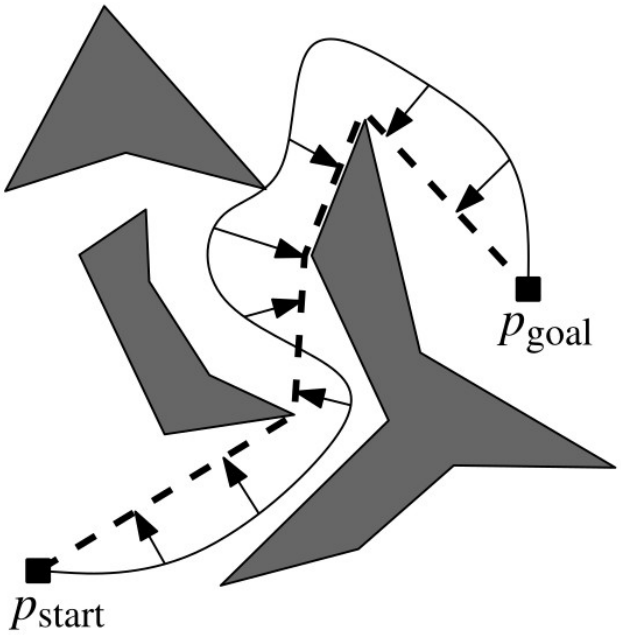


Environment Abstraction
Structures for continuous pathfinding

- Visibility graphs
- Waypoint graphs
  - also called "navigation graphs"
- Navigation meshes

Environment Abstraction
Visibility graphs

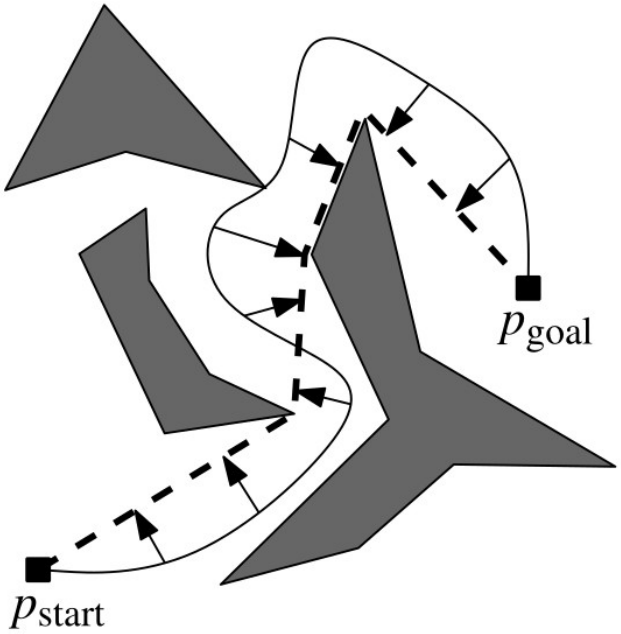
- From computational geometry
- Goal: Find shortest path in 2D from start to goal among a set of polygonal obstacles



de Berg et al, *Computational Geometry, 3rd ed.*, 15.1 "Shortest Paths for a Point Robot"

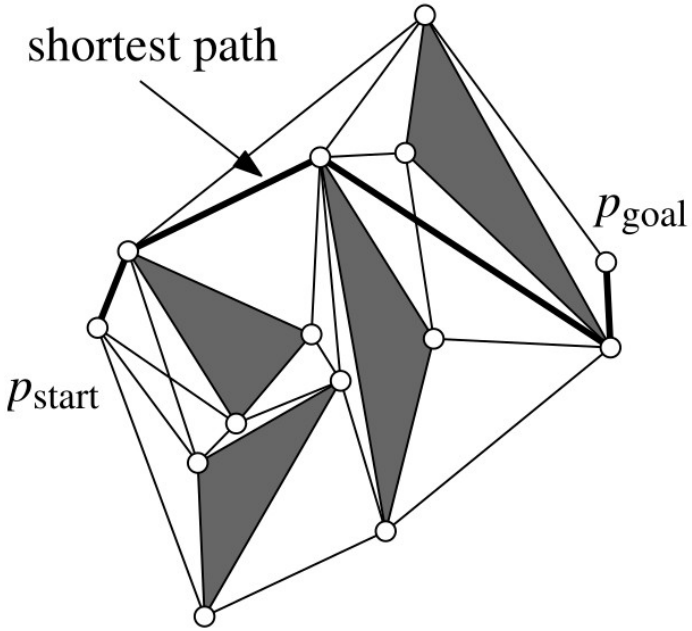
Environment Abstraction
Visibility graphs

- Lemma: Any shortest path from p_{start} to p_{goal} among a set S of polygonal obstacles is a sequence of line segments whose vertices (other than p_{start}/p_{goal}) are all vertices of S



Environment Abstraction
Visibility graphs

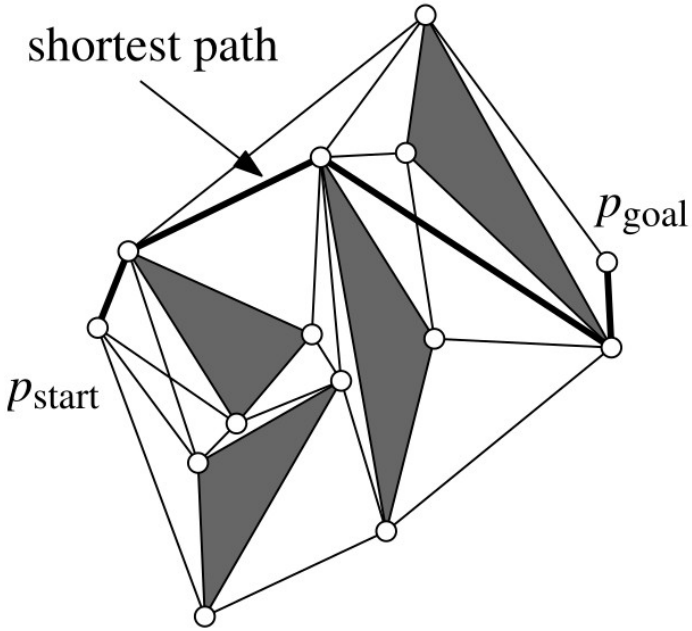
- Definition of *visibility graph* of *S*:
  - nodes = vertices of *S*
  - there is an edge between *v* and *w* if the segment *vw* does not pass through any polygon in *S*



de Berg et al, *Computational Geometry, 3rd ed.*, 15.1 "Shortest Paths for a Point Robot"

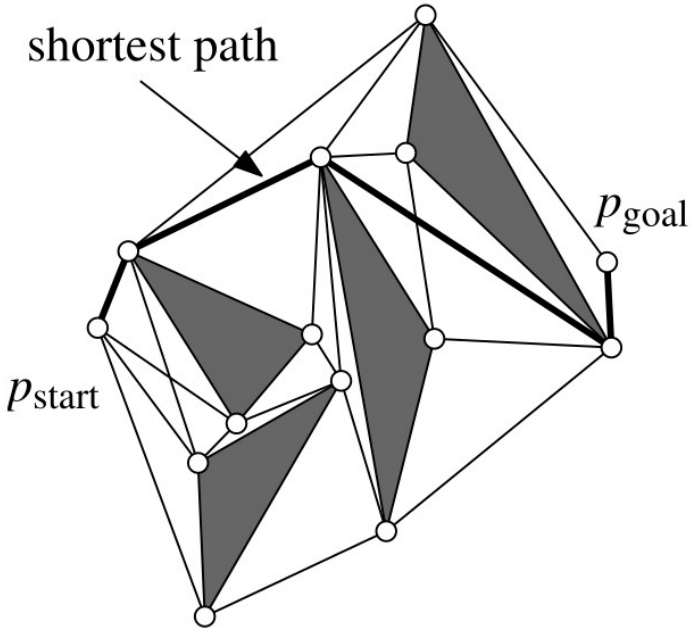
Environment Abstraction
Visibility graphs

- Using a visibility graph, we can find a shortest path using Dijkstra's algorithm
  - weight of each edge = Euclidean distance



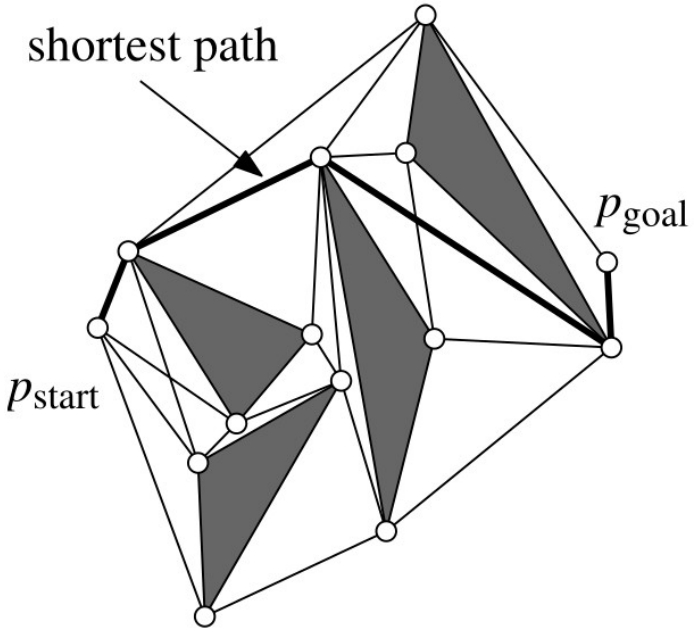
Environment Abstraction
Visibility graphs

- Suppose that polygons in *S* have a total of *n* vertices
- How efficiently can we compute a visibility graph for S?



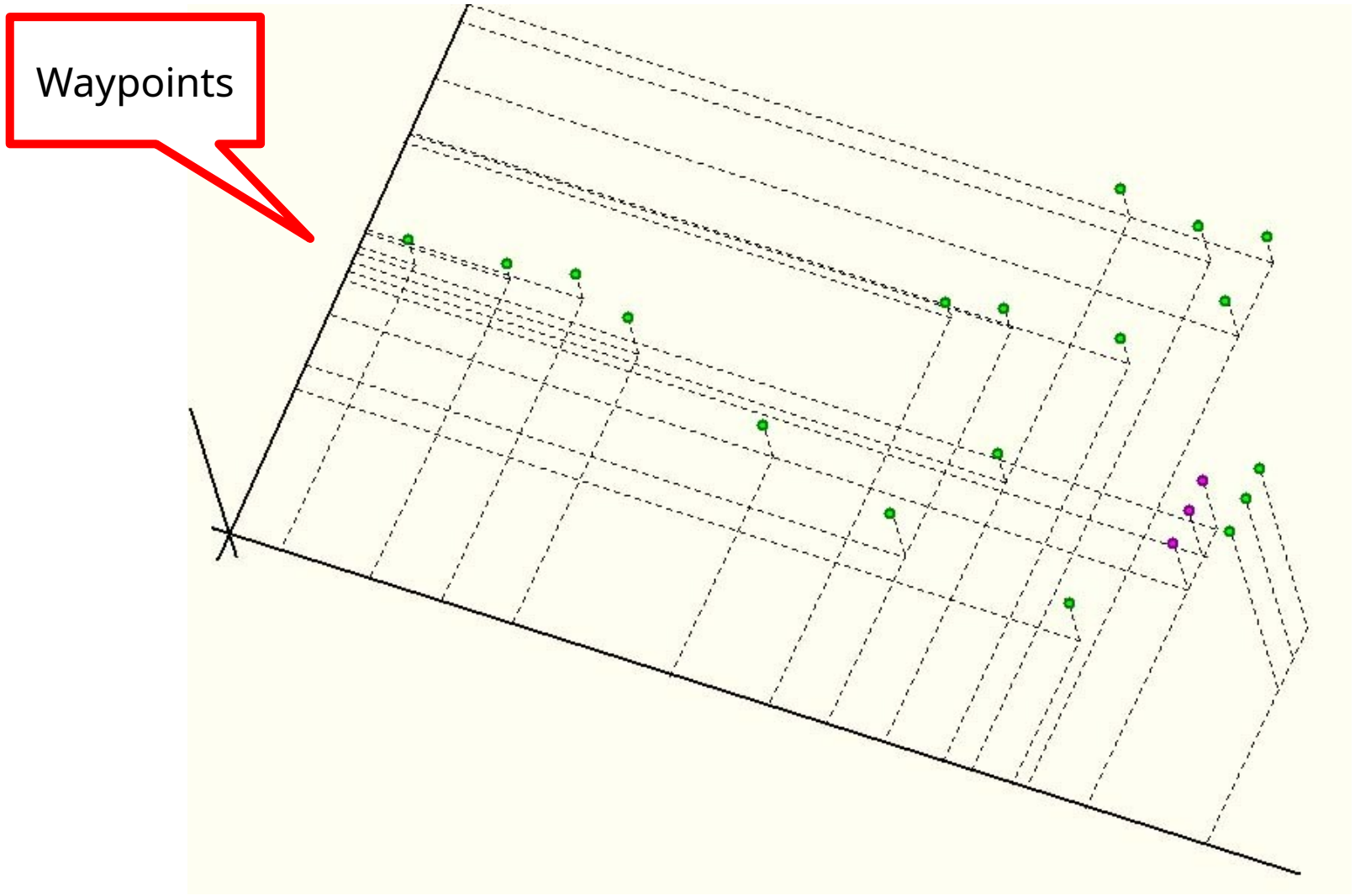
Environment Abstraction
Visibility graphs

- Naive algorithm runs in $O(n^3)$
  - for all pairs (v, w) of vertices, check whether vw intersects any polygon
- Faster algorithms are known
  - $O(n^2 \log n)$ (Lee, 1978)
  - $O(n \log n + k)$ where the graph has k edges (Ghosh and Mount, 1991)



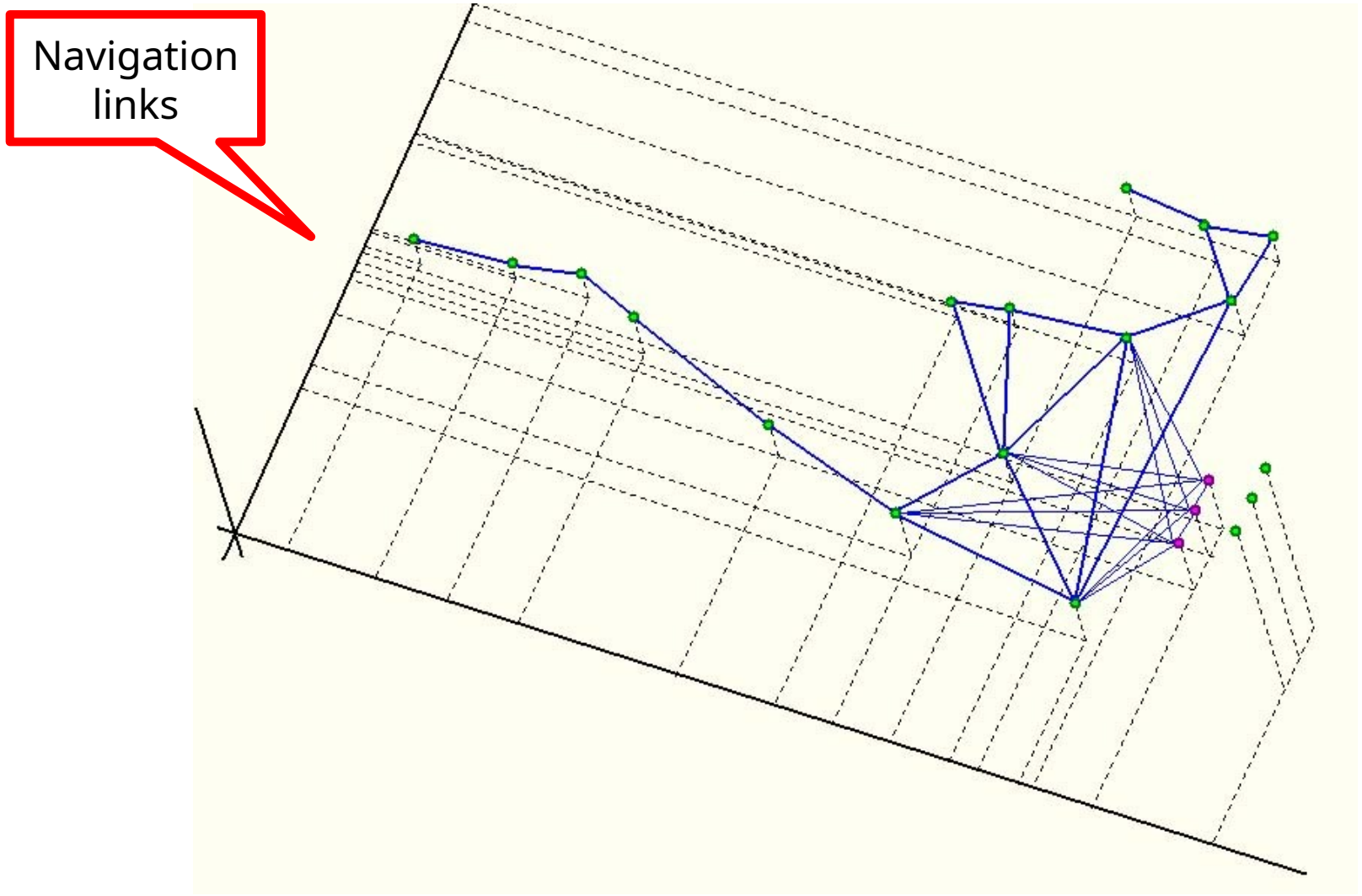
Environment Abstraction
Waypoint Graphs

Environment Abstraction
Waypoint Graphs

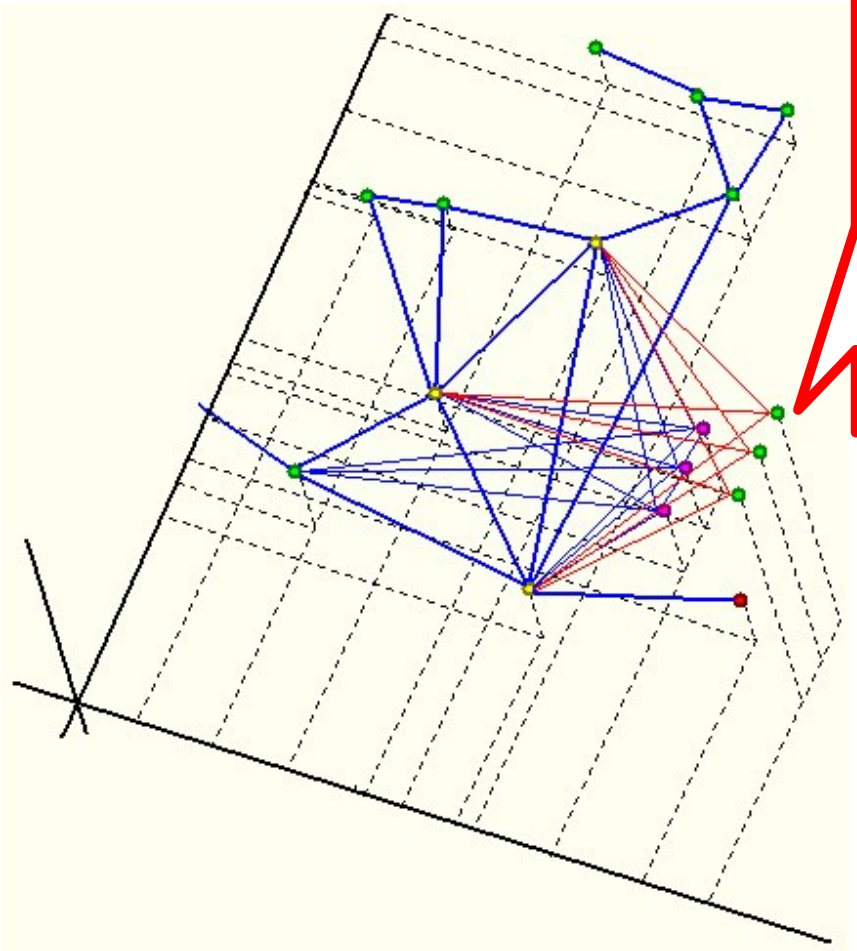


Environment Abstraction

Waypoint Graphs



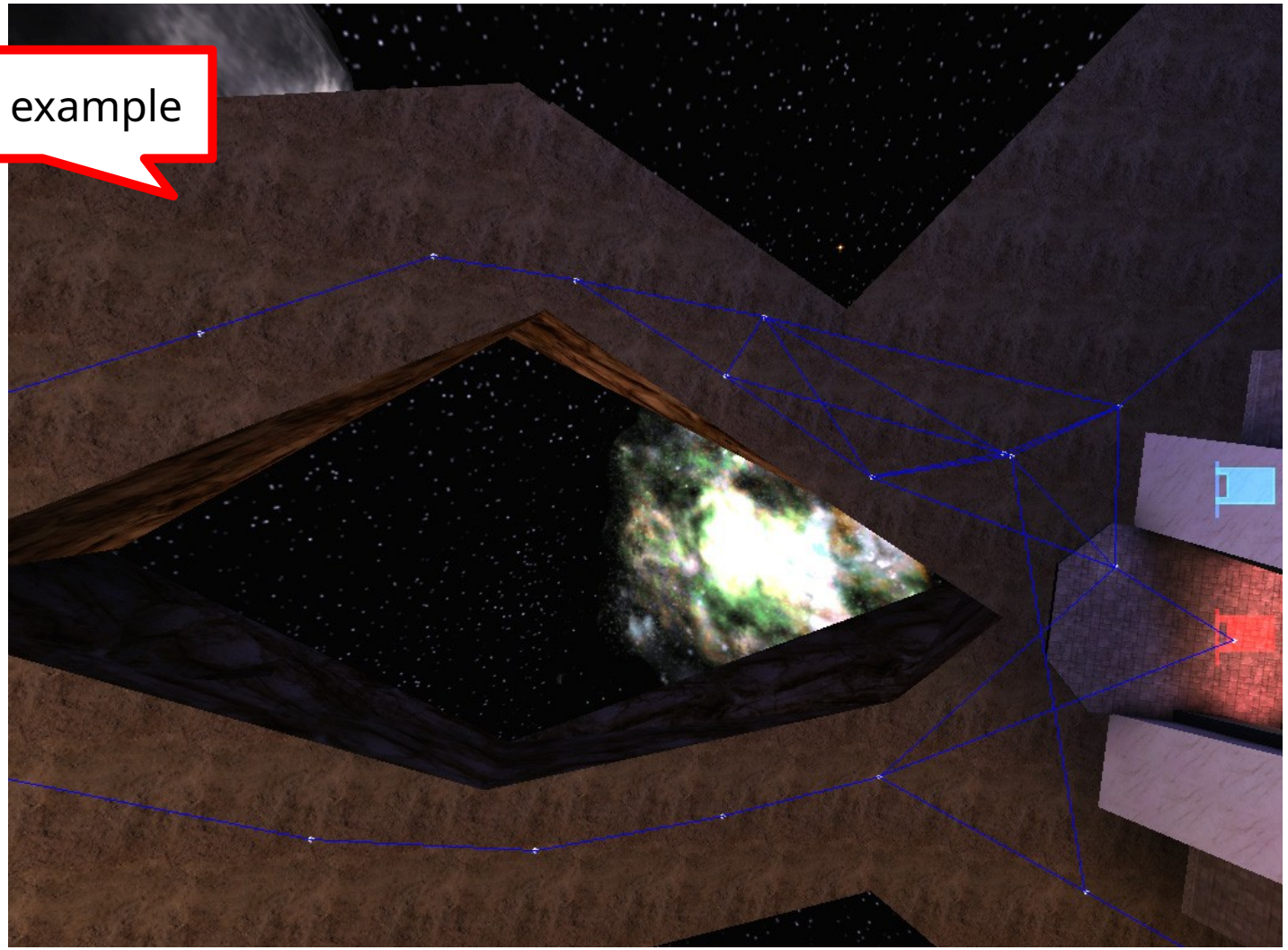
Environment Abstraction
Waypoint Graphs



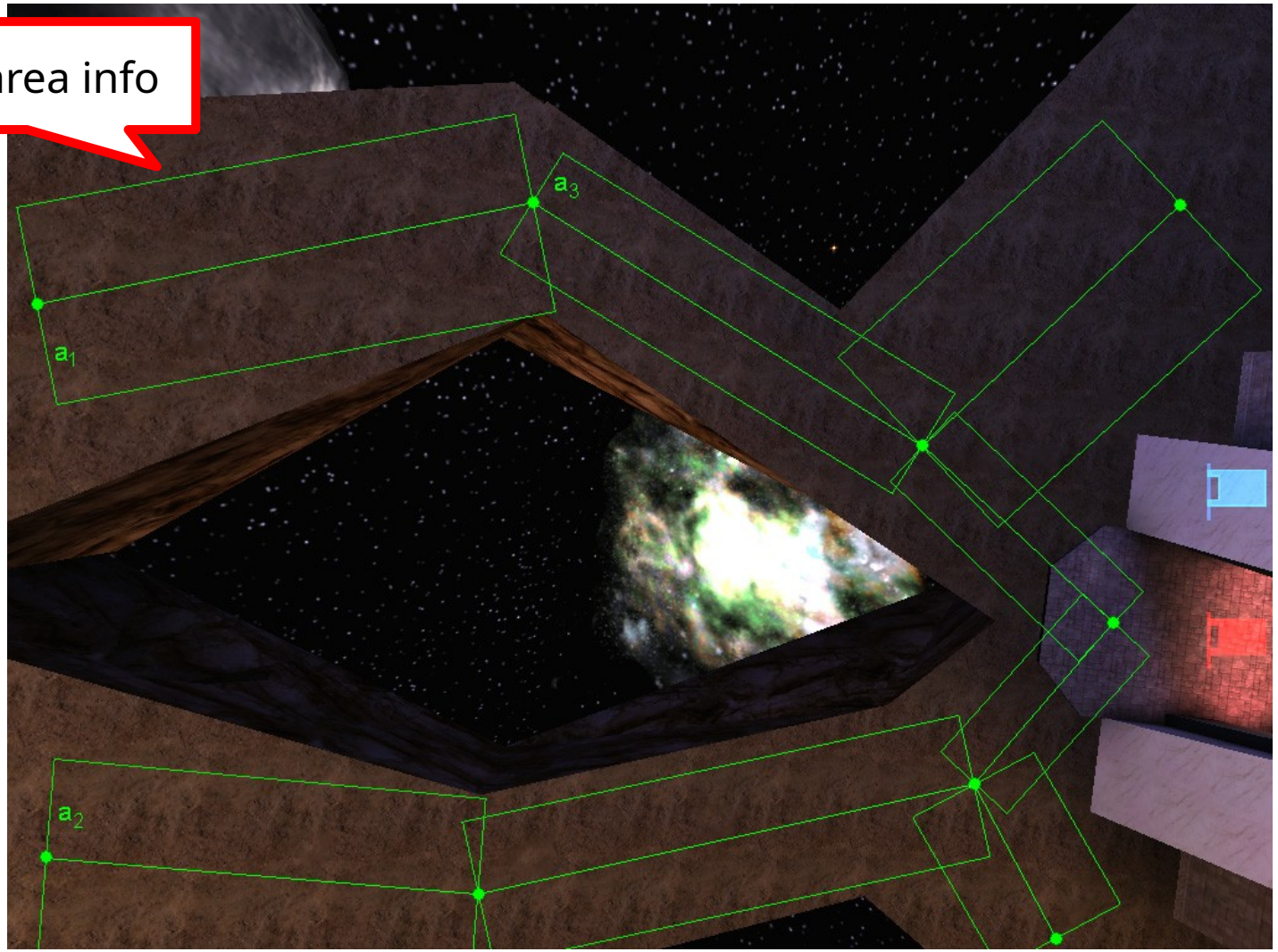
This is an item point.
This is a sniping point.
This is a guard point.
This is a teleport point....

Both points and edges may be then annotated

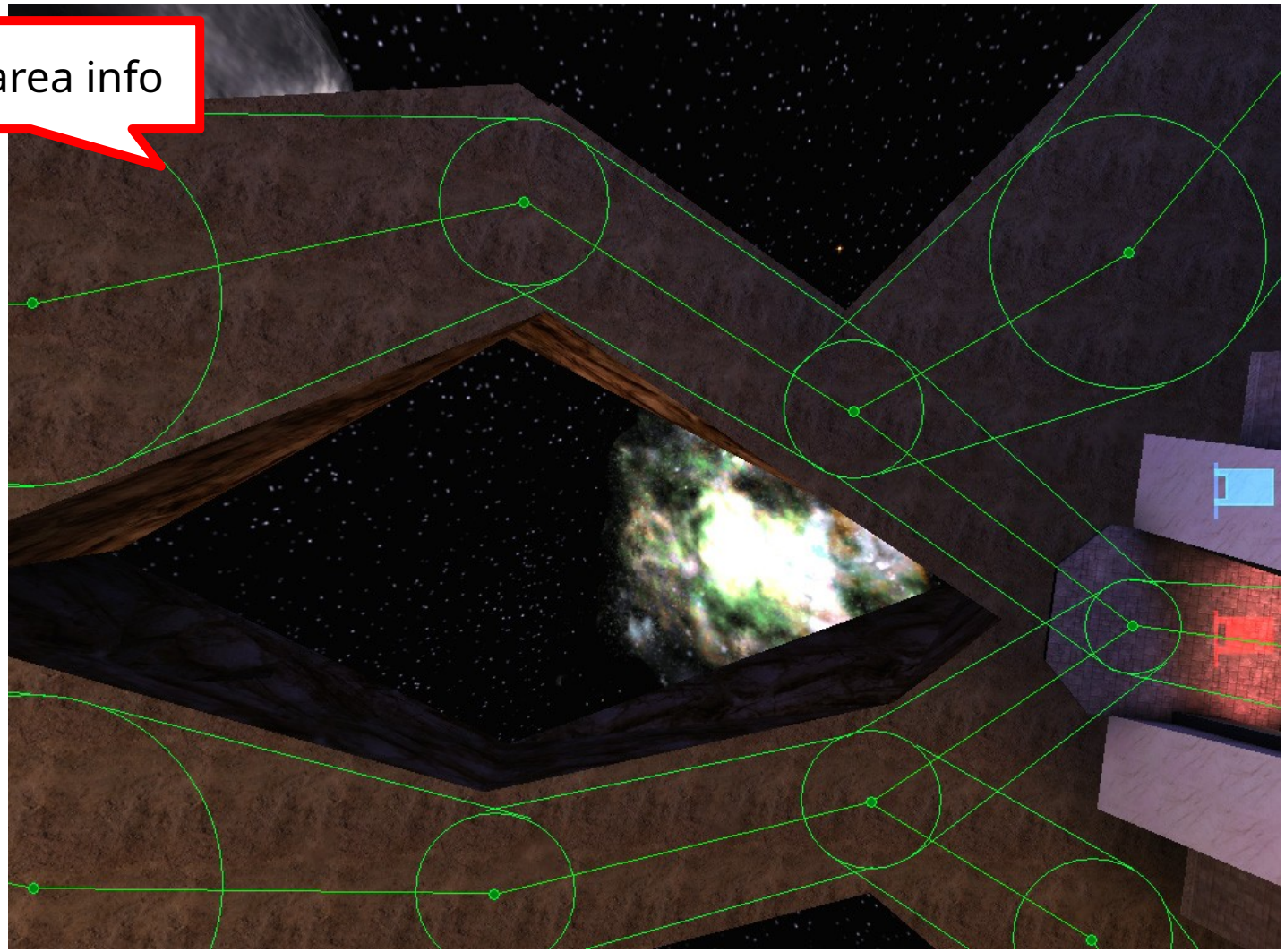
Environment Abstraction
Waypoint Graphs – 2D-ish improvements



Environment Abstraction
Waypoint Graphs - 2D-ish improvements



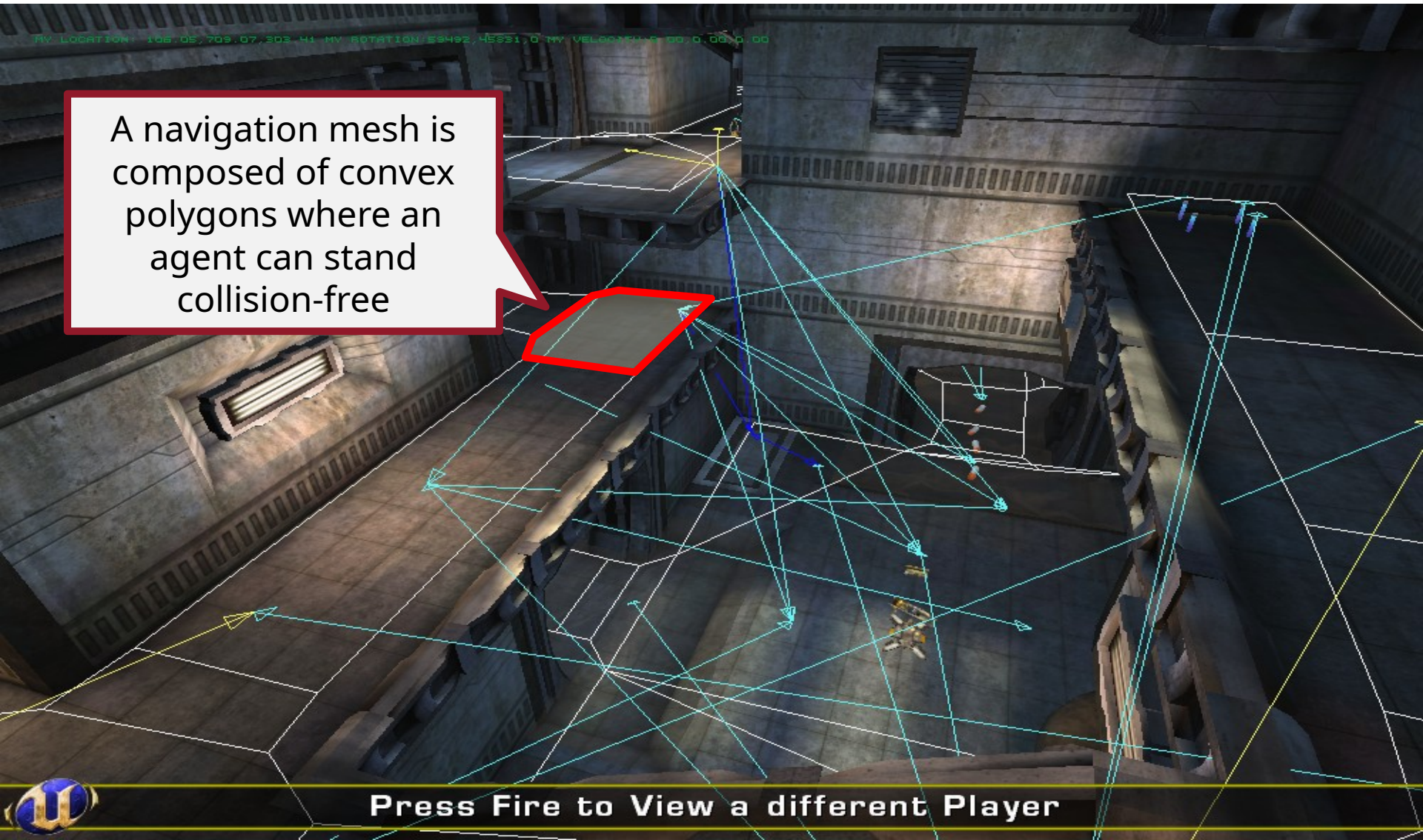
Environment Abstraction
Waypoint Graphs - 2D-ish improvements



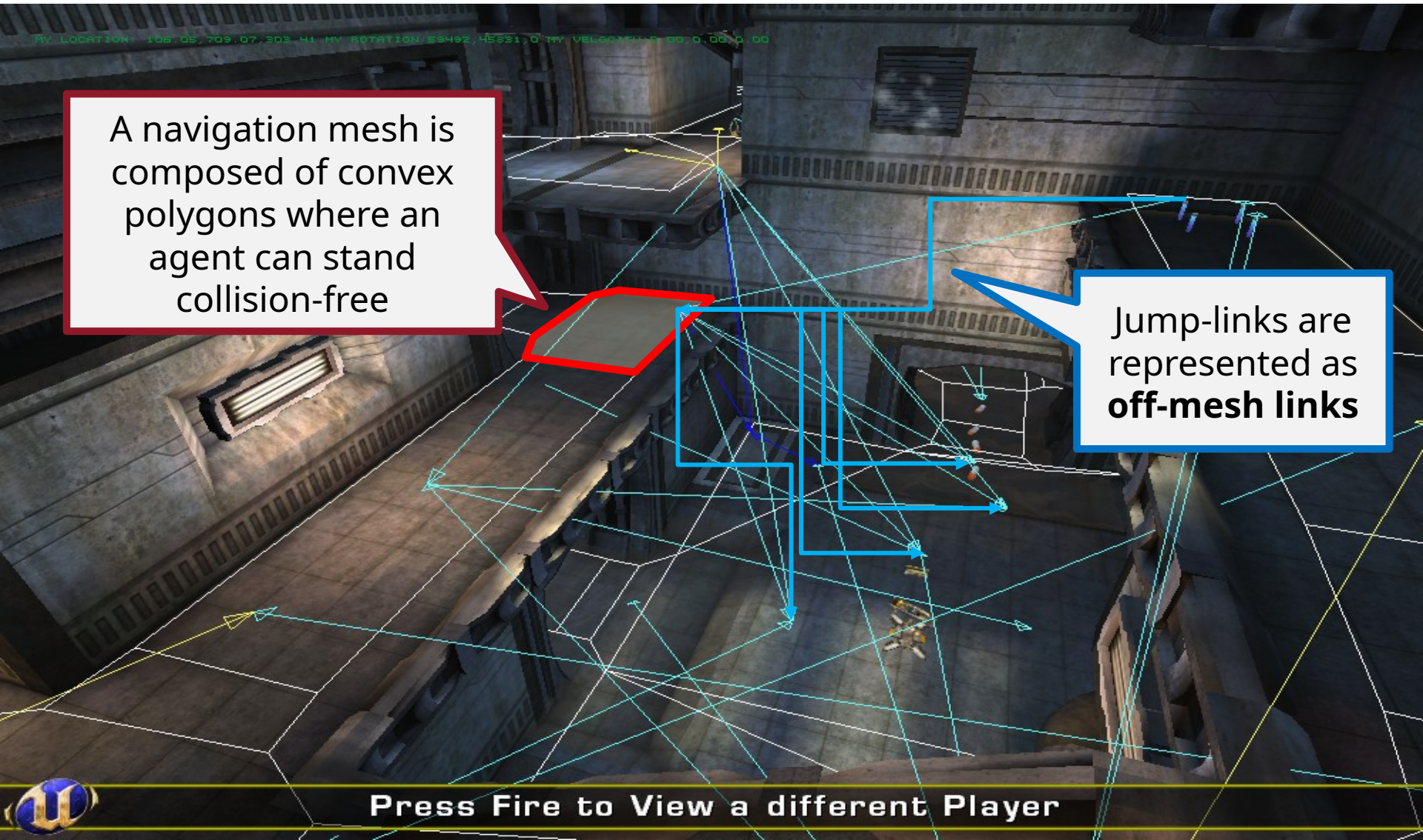
Environment Abstraction

Navigation Meshes

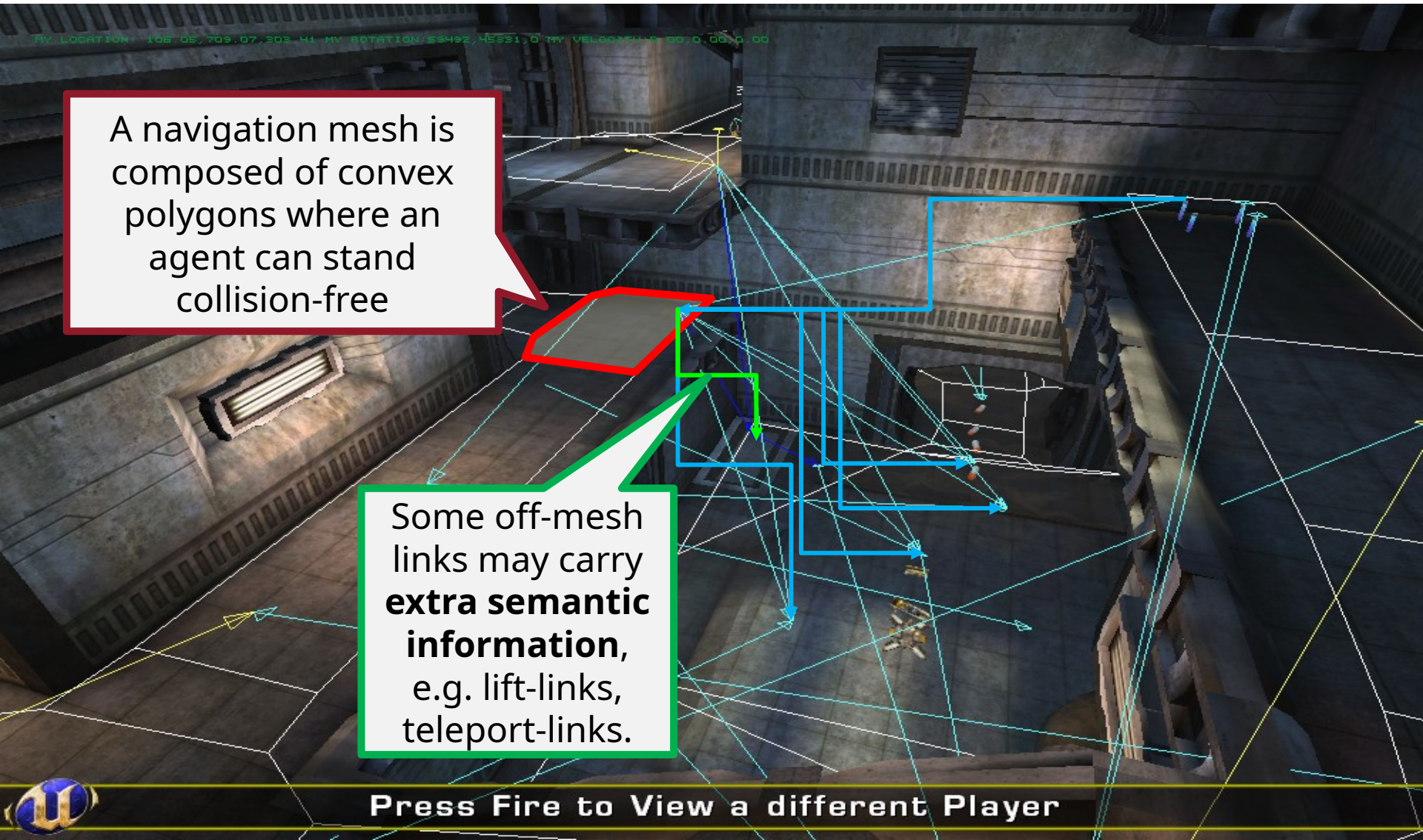
Environment Abstraction
Navigation Meshes



Environment Abstraction
Navigation Meshes



Environment Abstraction
Navigation Meshes



Environment Abstraction
Navigation Meshes

- Recast (Mikko Mononen, 2009)
  - Automatically creates navigation mesh from triangle data
  - open source
  - used in Godot, Unity, Unreal Engine
- Inputs:
  - triangle data
  - height, radius of agent(s) that will use mesh
  - maximum step-up distance
  - maximum slope that can be climbed
- Comes with Detour library for pathfinding

Environment Abstraction
Navigation Meshes

- How big is the benefit of this abstraction?

UT2004 Map	Text (XML)	Vertices	Triangles	log2(Tris)	NavPoints	NavMesh
	[MB]	[Count]	[Count]		[Count]	[Count]
DM-Flux2	6	86615	63611	15,95698865	194	413
CTF-FaceClassic	10	82189	68357	16,06080146	313	2492
CTF-January	30	502051	354342	18,43478295	438	3296
CTF-MoonDragon	60	745755	570444	19,12172574	498	4425

- Key points about navigation meshes
  - Better representation of the floor, cheap “nm raycast”
  - Automatic creation
  - Suitable for steerings, movement can be refined

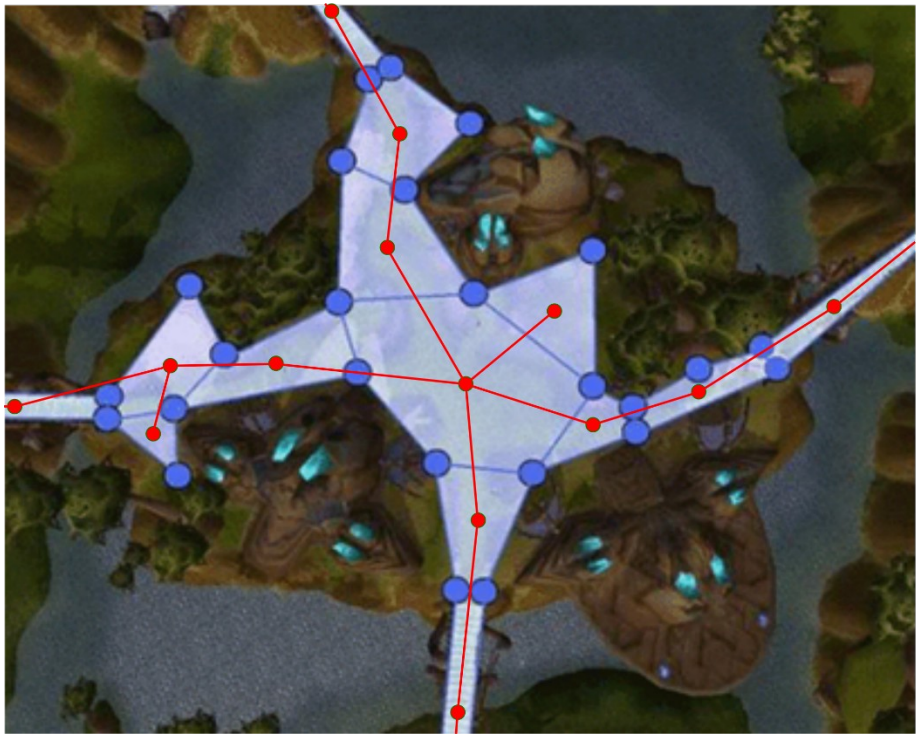
Environment Abstraction
Navigation Mesh – Finding the Path

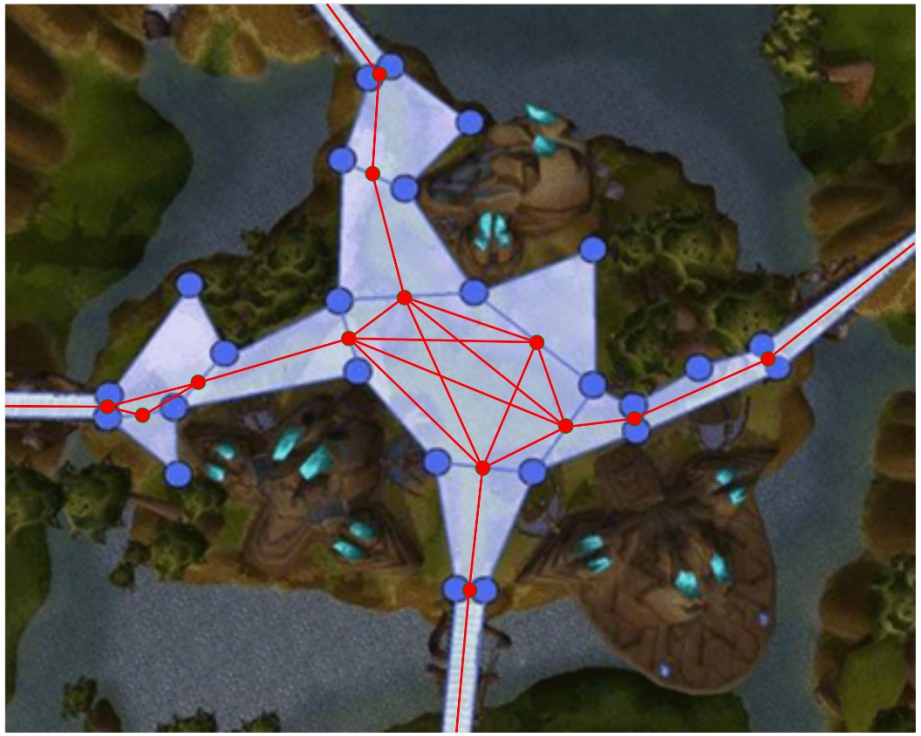
- We can use A* to search in a navmesh!
- We must choose points for distance calculations

http://gamedevevacademic.tumblr.com/post/69849130309/ai-pathing

Environment Abstraction
Navigation Mesh – Finding the Path

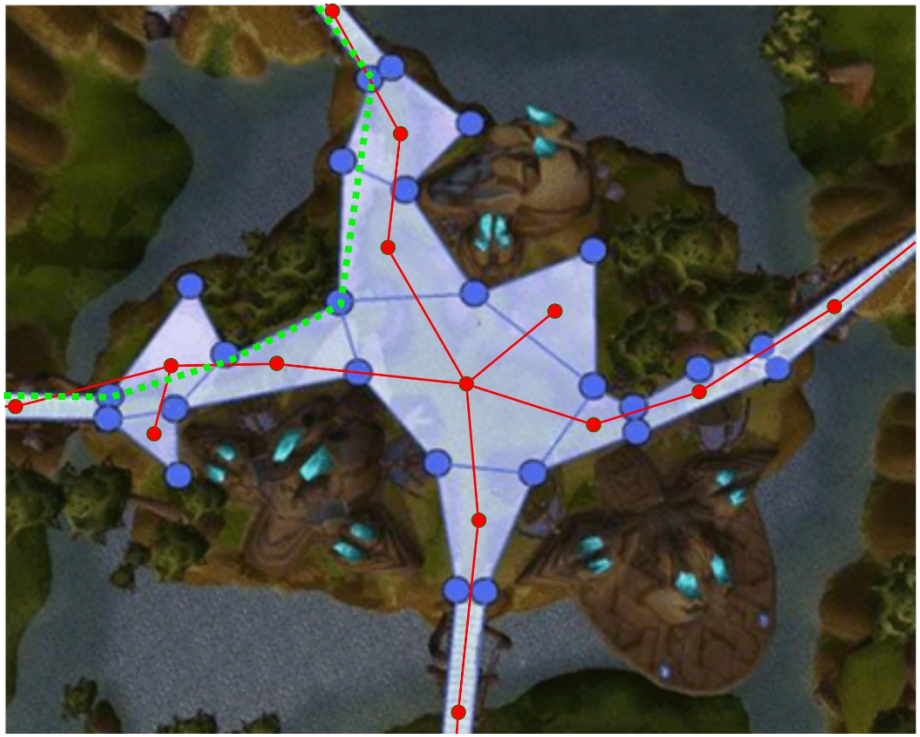
- Navmesh “graph” – two options

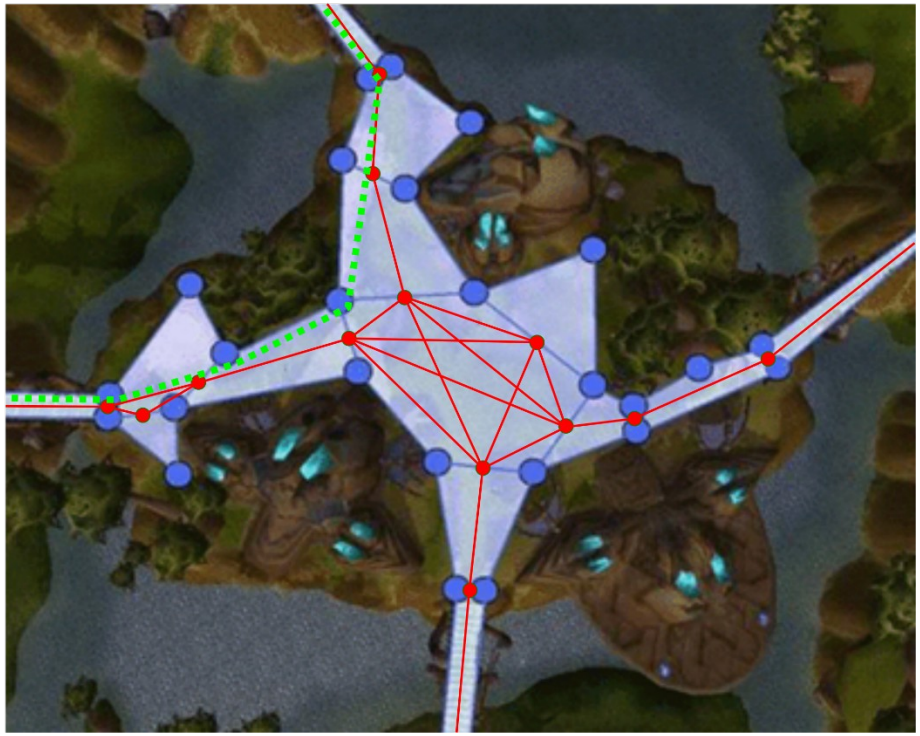




Environment Abstraction
Navigation Mesh – Finding the Path

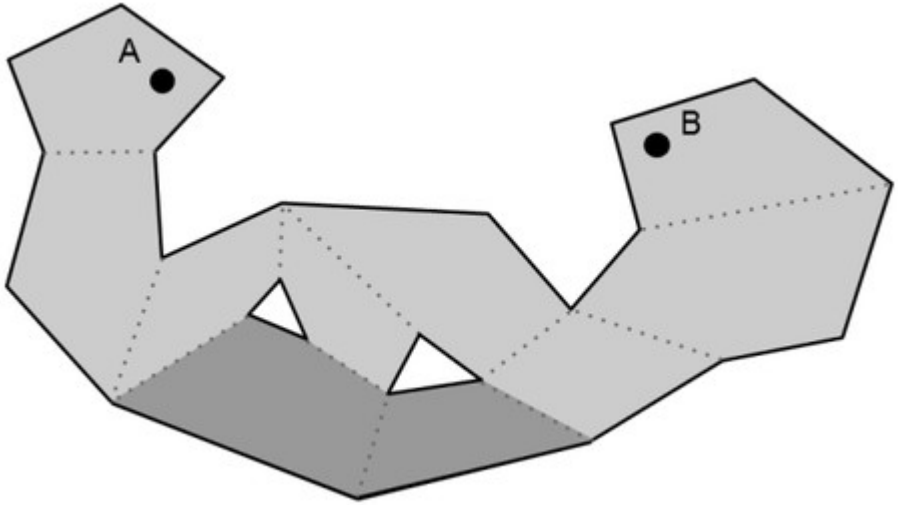
- Navmesh “graph” – two options
- Green path is the shortest (final) path





Environment Abstraction
Navigation Mesh – Refining the path

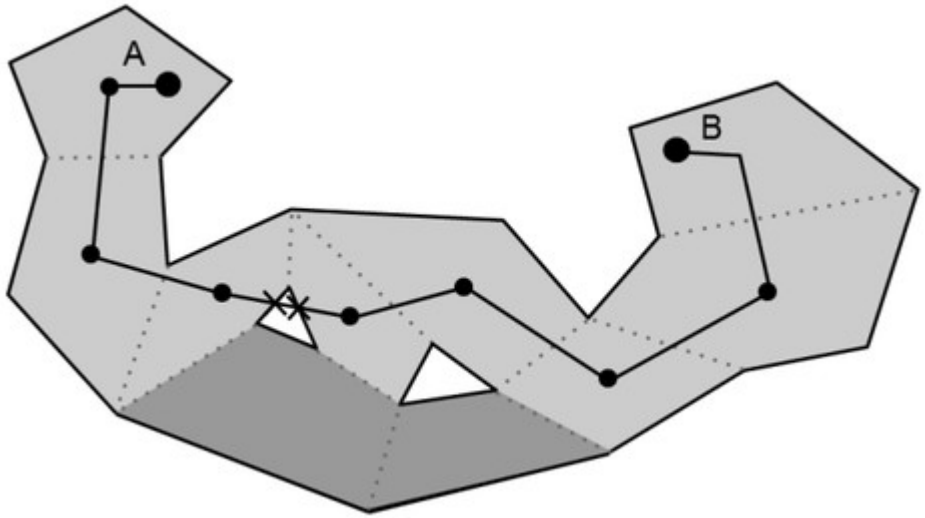
- We've used A* to find a corridor of polygons from A to B
- Now, how to produce a path?



http://gamedevevacademic.tumblr.com/post/69849130309/ai-pathing

Environment Abstraction
Navigation Mesh – Refining the path

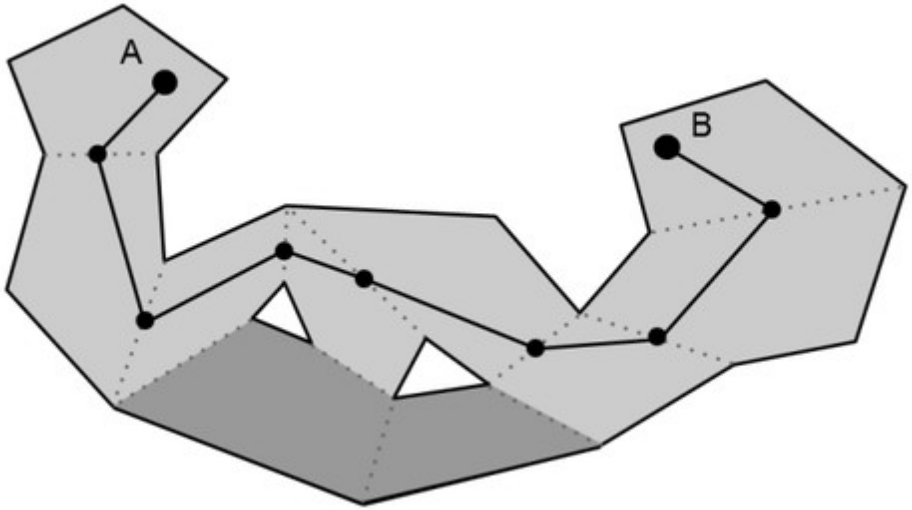
- Using polygon centroids is no good



http://gamedevevacademic.tumblr.com/post/69849130309/ai-pathing

Environment Abstraction
Navigation Mesh – Refining the path

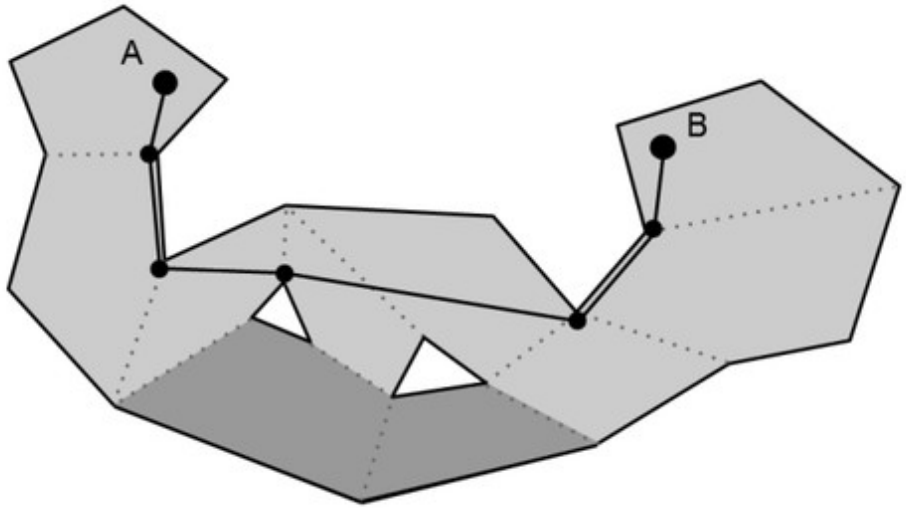
- Using middle points of sides is far from optimal



http://gamedevevacademic.tumblr.com/post/69849130309/ai-pathing

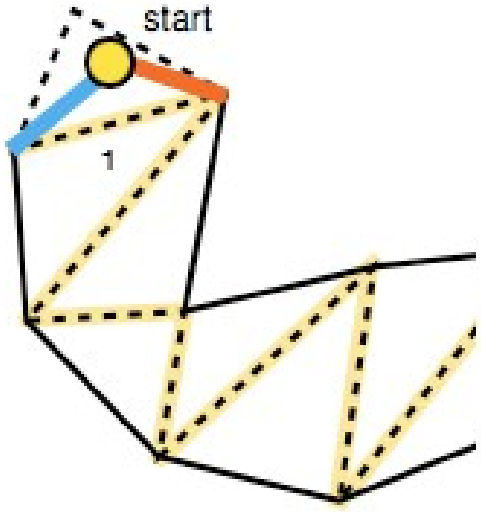
Environment Abstraction
Navigation Mesh – Refining the path

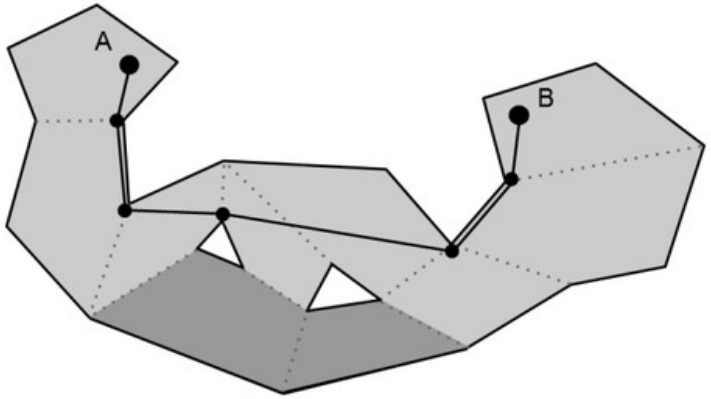
- The *funnel algorithm* finds a shortest path from points A to B along a corridor of polygons
- Sometimes called "string pulling"



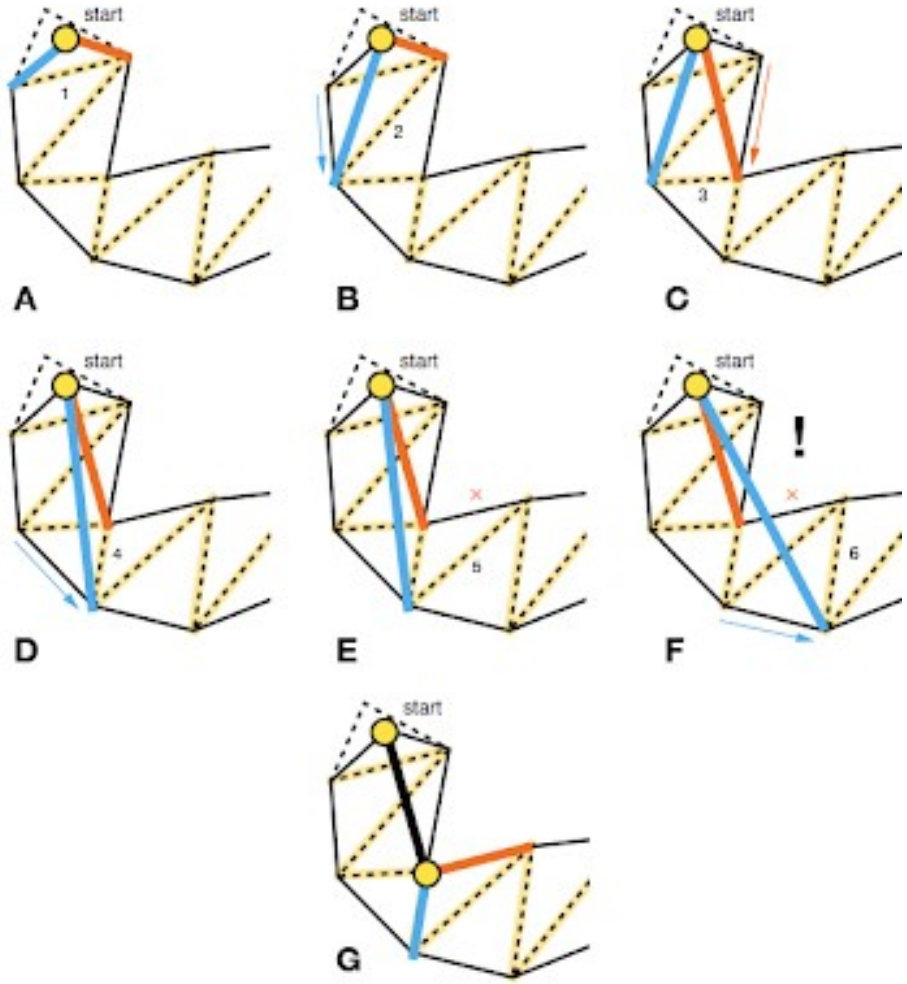
Environment Abstraction
Navigation Mesh – Funnel algorithm

- Simple Stupid Funnel Algorithm (2010)
  - from Mikko Mononen, author of Recast
  - published in a blog post
  - much simpler than previous funnel algorithms





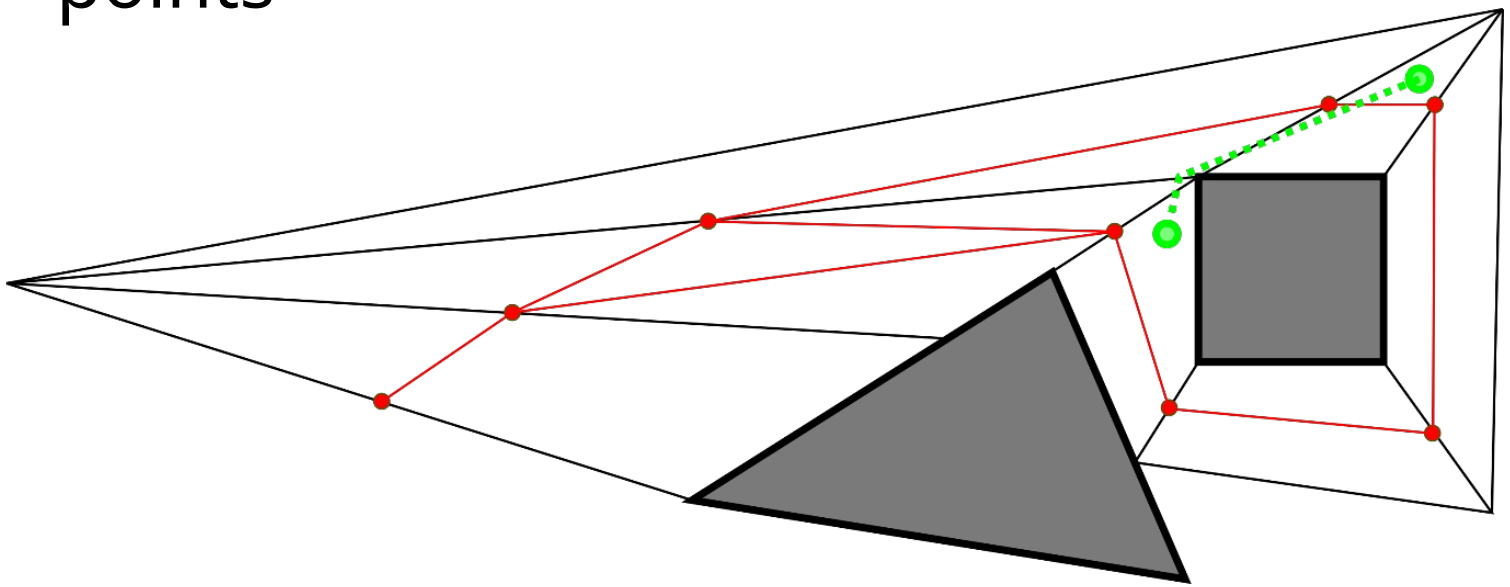
Environment Abstraction
Navigation Mesh – Funnel algorithm



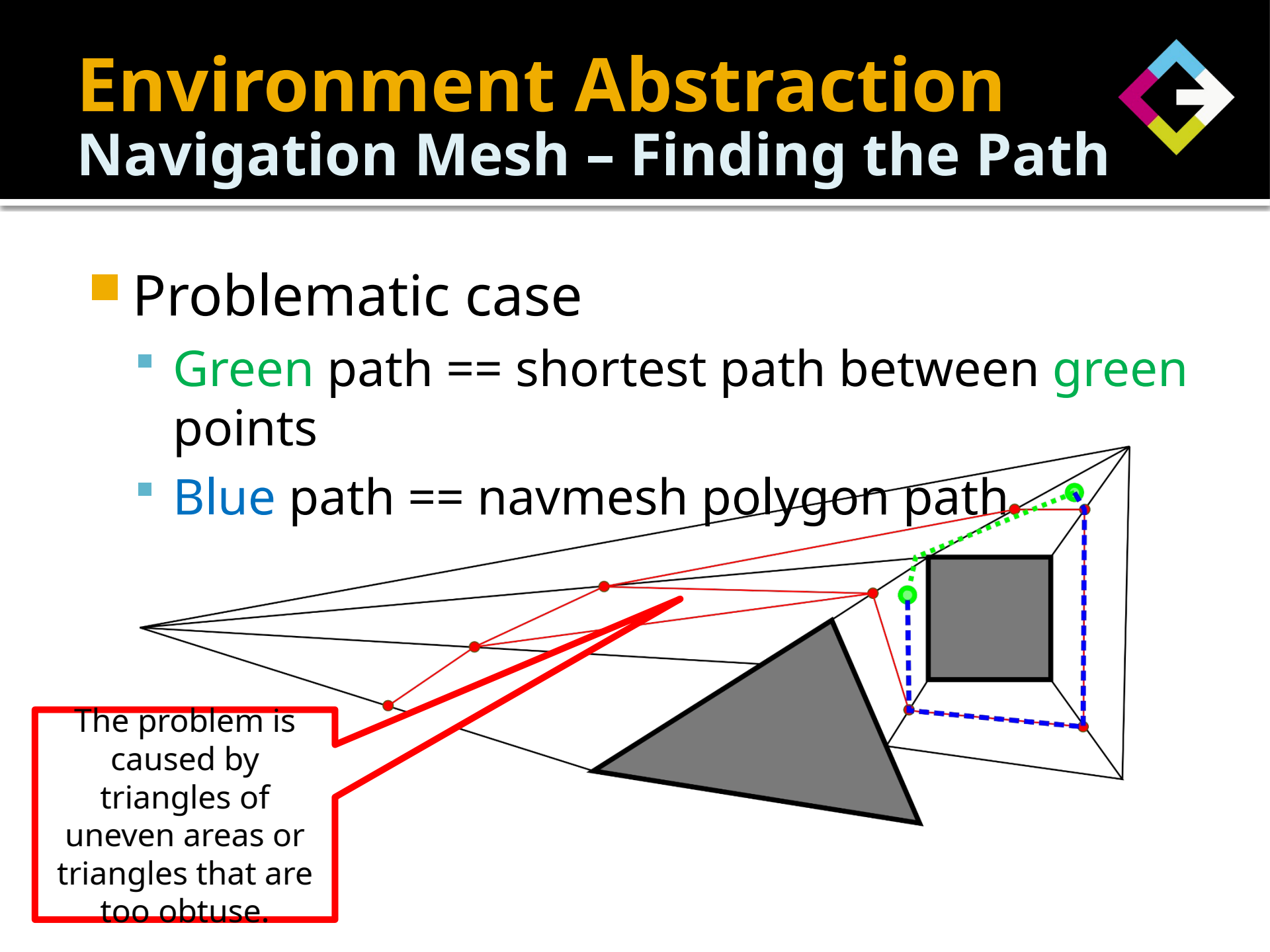
- On each iteration, we move to the next polygon and advance the left and/or right points
- If a point is *inside* the funnel (A-D), we simply advance it
- If a point is outside the funnel *on its own side* (E), we leave it in place
- If a point is outside the funnel *on the other side* (F), we add the other point to the path and restart the algorithm from that point (G)

Environment Abstraction
Navigation Mesh – Finding the Path

- Problematic case
  - Green path == shortest path between green points



Environment Abstraction
Navigation Mesh – Finding the Path



- Problematic case
  - Green path == shortest path between green points
  - Blue path == navmesh polygon path

The problem is caused by triangles of uneven areas or triangles that are too obtuse.

Environment Abstraction
Navigation Mesh – Finding the Path

- Demo: pathfinding in Godot

Environment Abstraction
Navigation Mesh – Finding the Path

- Navigation meshes in a larger game
- video "Death Stranding:
  An AI Postmortem" (YouTube)