Crowds In A Polygon Soup

Next-Gen Path Planning

David Miles
3/23/2006

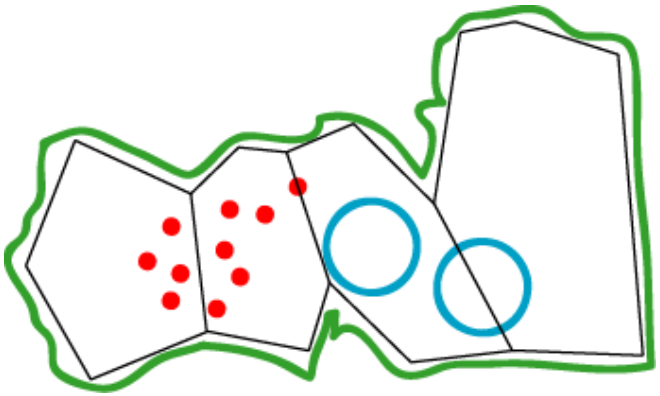
“Large-Scale Fluid AI Navigation in Complex, Changing Worlds”

1. Build a “usable free space” nav graph
2. Update graph for dynamic obstacles
3. Use graph for fluid navigation

1. Building the Nav-Graph

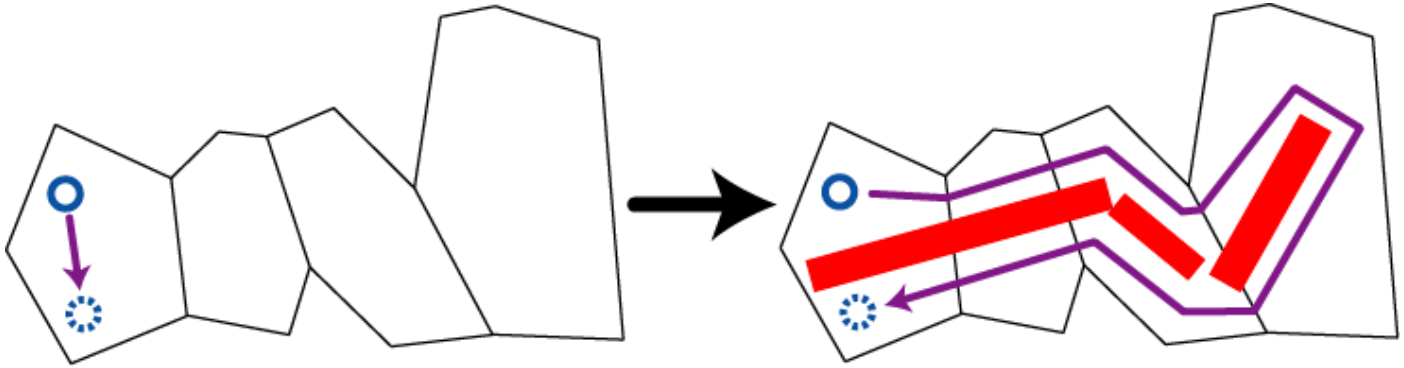
Automated Build Process

1. Large, Complex Worlds
2. Polygon Soup Mesh
3. Multiple Unit Sizes



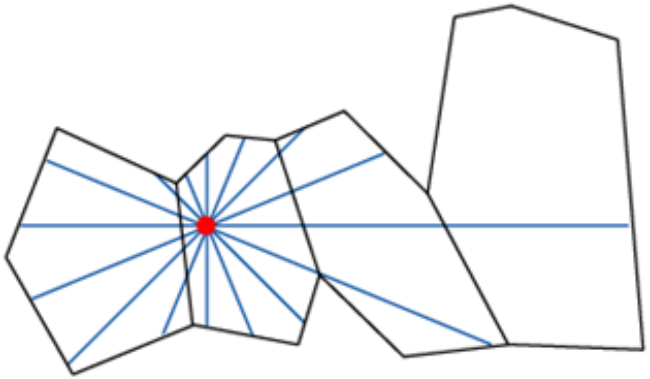
2. Dynamic Obstacle Updates

1. Obstacles reconfigure the world
2. Dynamic areas replace static areas
3. Underlying algorithms work exactly as before

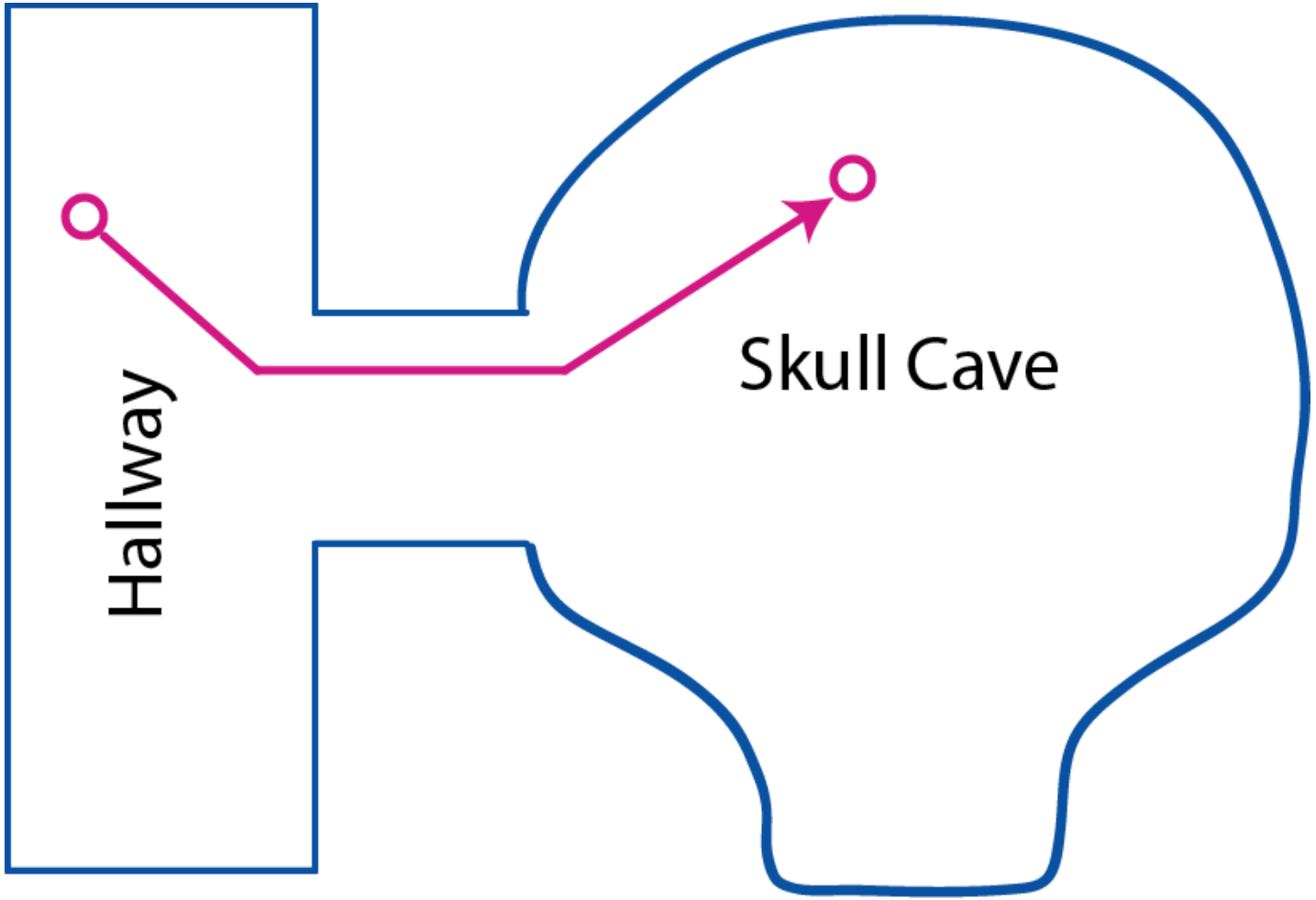


3. Fluid AI Navigation

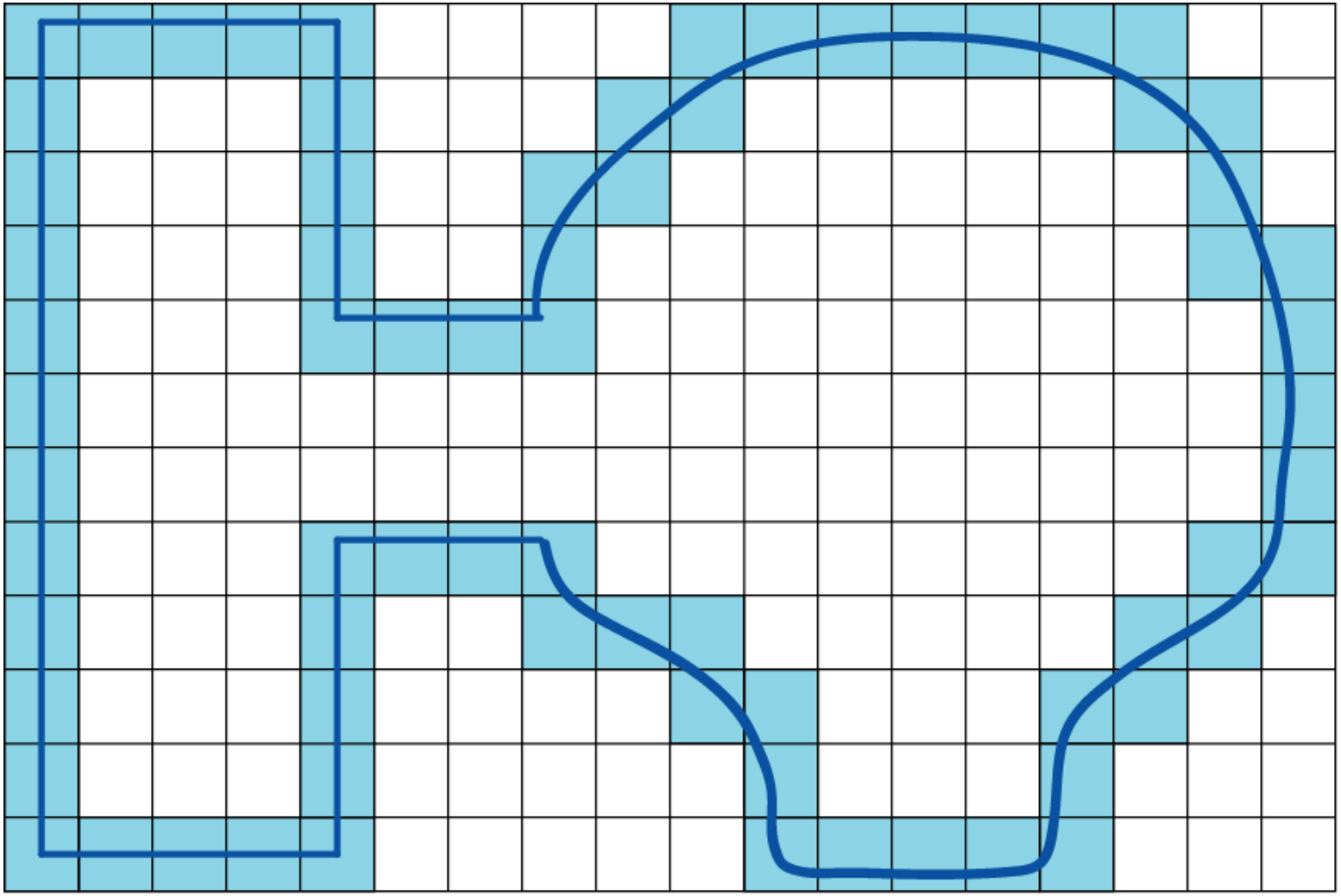
1. Fast ray casts
2. Edge detection
3. Potential Fields
4. Path Following



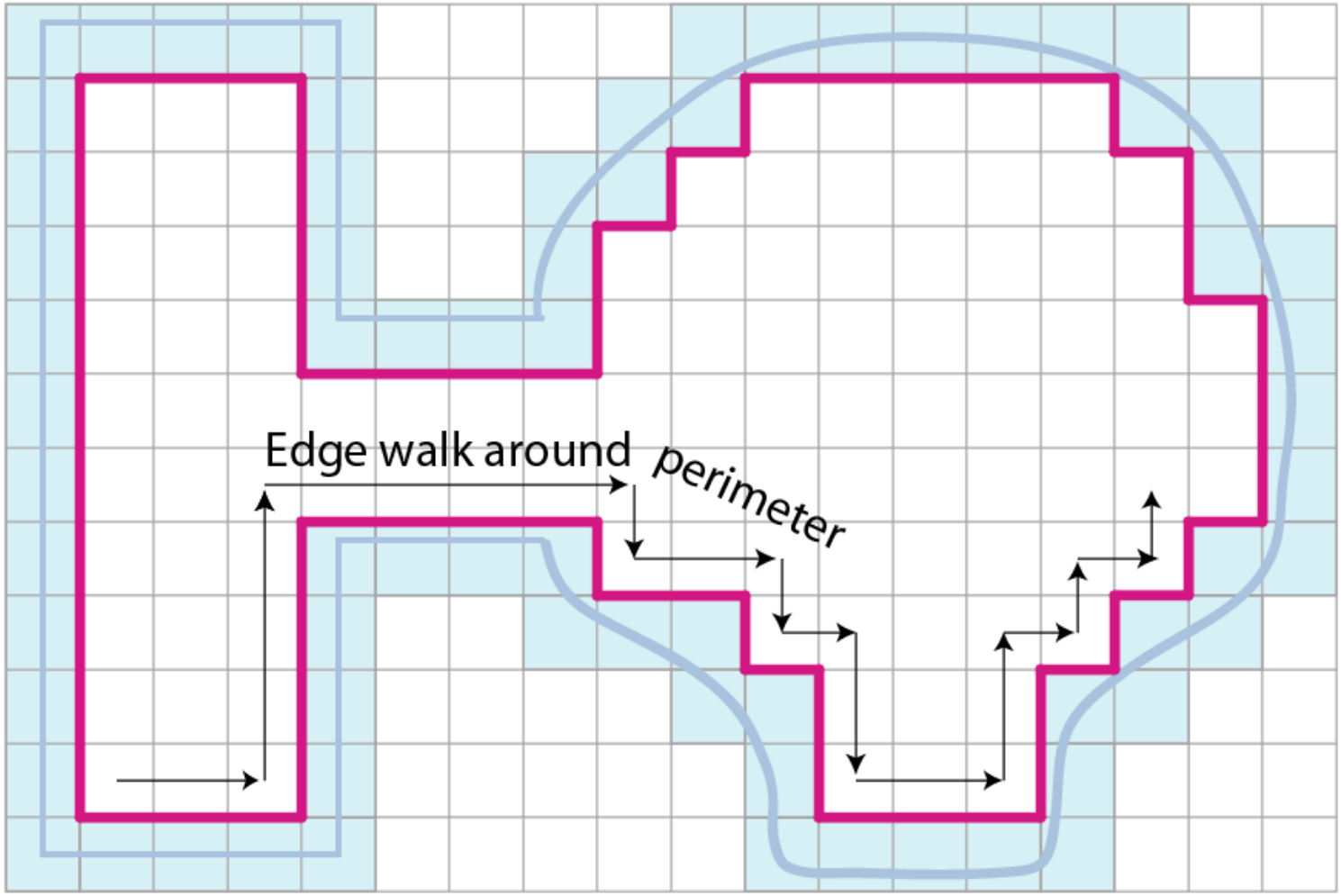
Automated Build Process



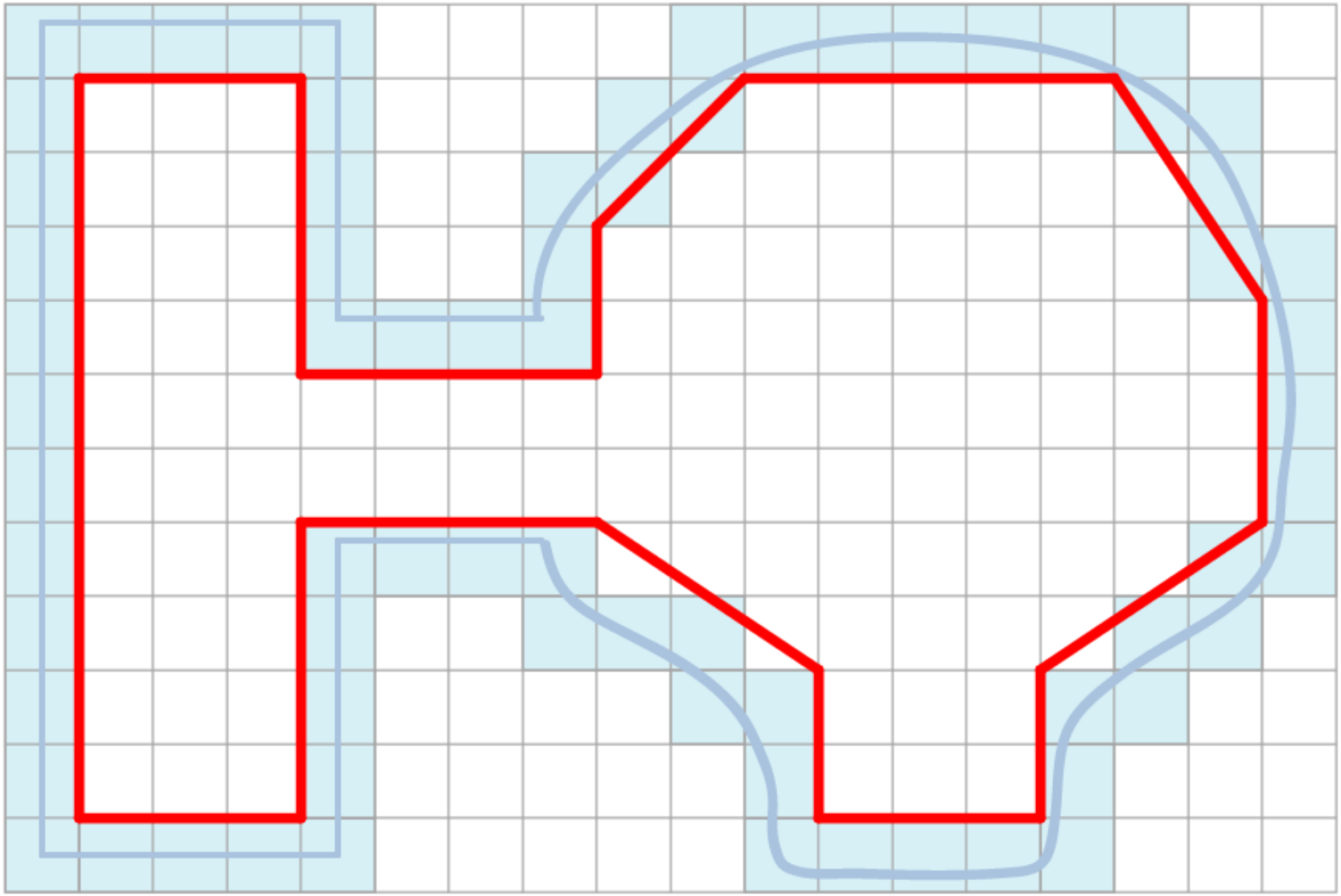
1. Voxelize Polygon Soup



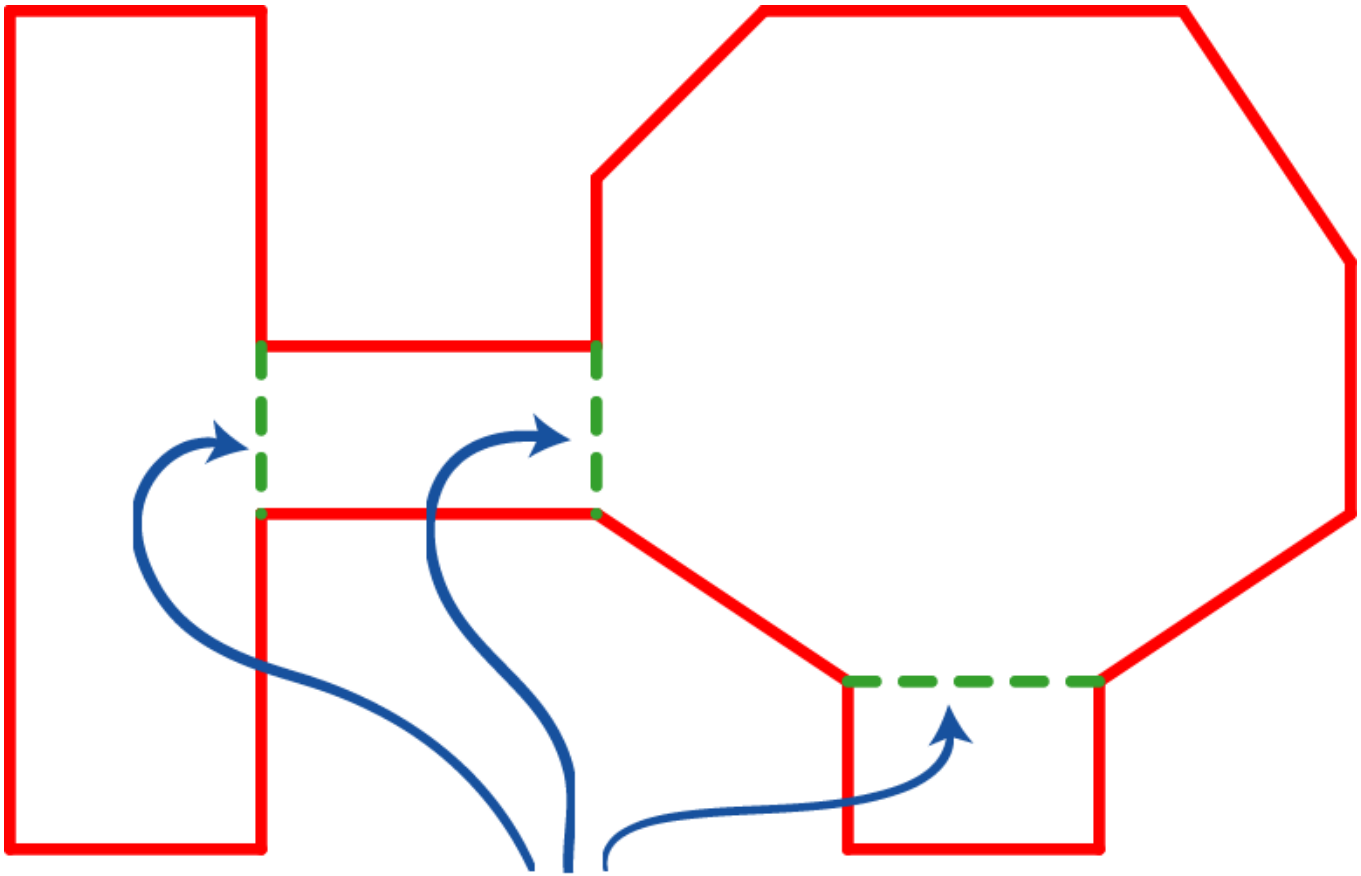
2. Extract Edge



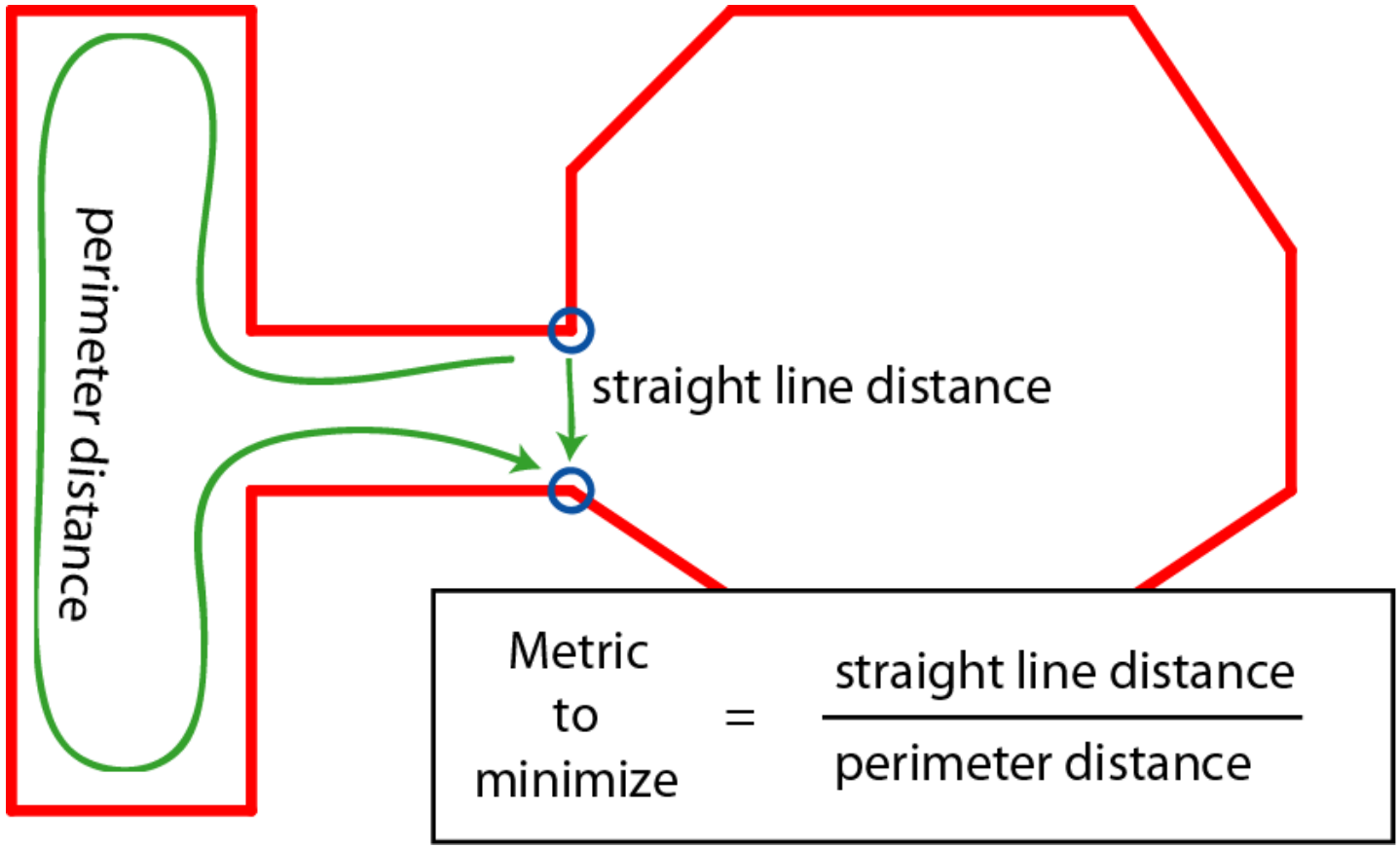
3. Simplify Polygon



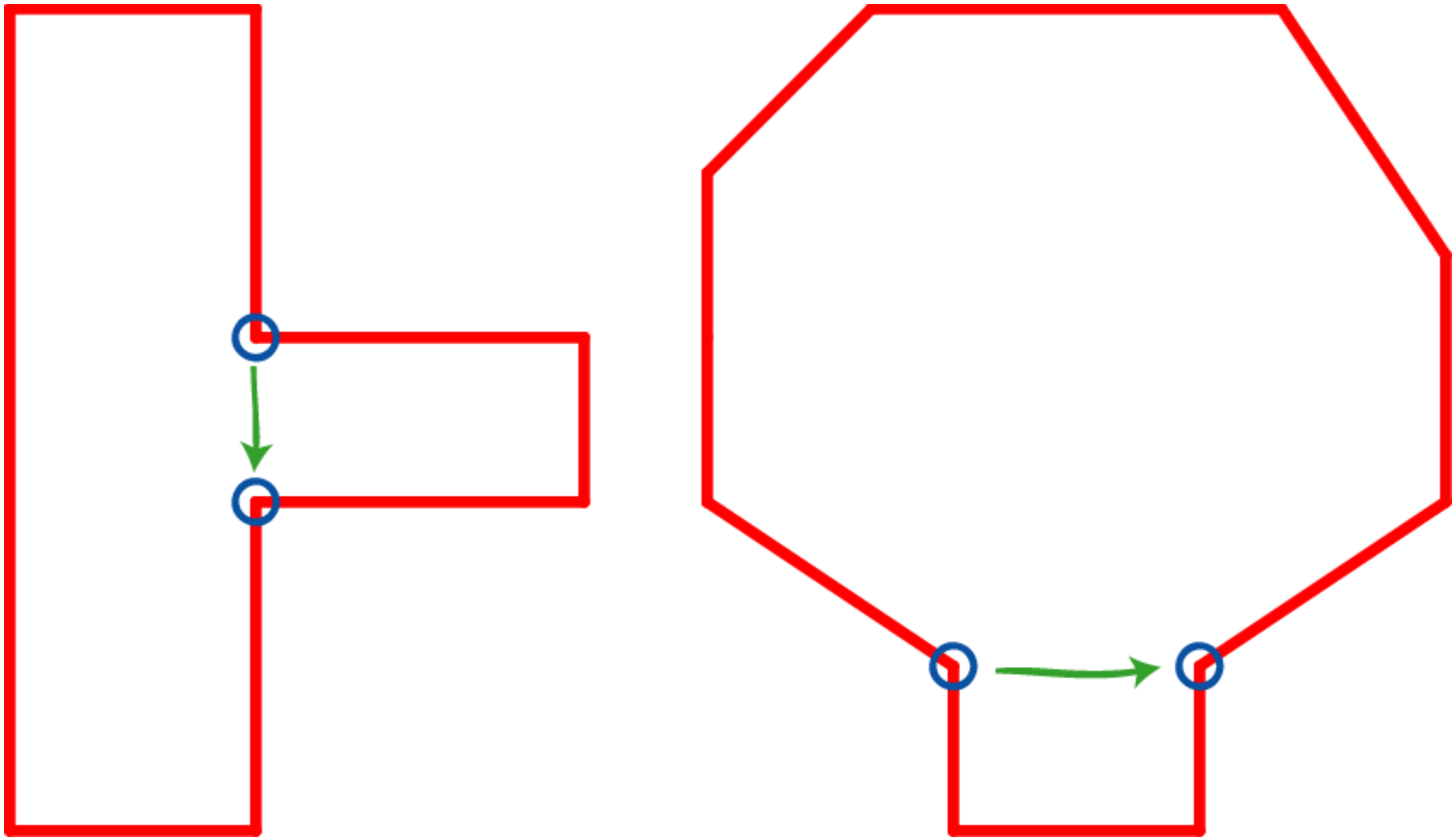
4. Partition Into Convex Areas



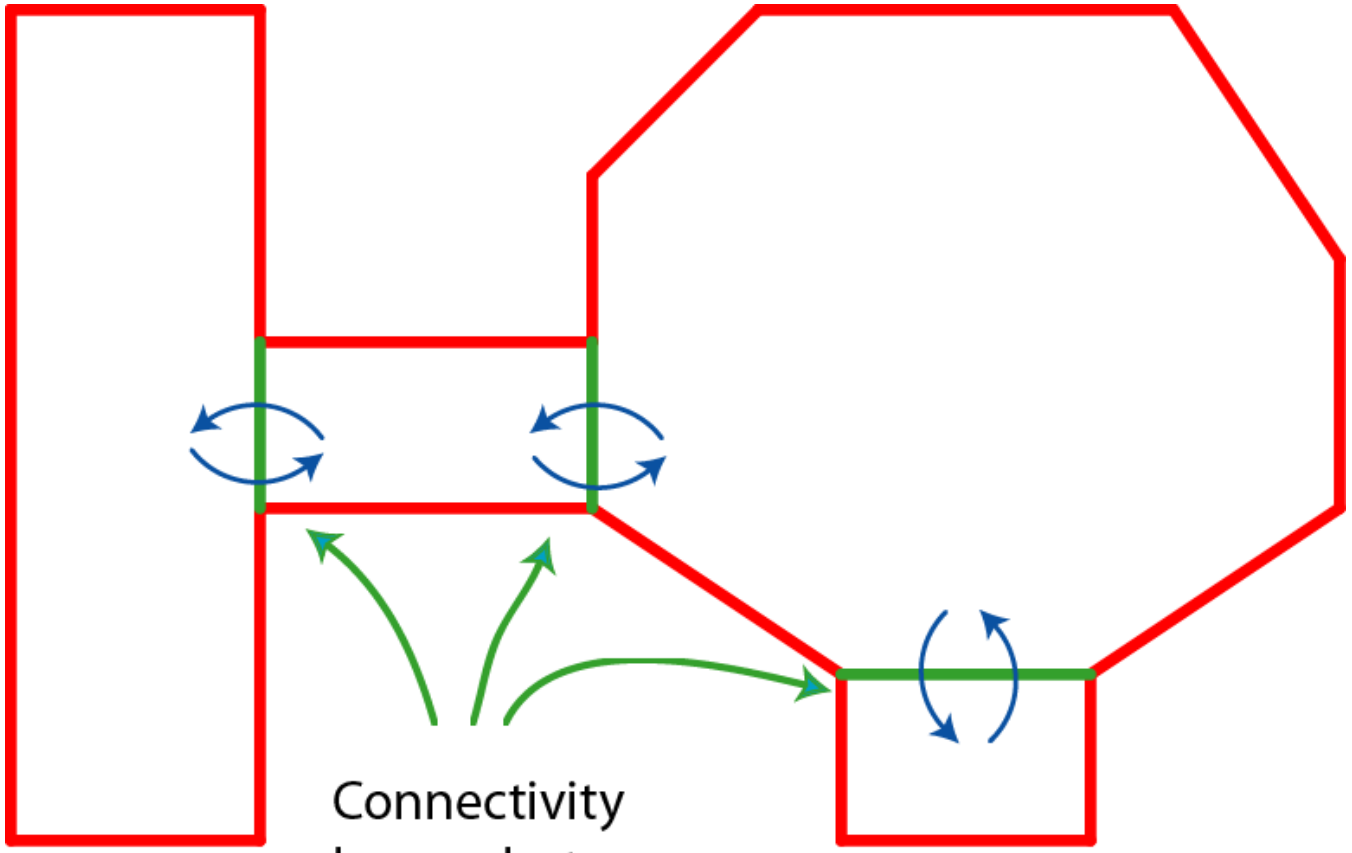
Convex Partitioning



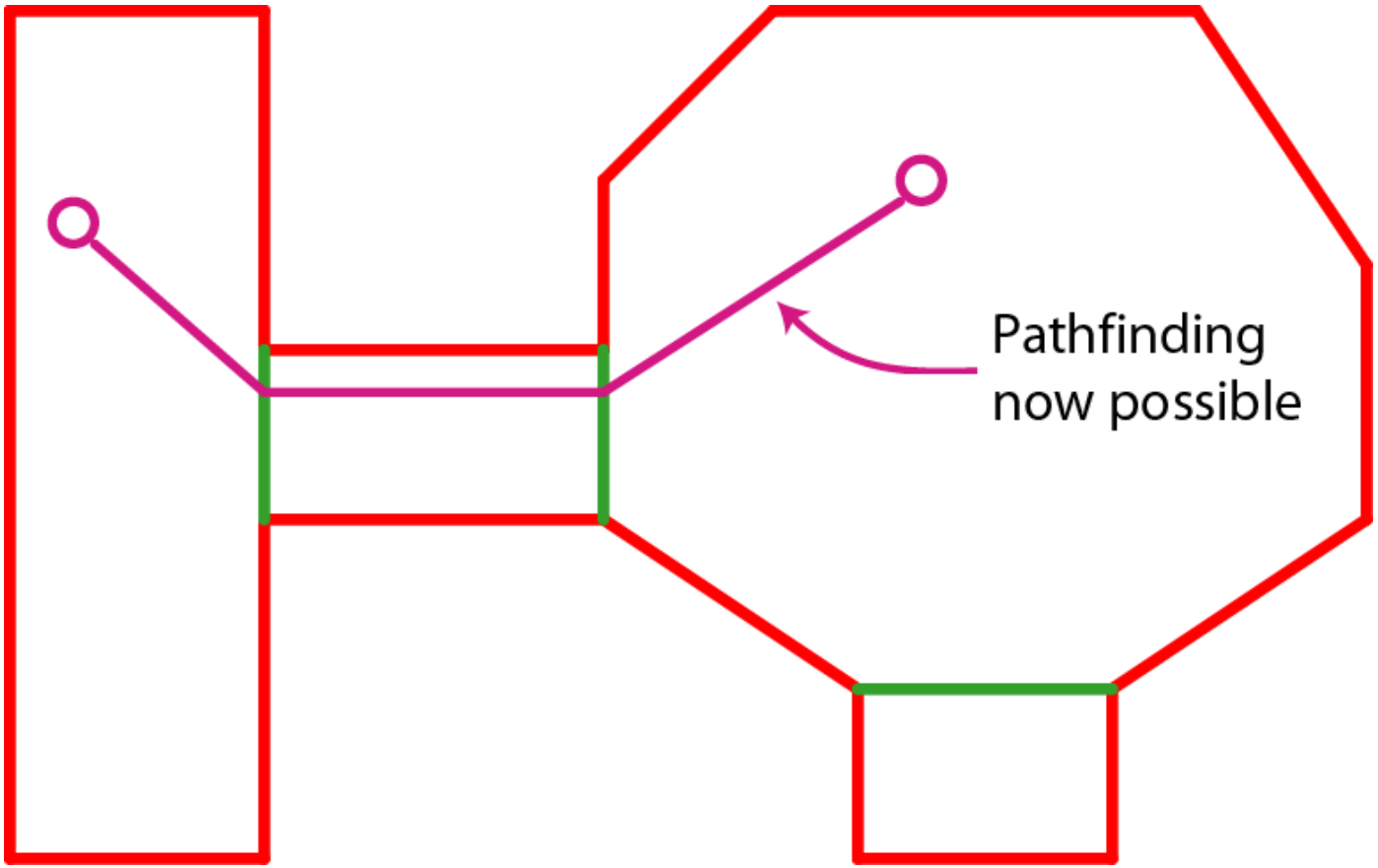
Recursively Partition



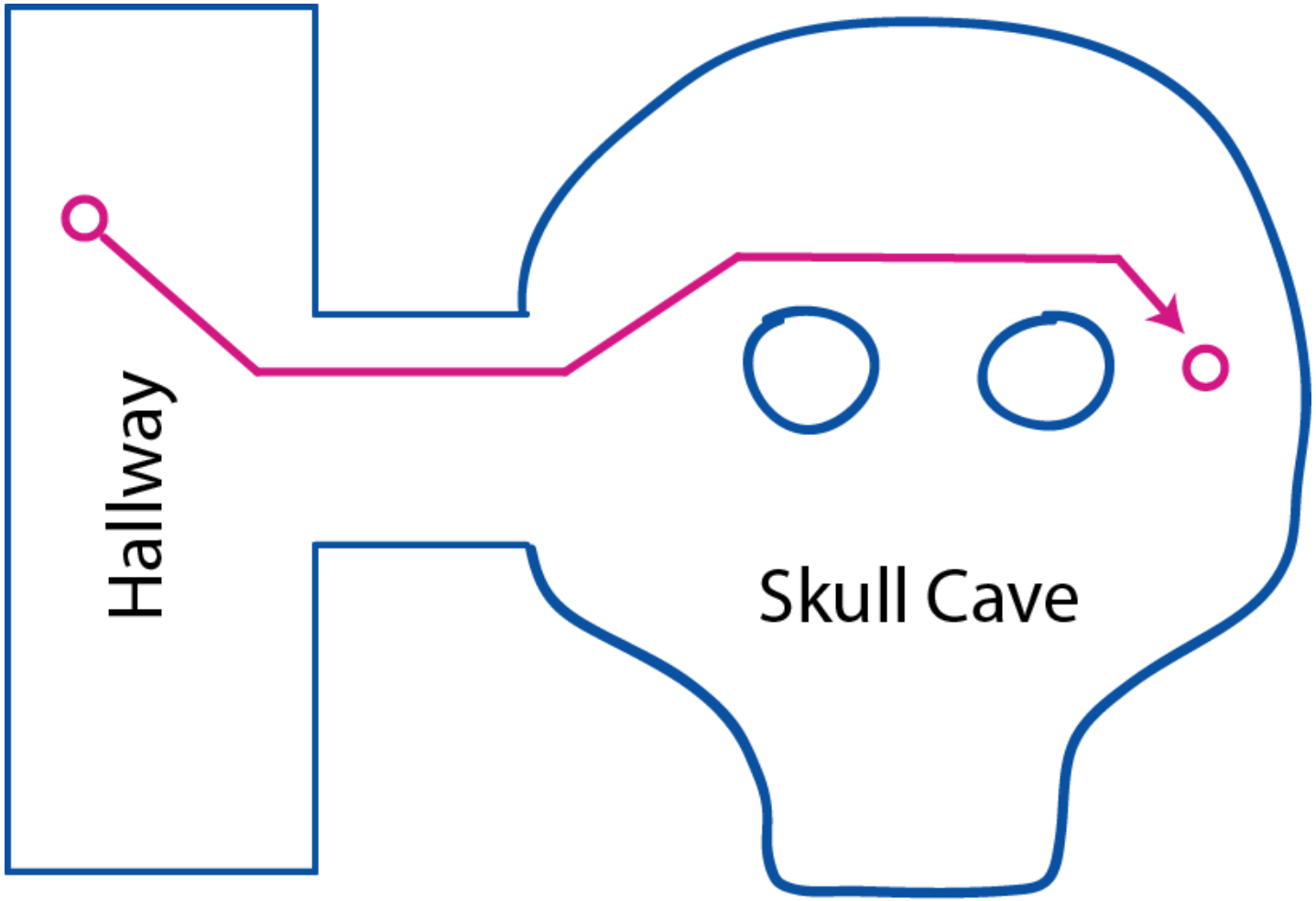
Convex Area Graph Complete



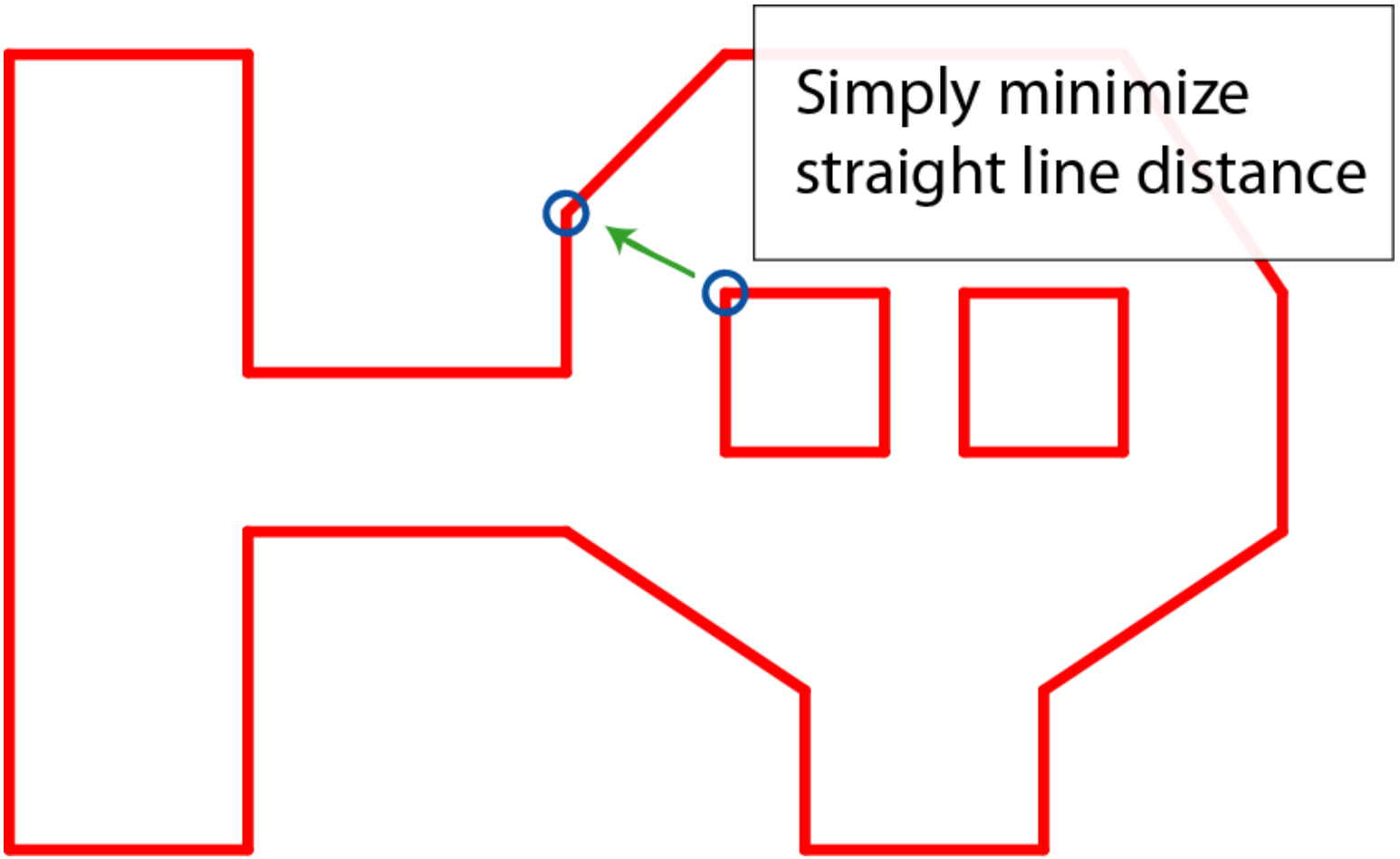
Pathfinding



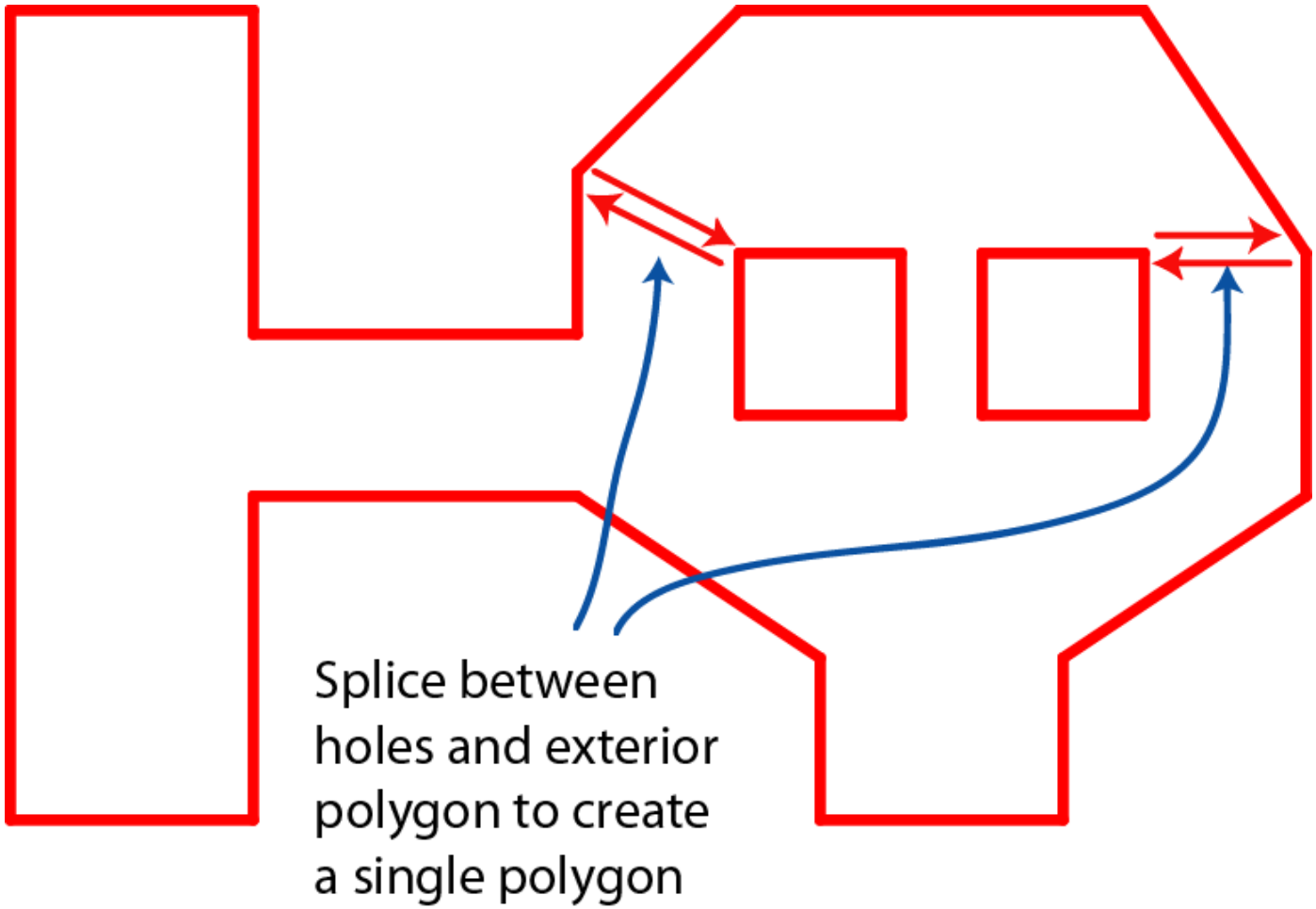
Holes in the Free-Space



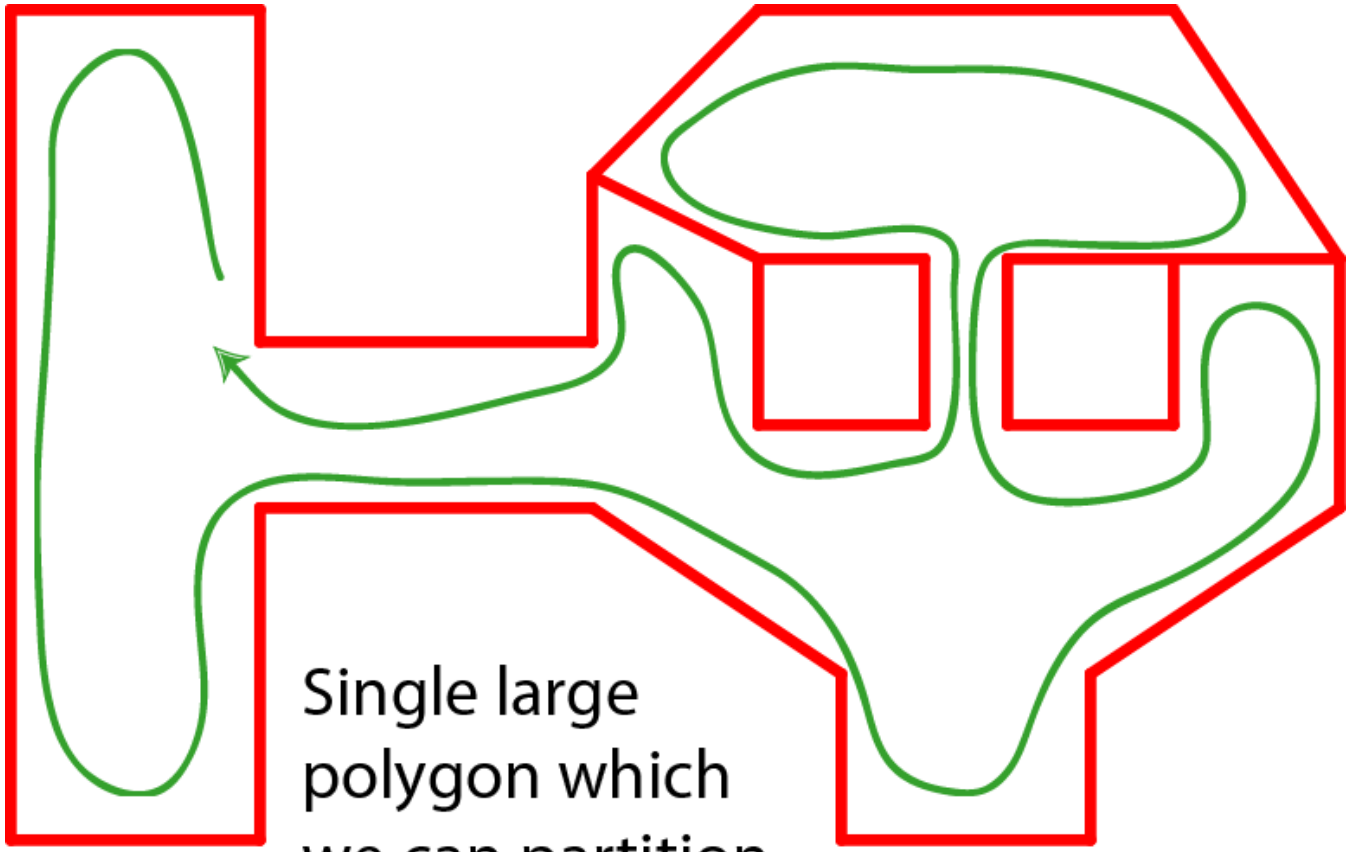
Splice Location



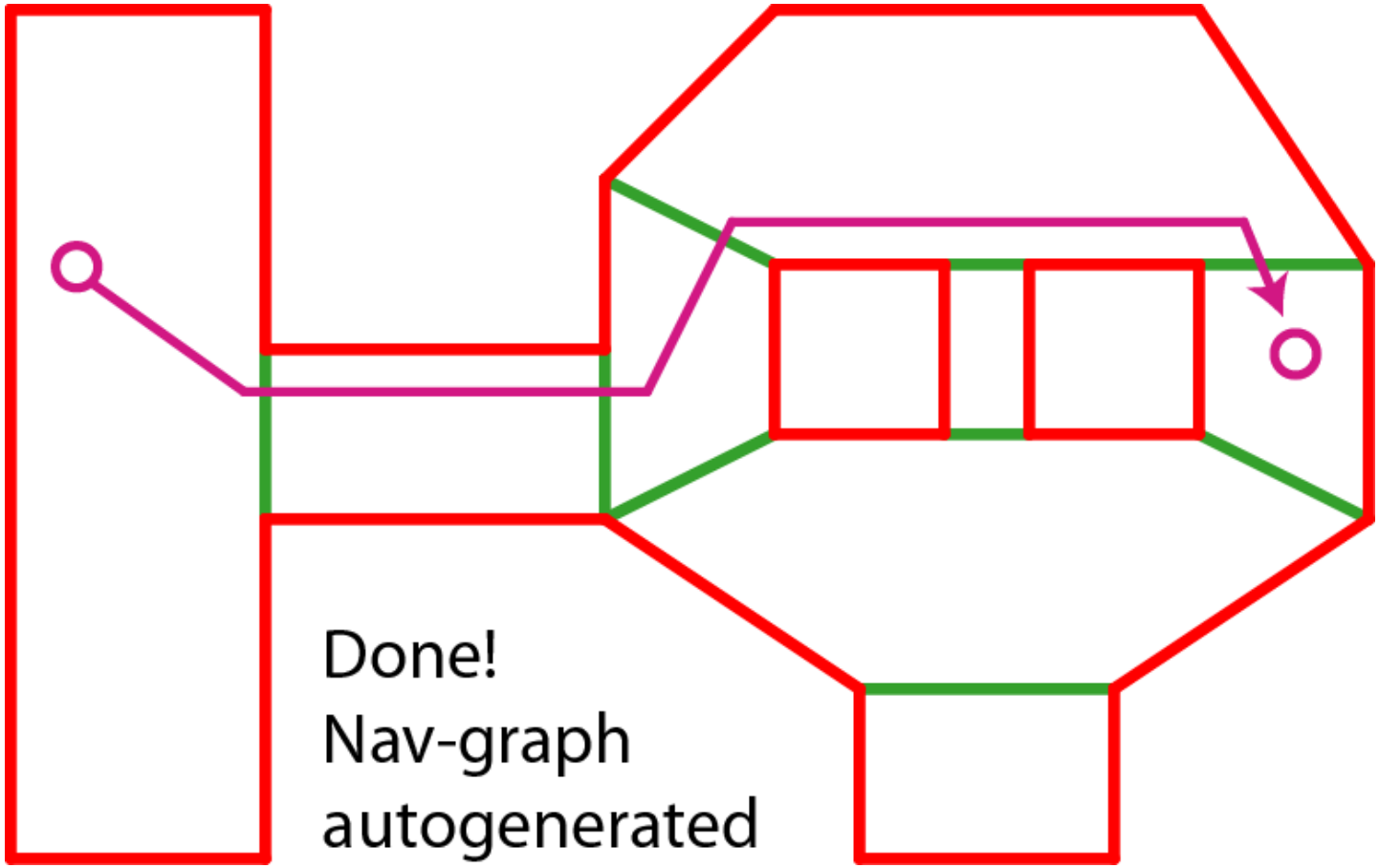
Add Splice Segments



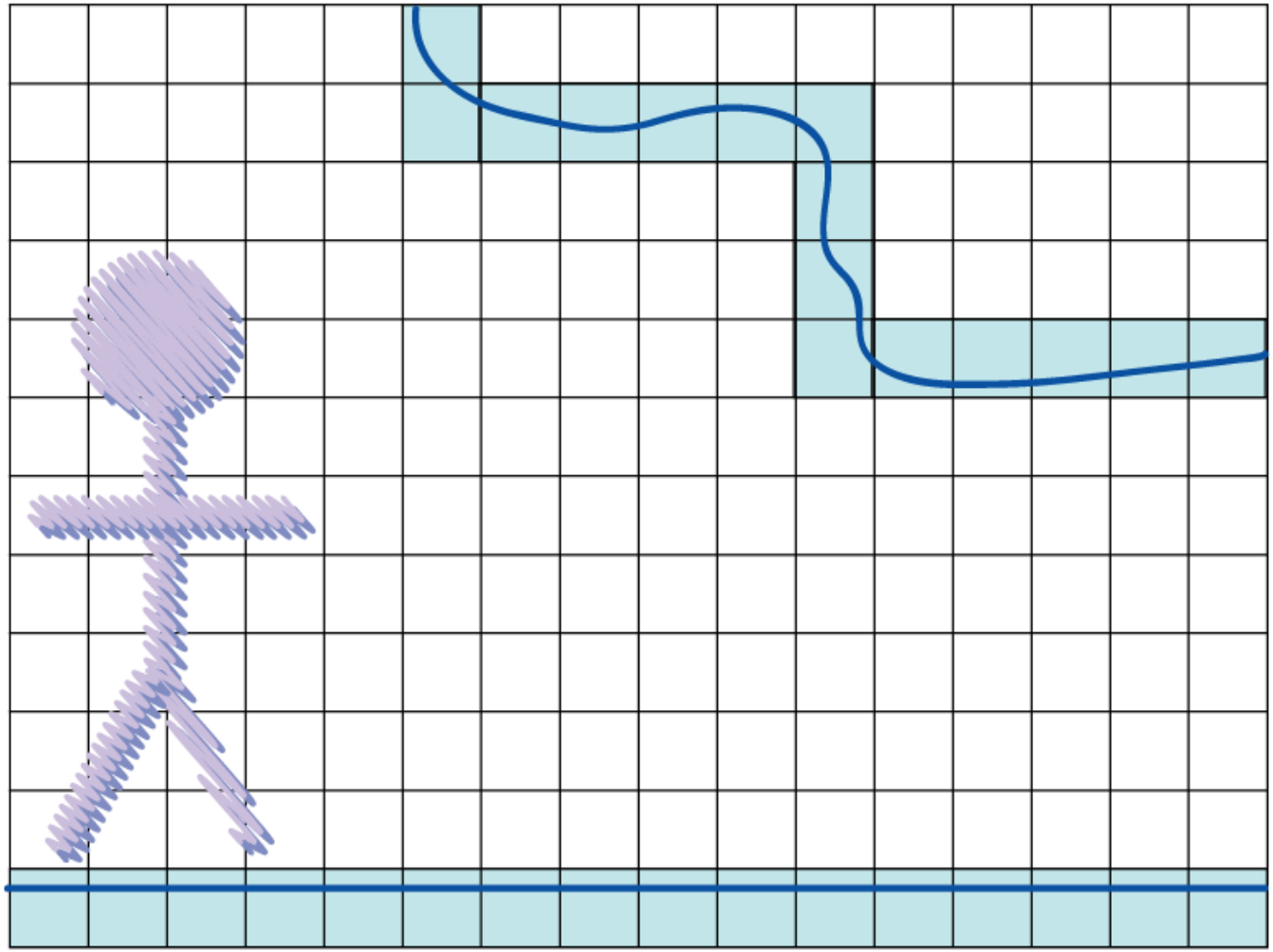
Single Polygon Surface



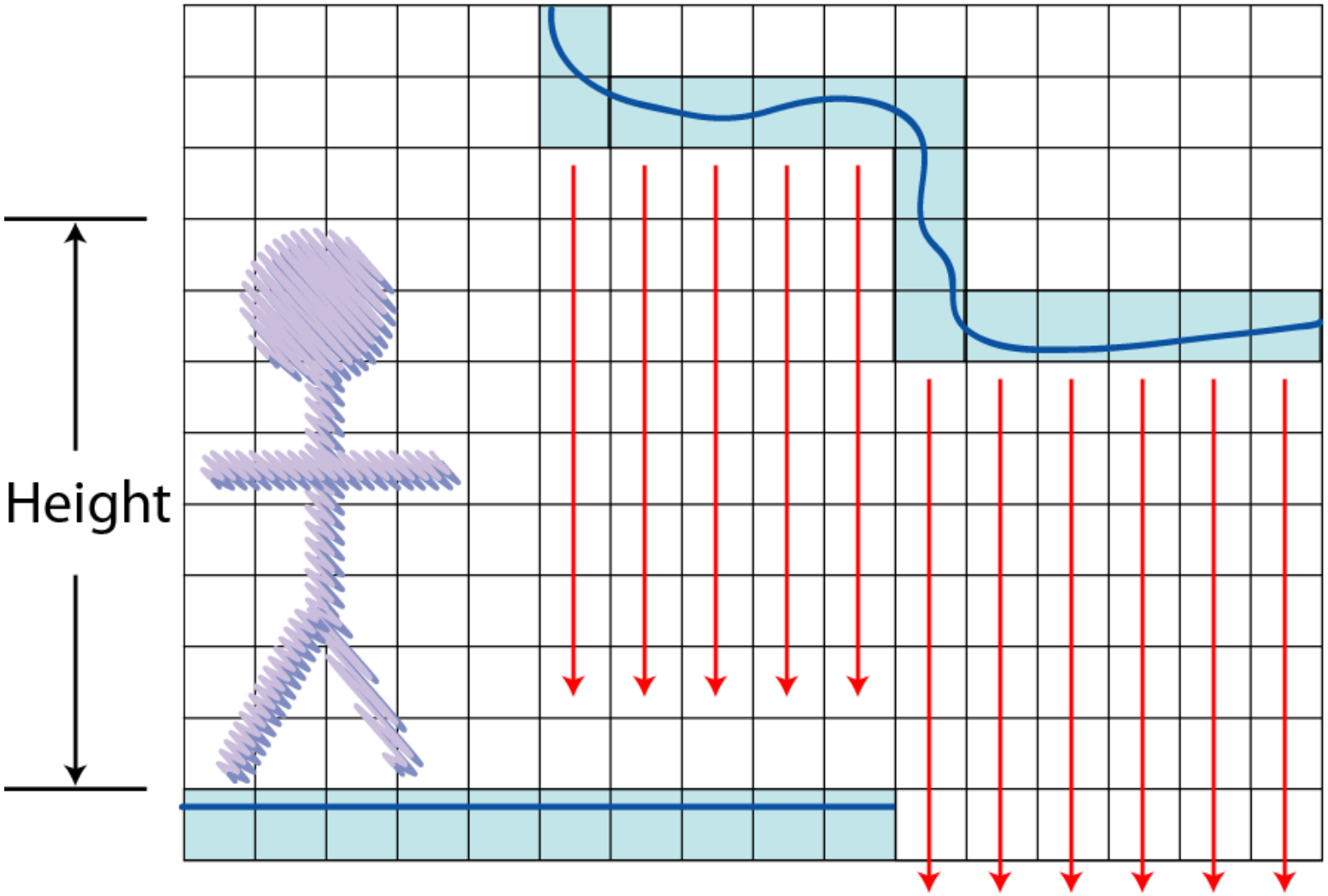
Convex Area Graph Complete



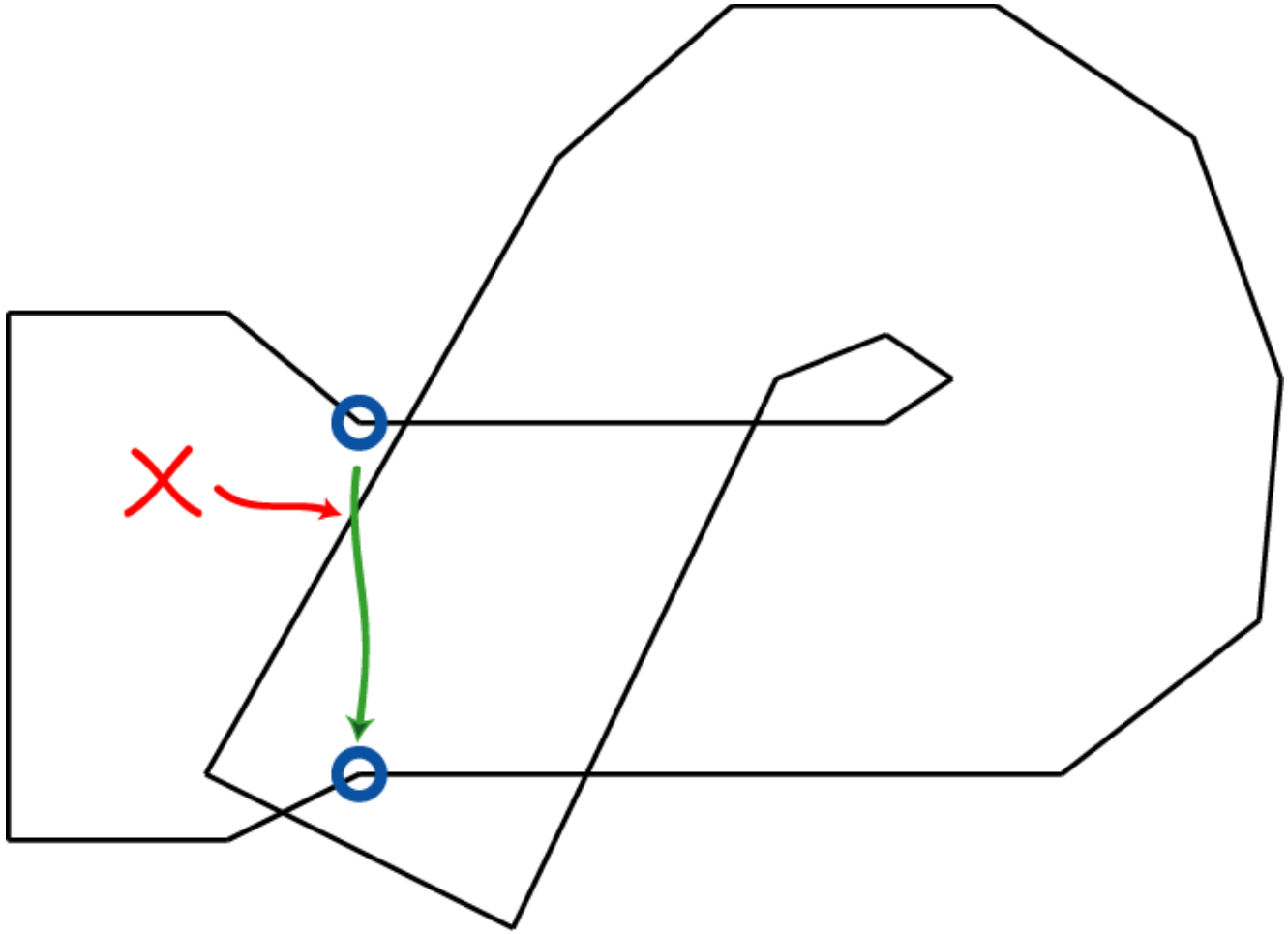
Overhead Obstacles



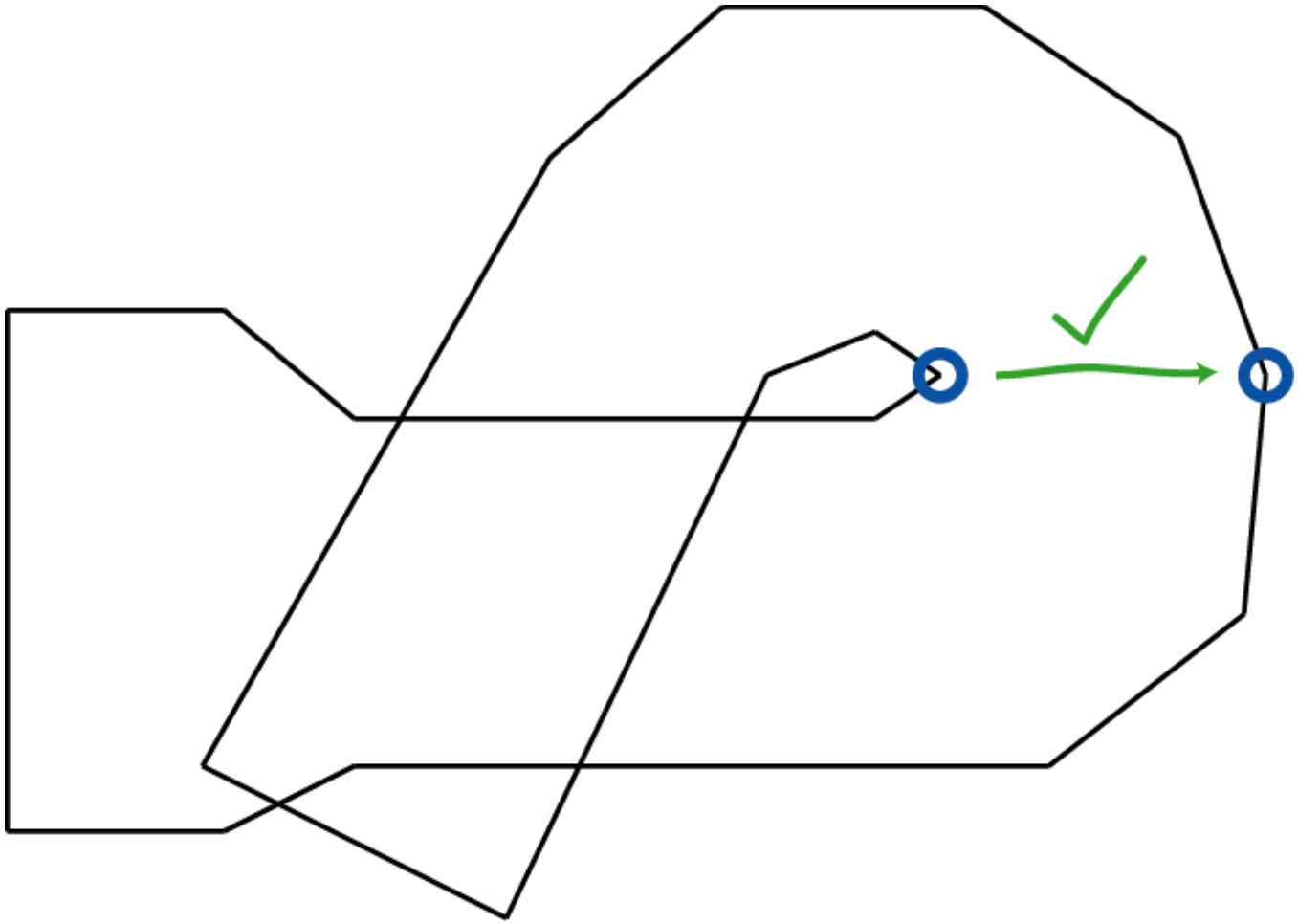
Walkable Surface Removal



3d overlap

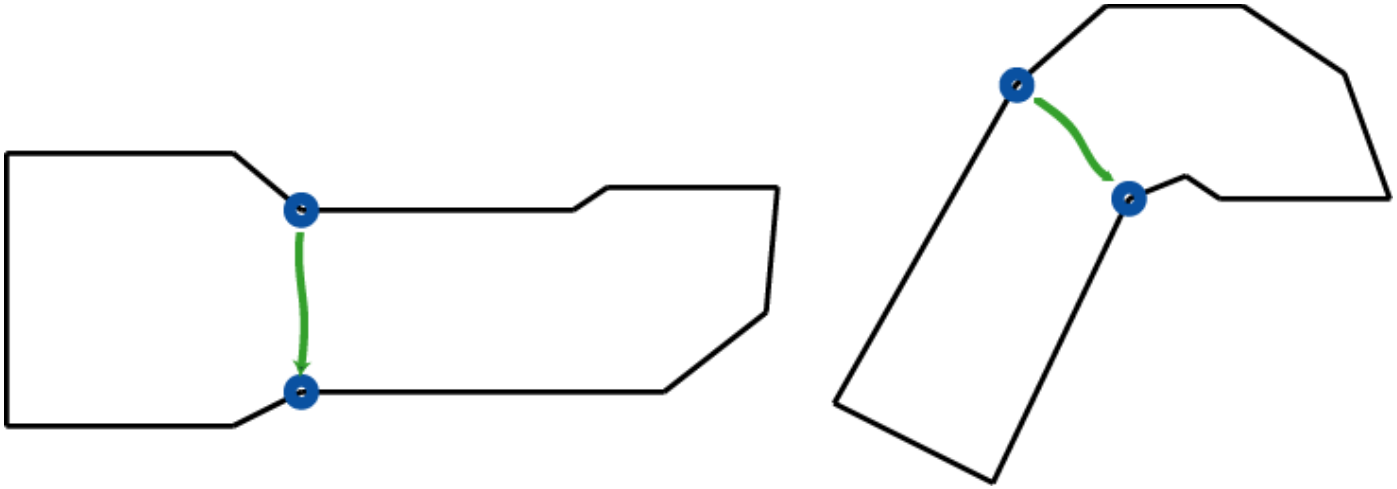


3d Partitioning

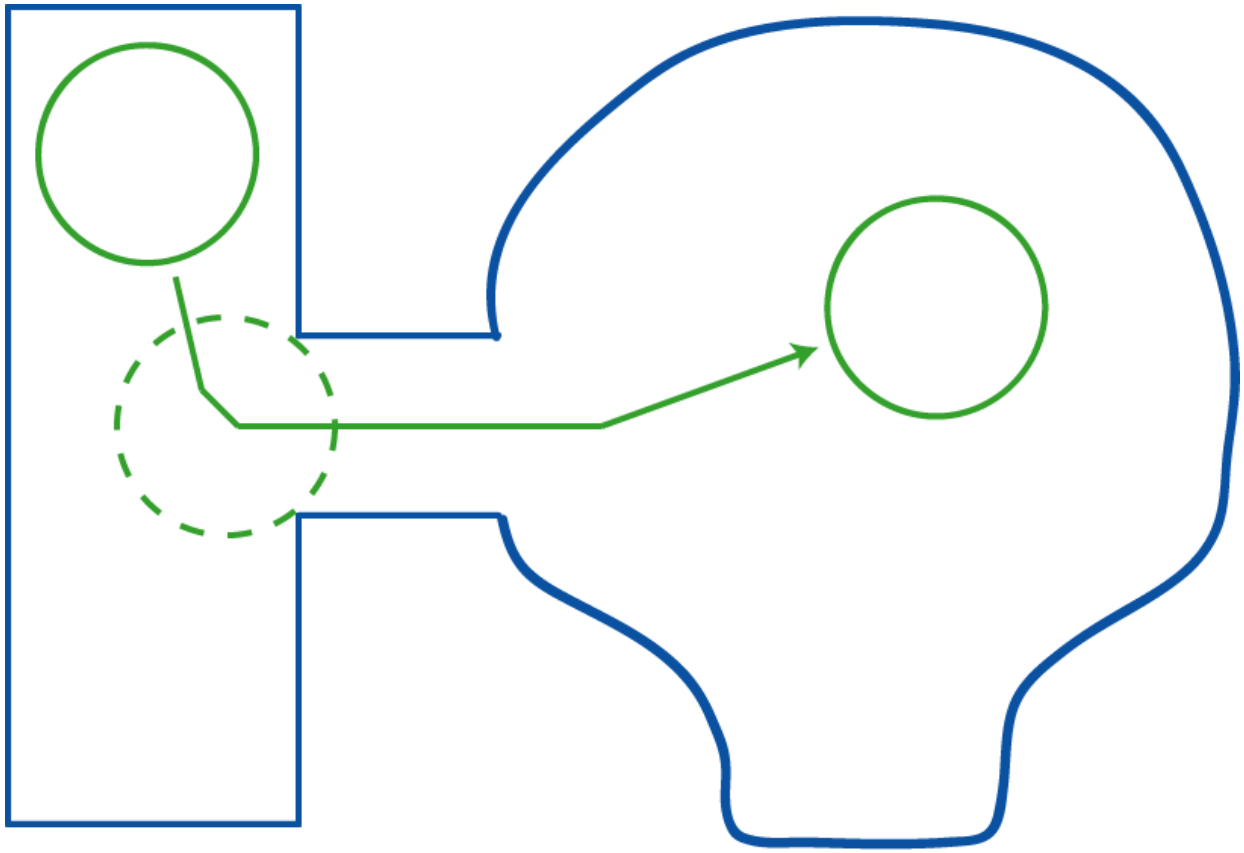


3d Partitioning

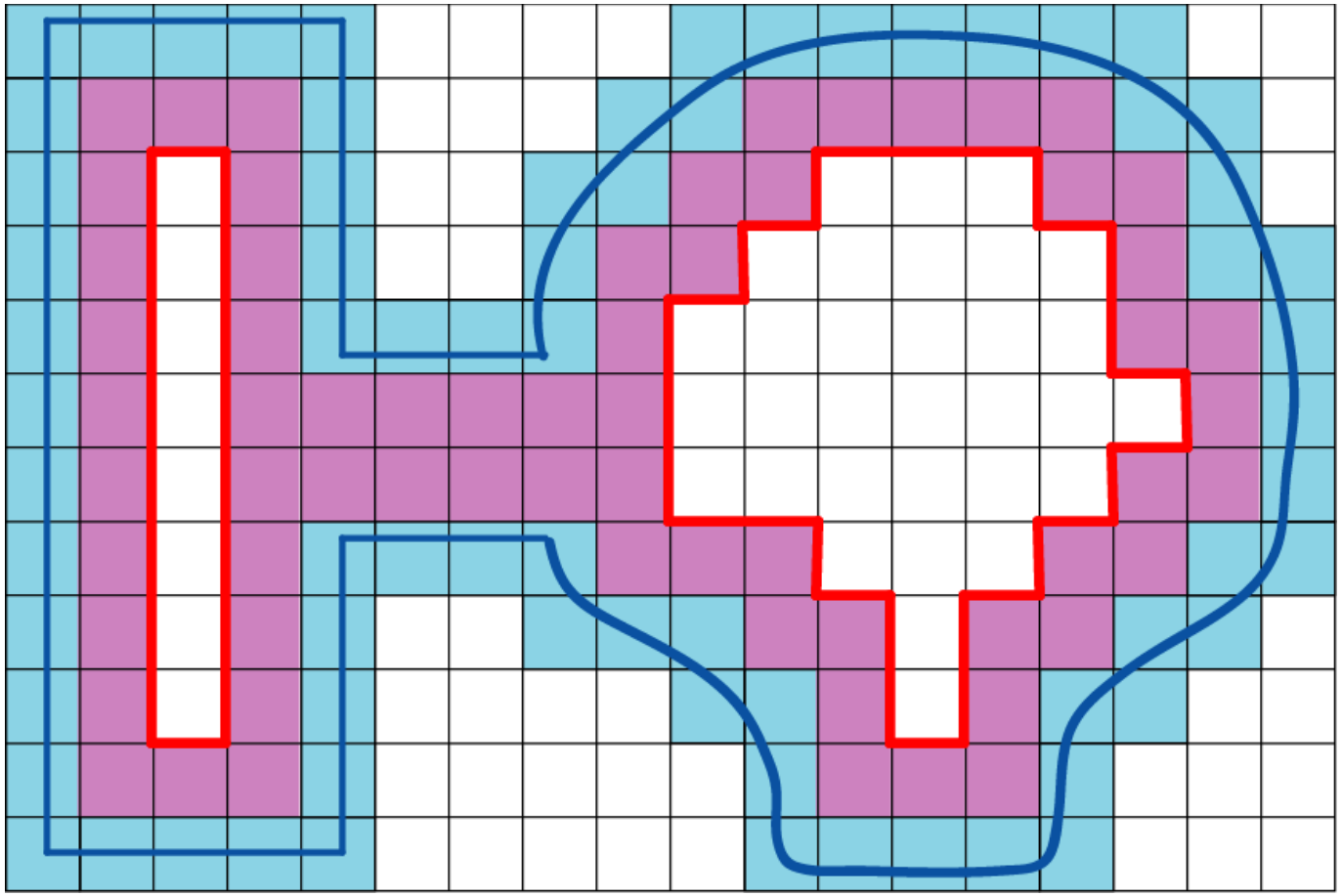
After initial split, overlap no longer exists



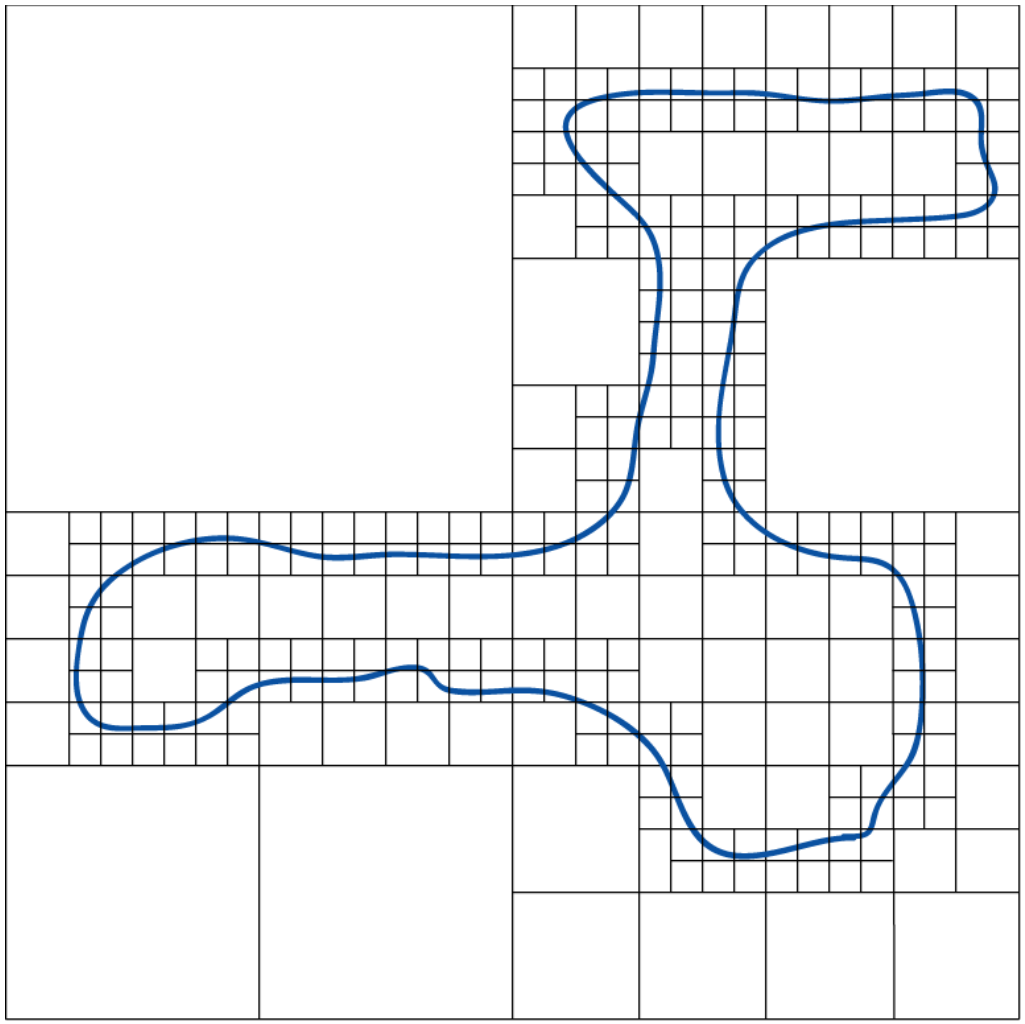
Different Creature Shapes



Different Creature Shapes

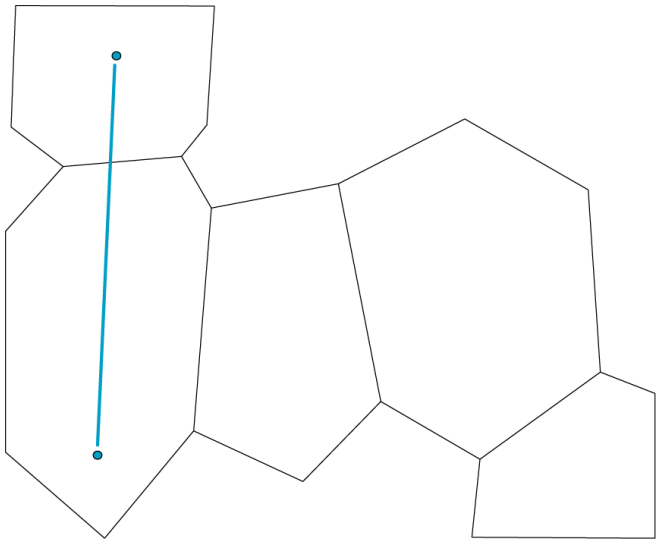


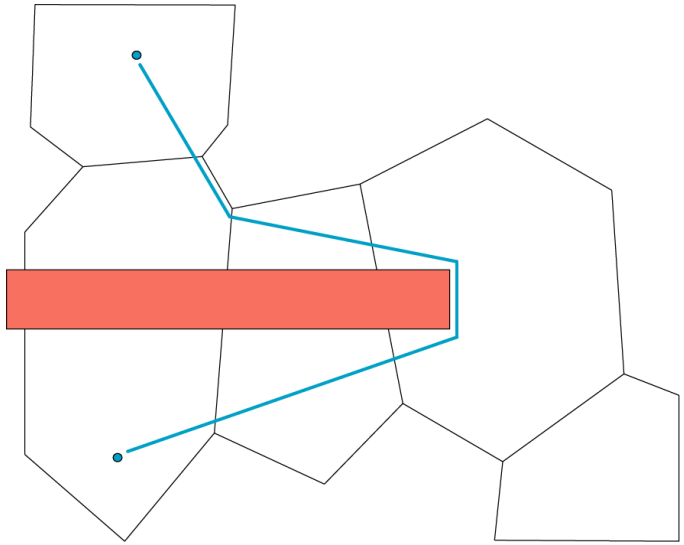
Non-Uniform Voxelization



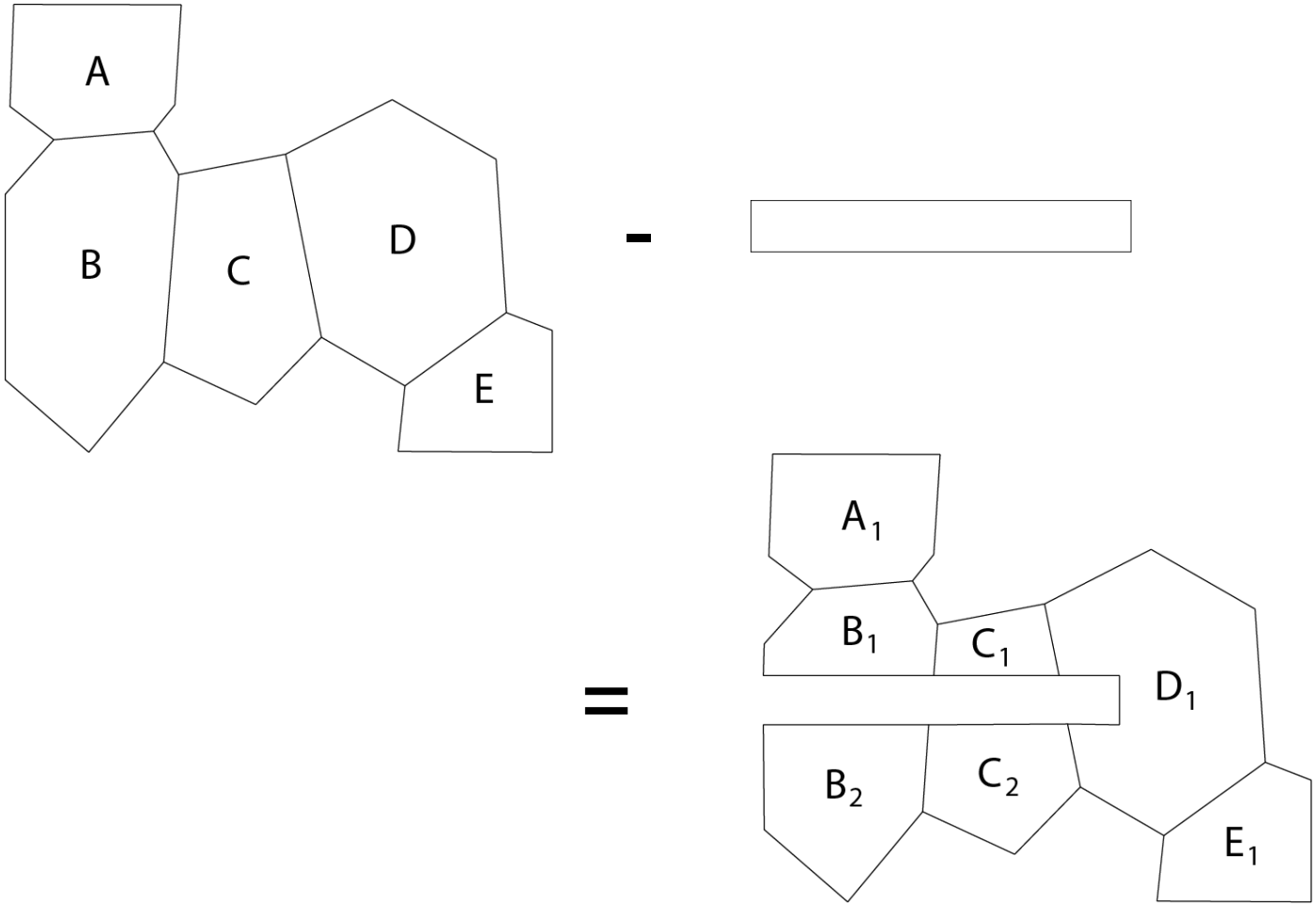
Part2: Dynamic Obstacles

- Dynamic obstacles reconfigure world
- Underlying algorithms unchanged

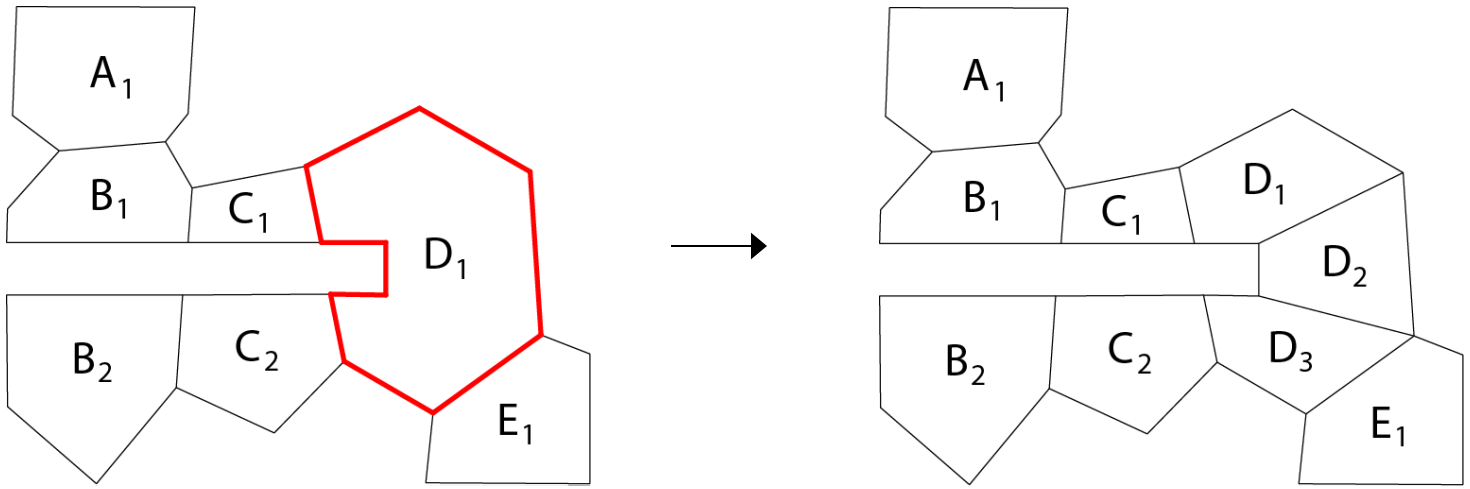




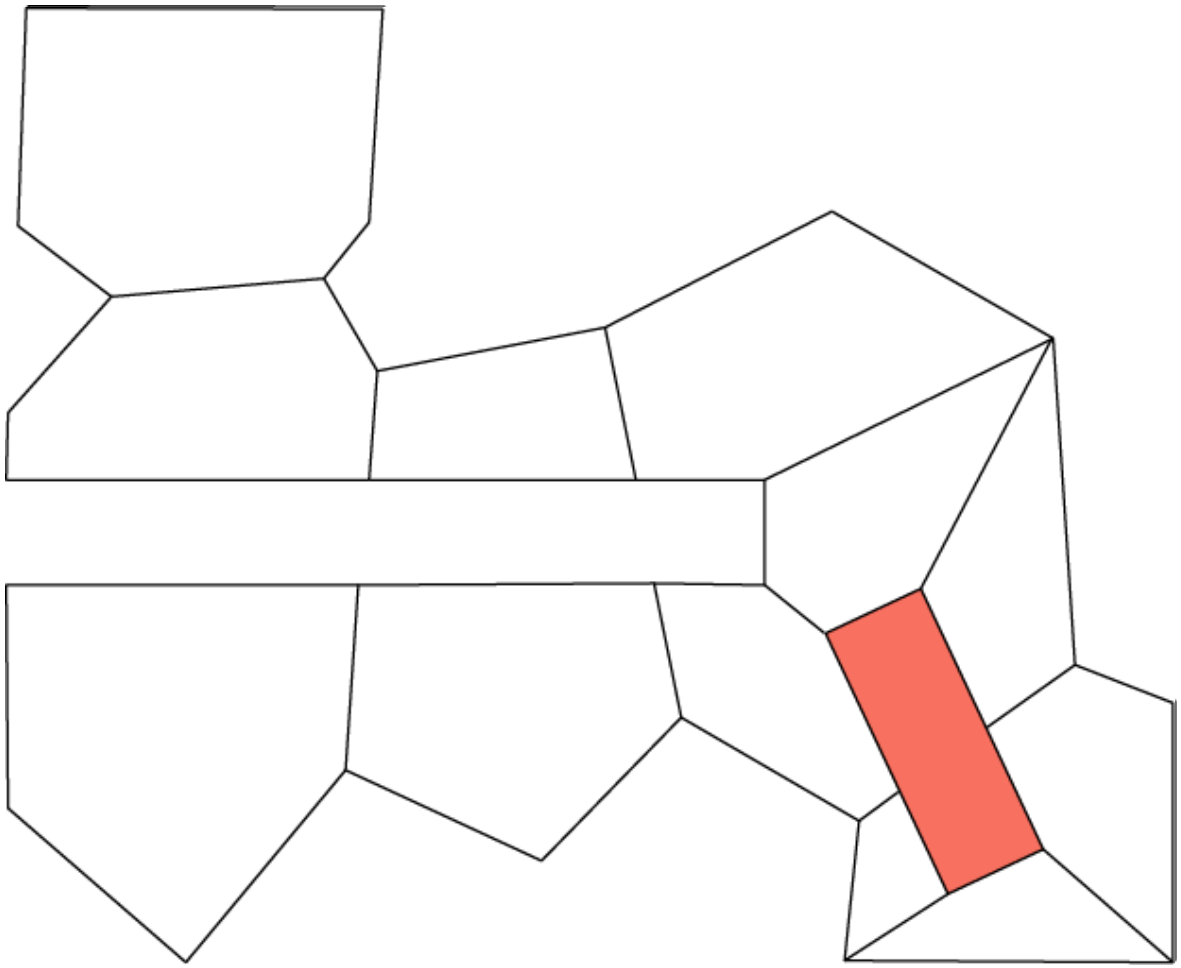
Boolean Subtract



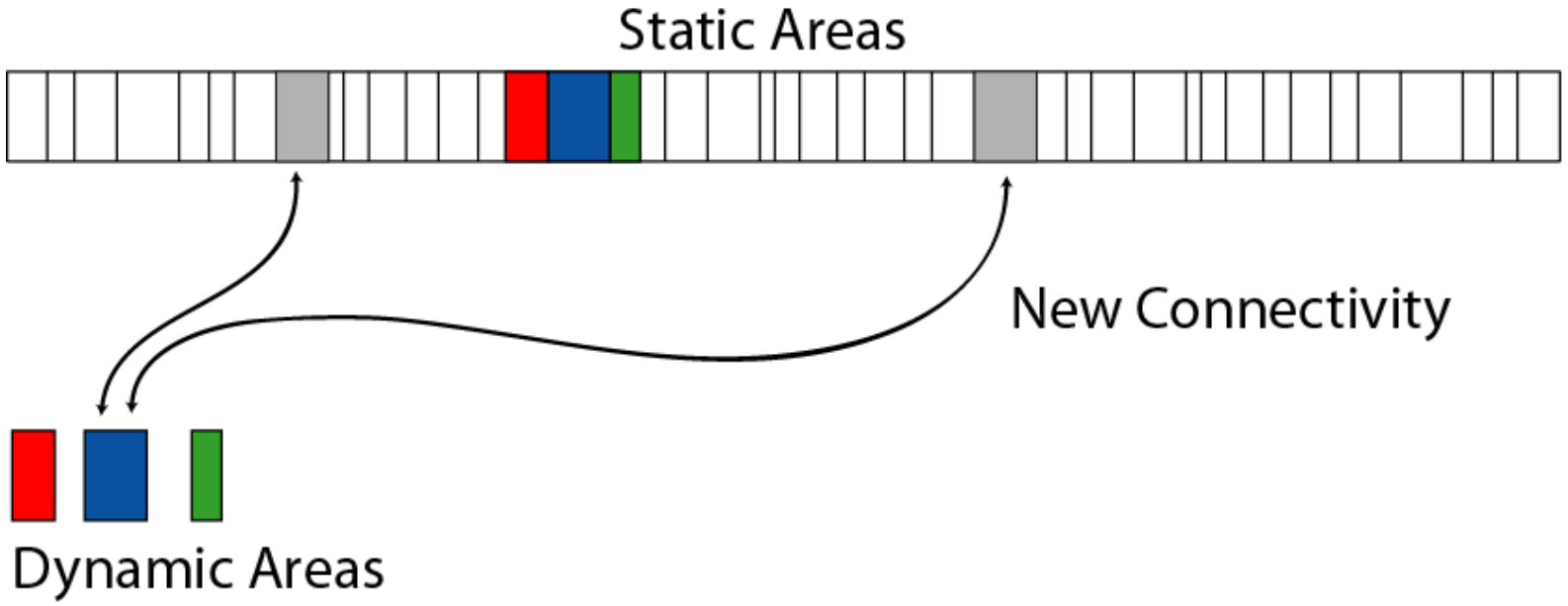
Partition Non-Convex Areas



Multiple Obstacles



Memory Layout

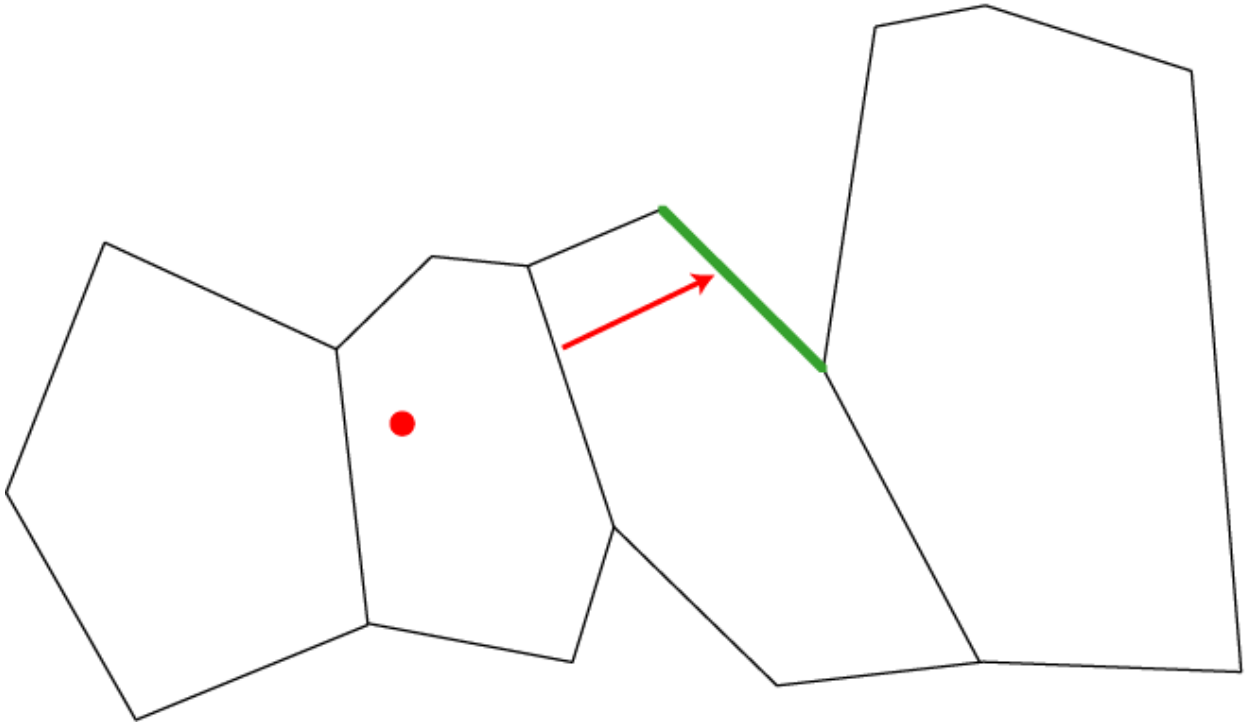


Part3: Fluid AI Navigation

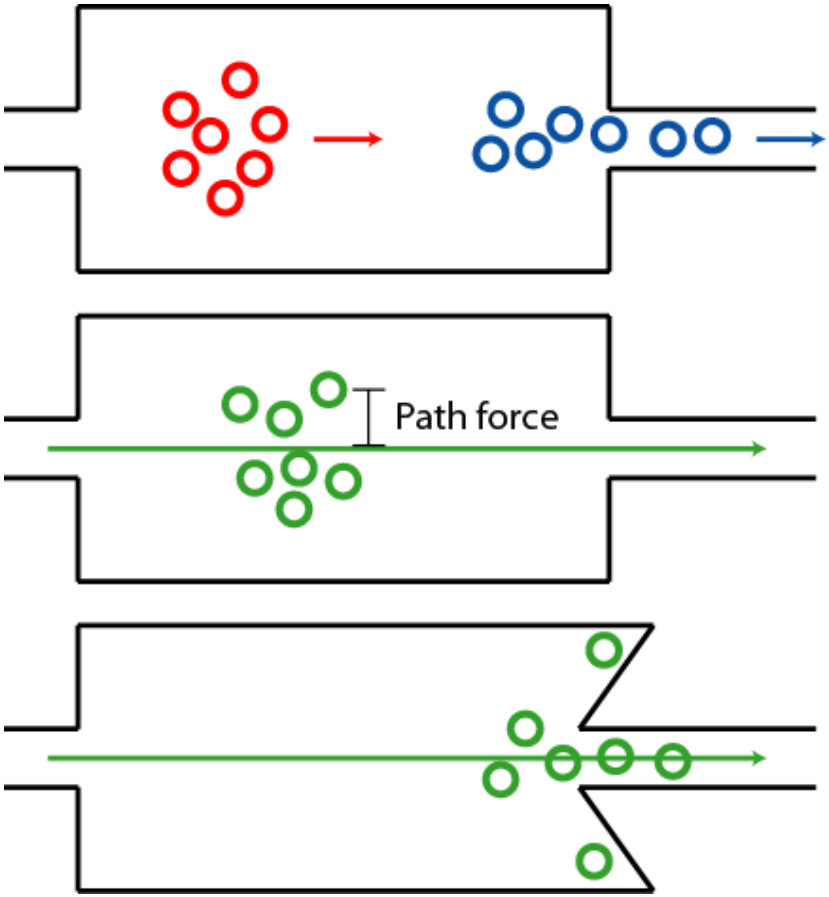
What can I do with my convex area graph?

- Very fast navigational ray-casts
- Distance to edge queries
- Repulsion fields on edges
- Smart path following

Ray Casts



Smart Path Following

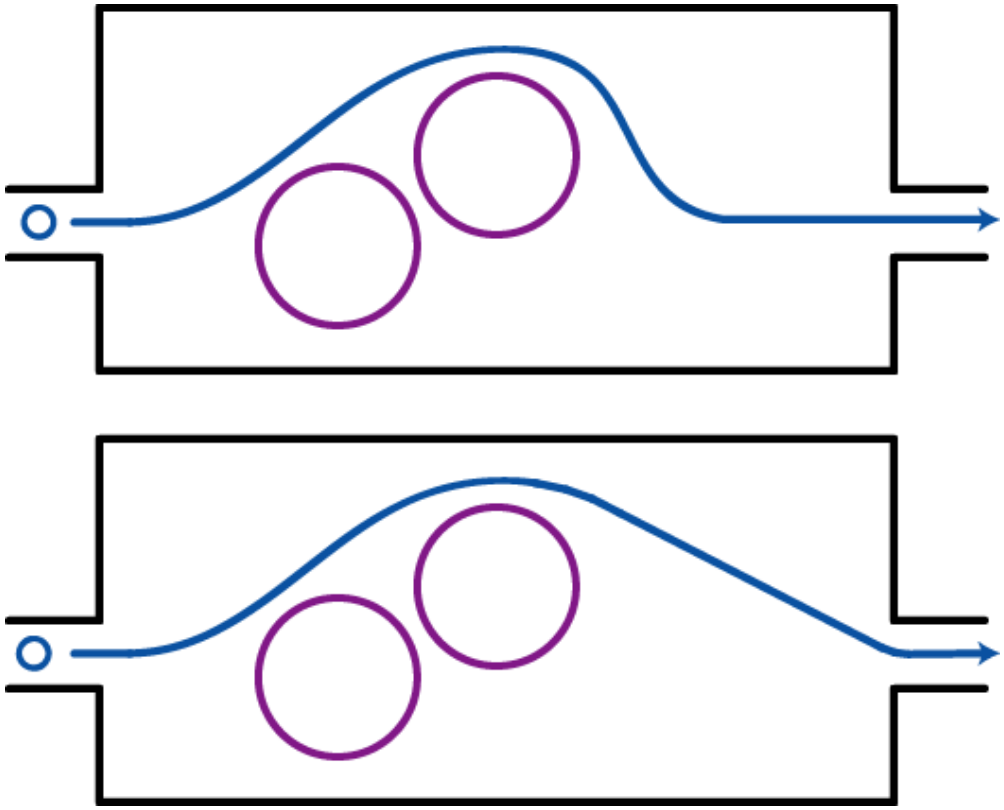


Desired flocking behavior

Common approach attracts each flock member to the minimum distance path. Does not adapt to available freespace.

Repulsion from edges can easily trap individuals in local minima.

For The Pathing Connoisseur

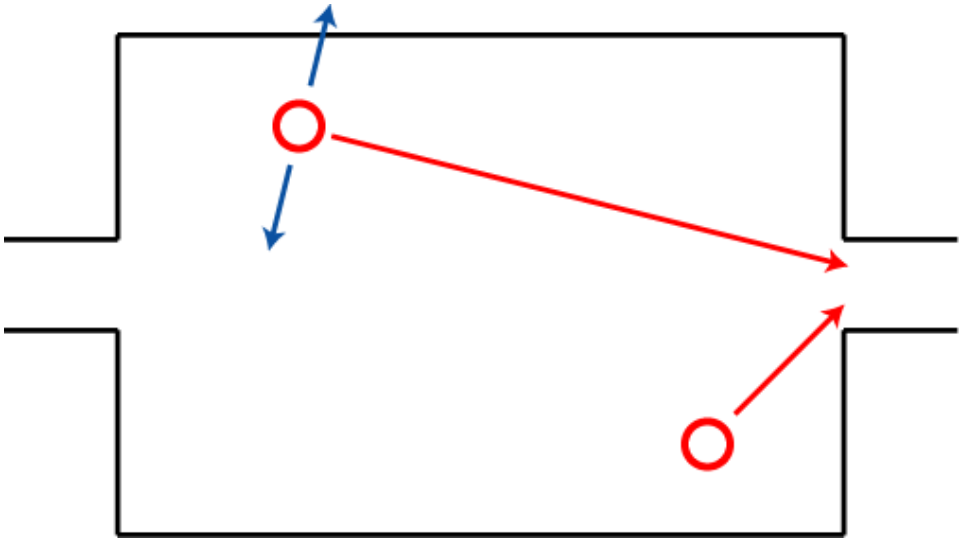


Unneeded return to the original path annoys the pathing connoisseur

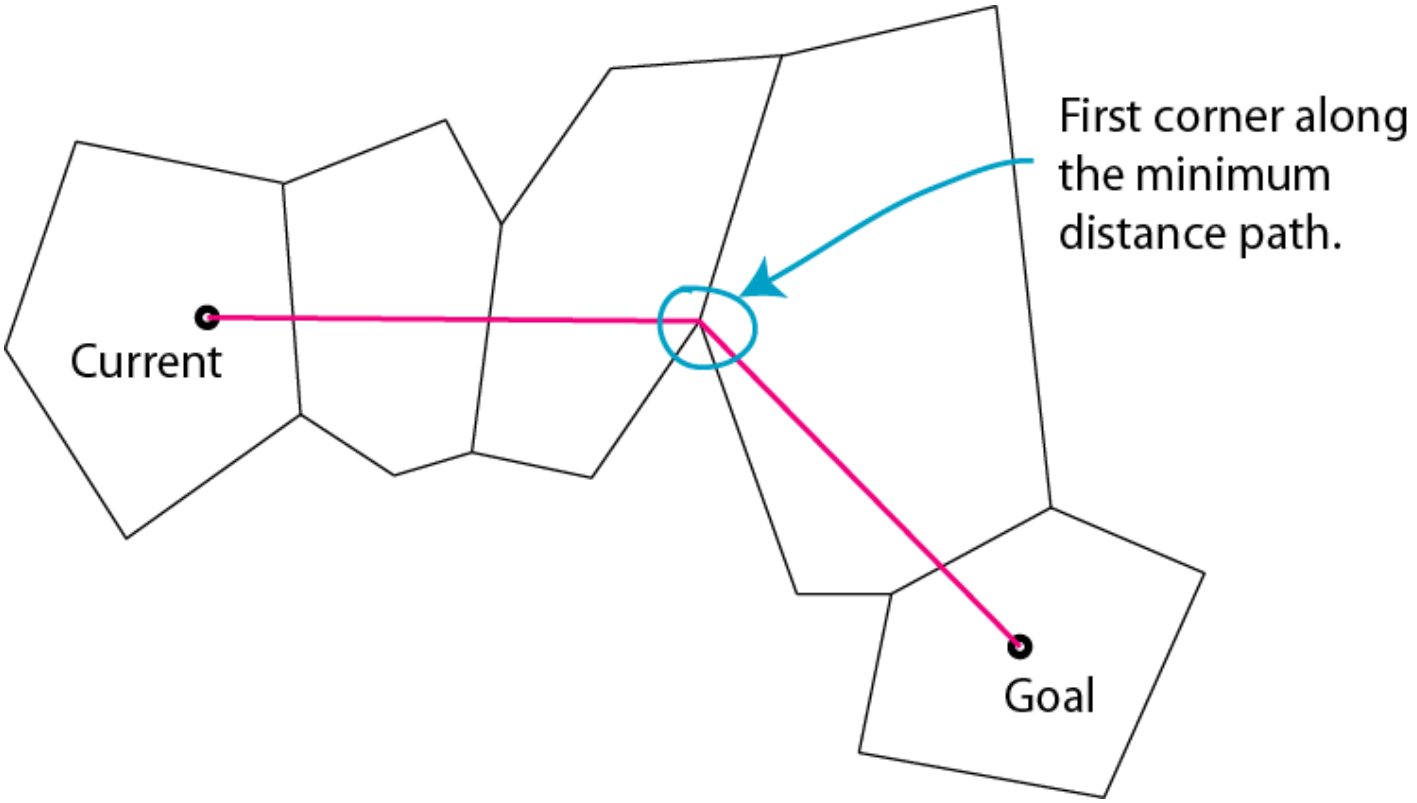
Ahhh, much better

Next Corner

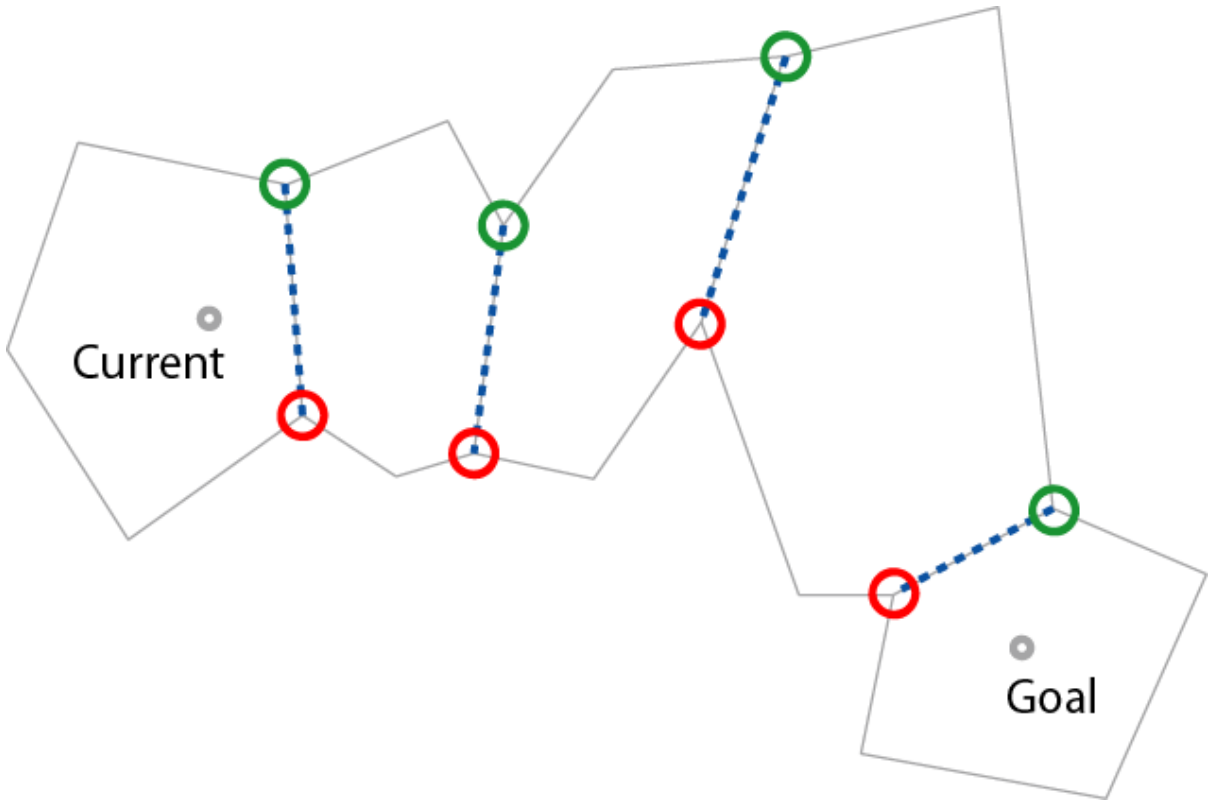
Each member of the flock steers left or right to head towards the “next corner”.



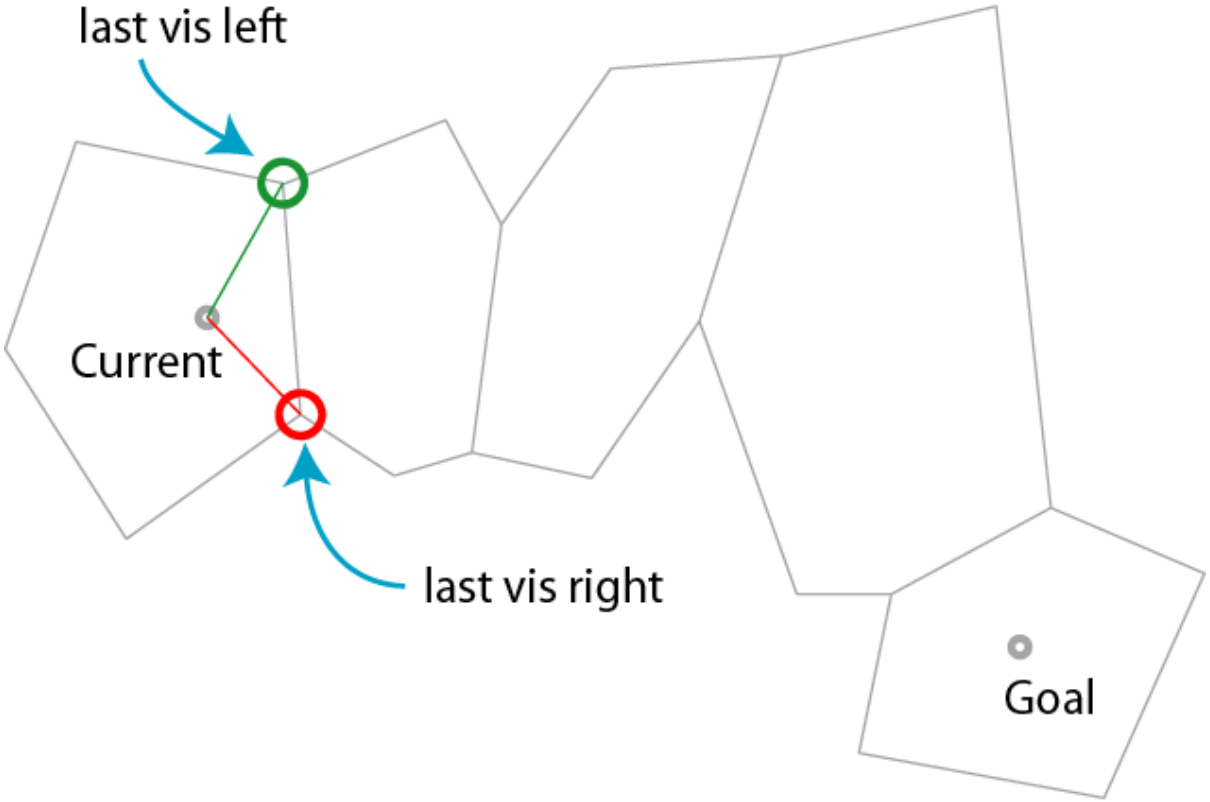
Finding the Next Corner



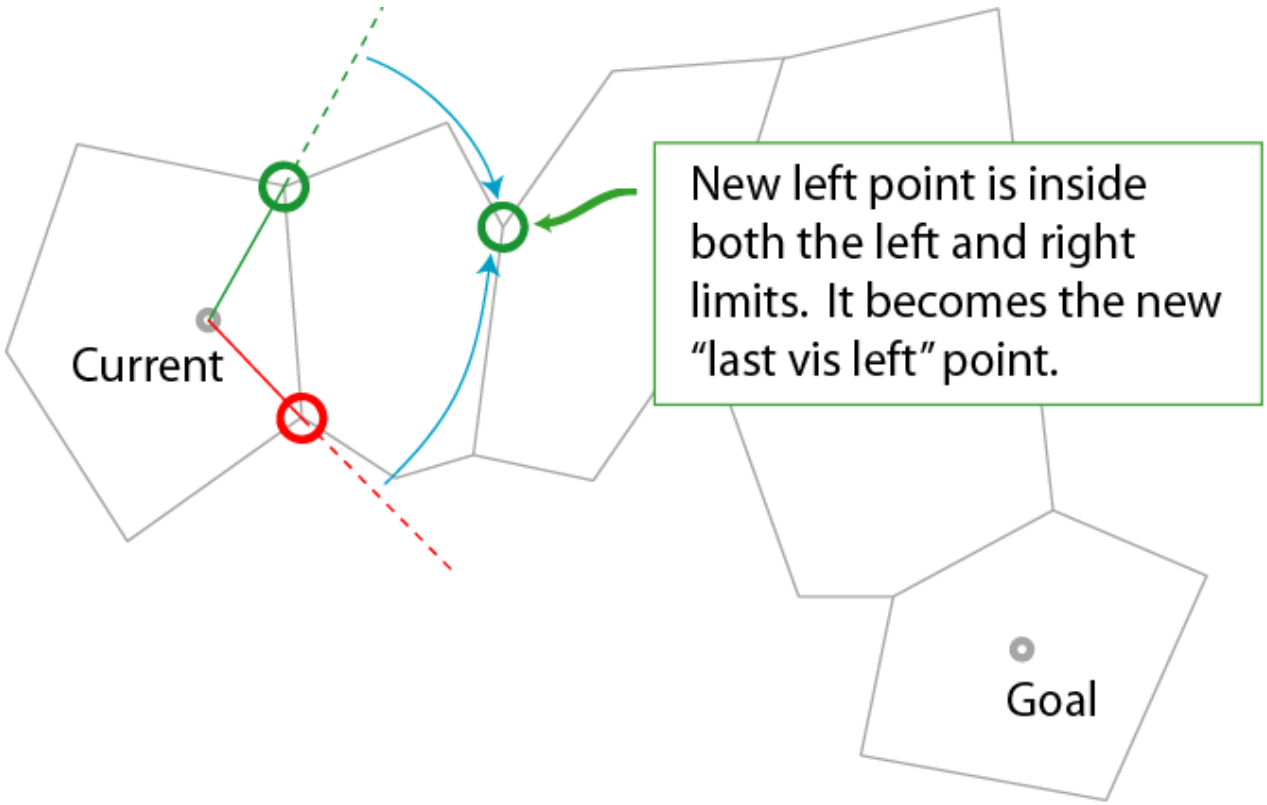
“Portals” Between Areas



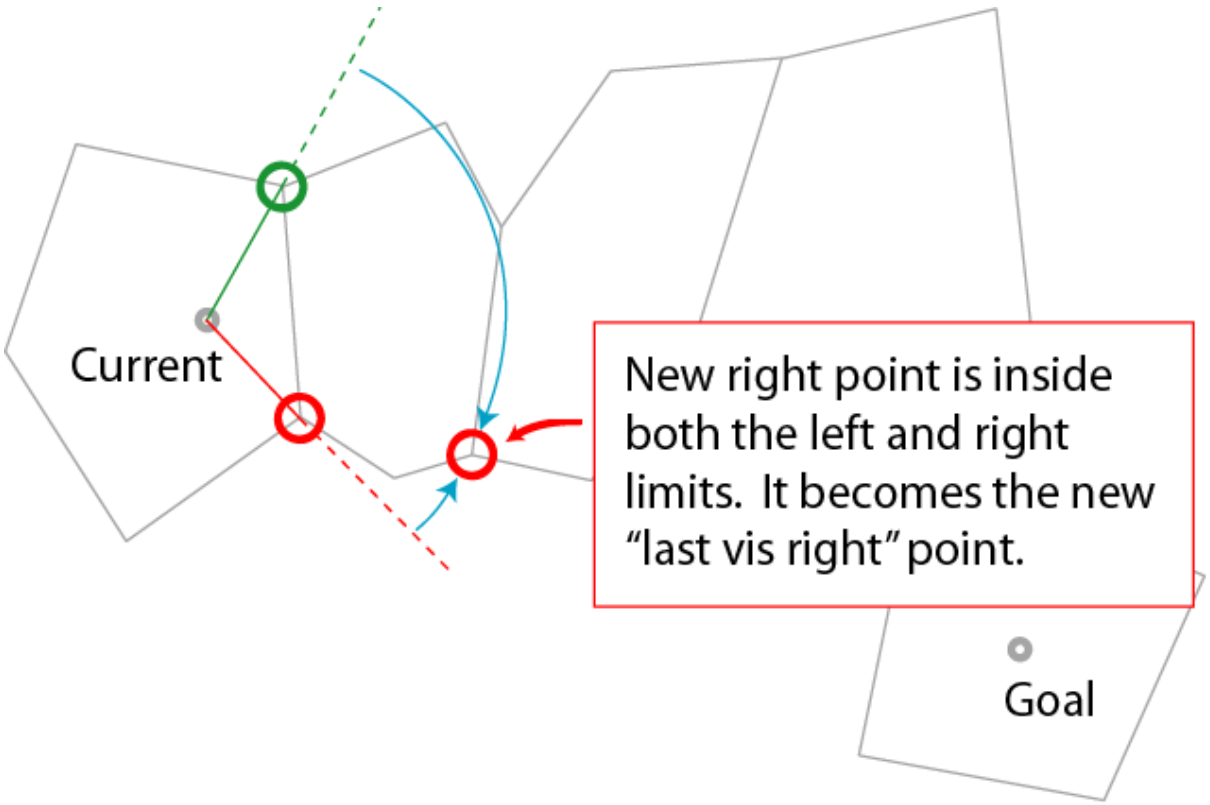
Initializing the Loop



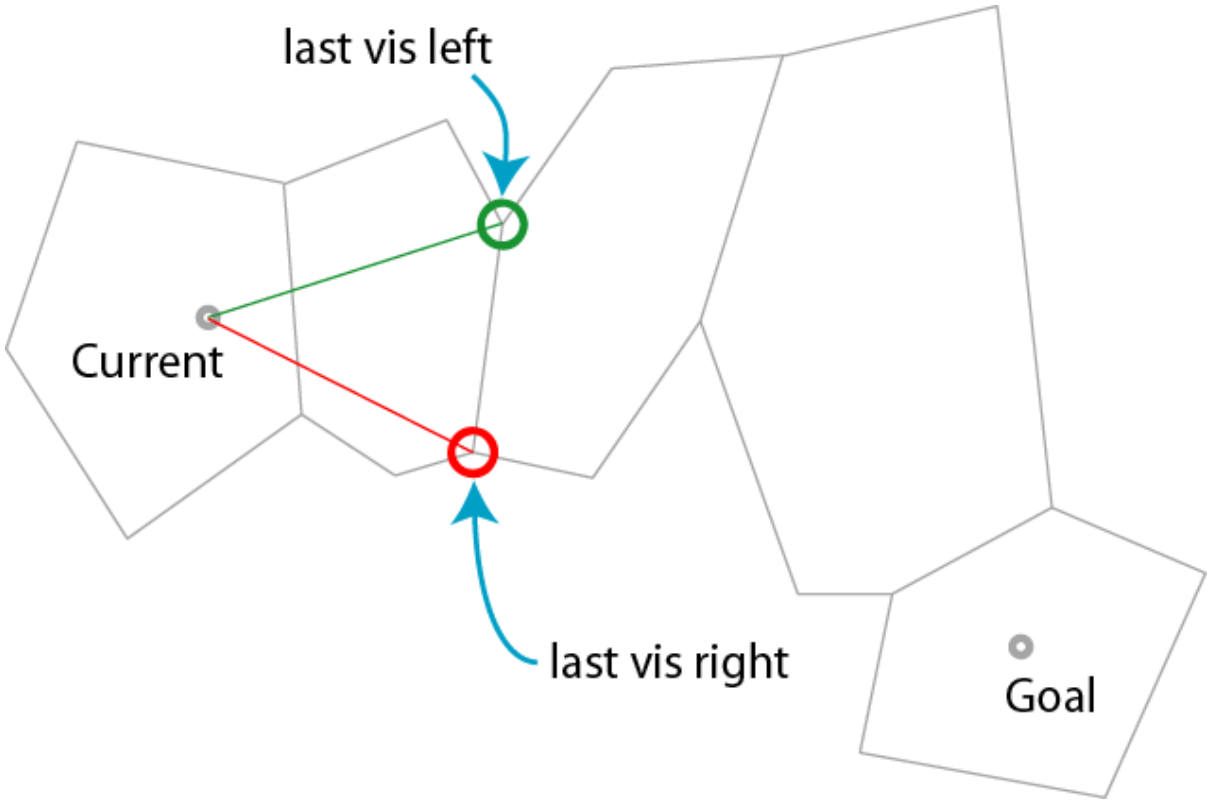
First Iteration, Left Side



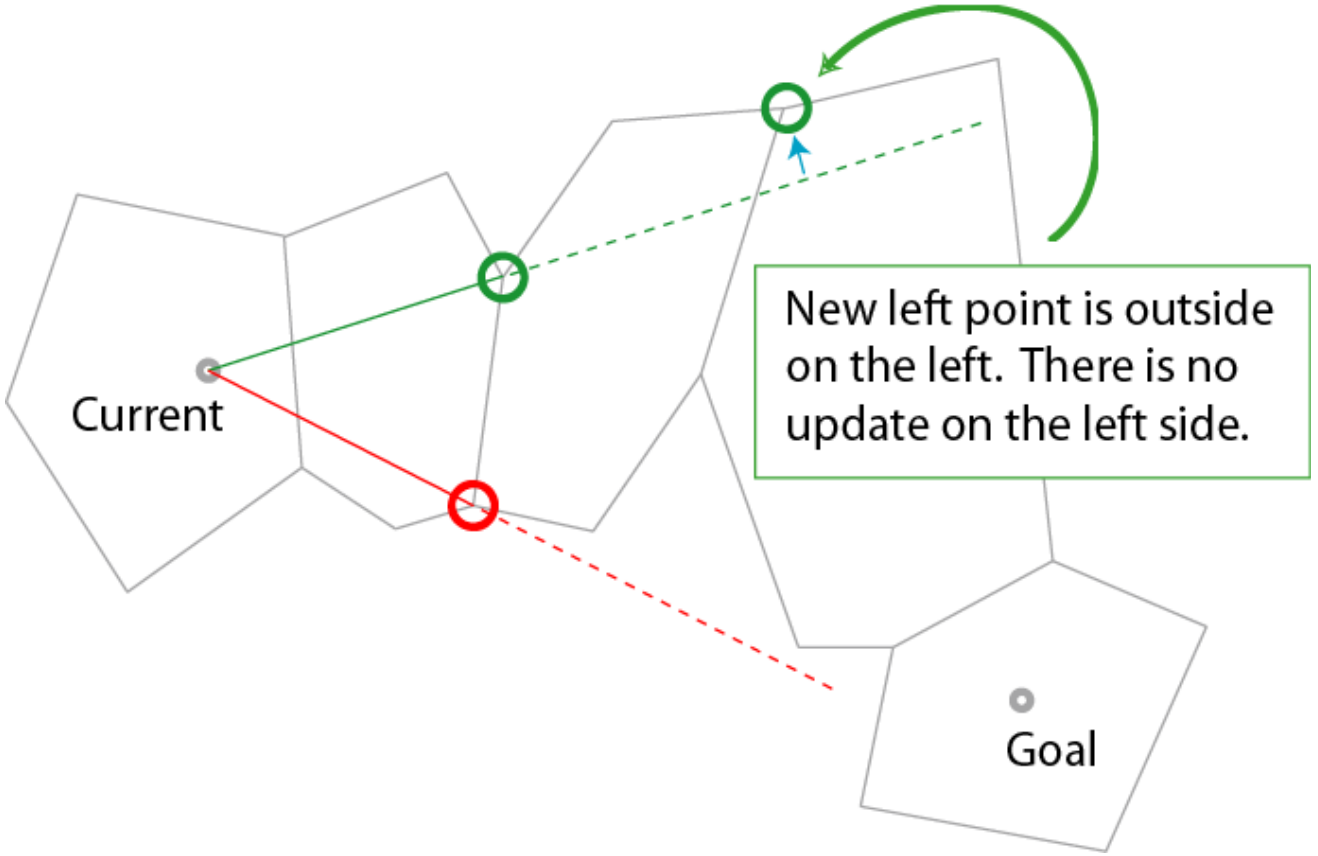
First Iteration, Right Side



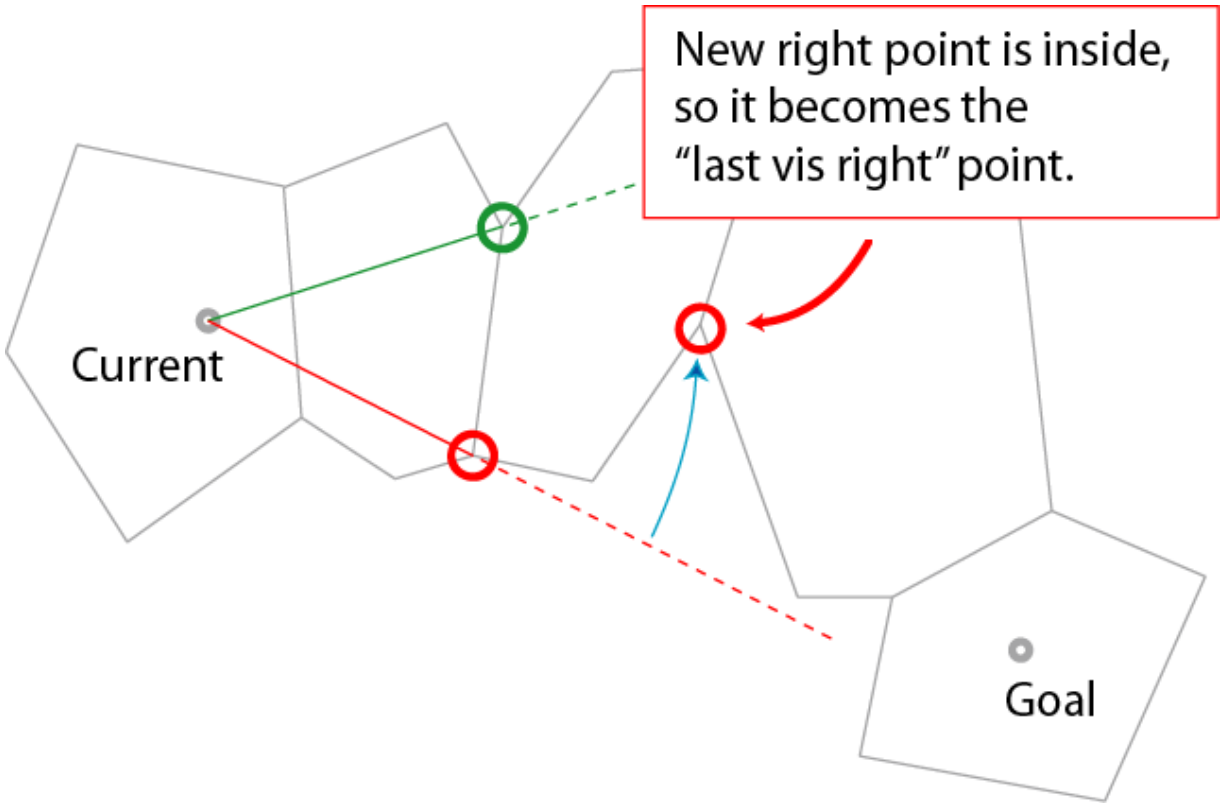
First Iteration Complete



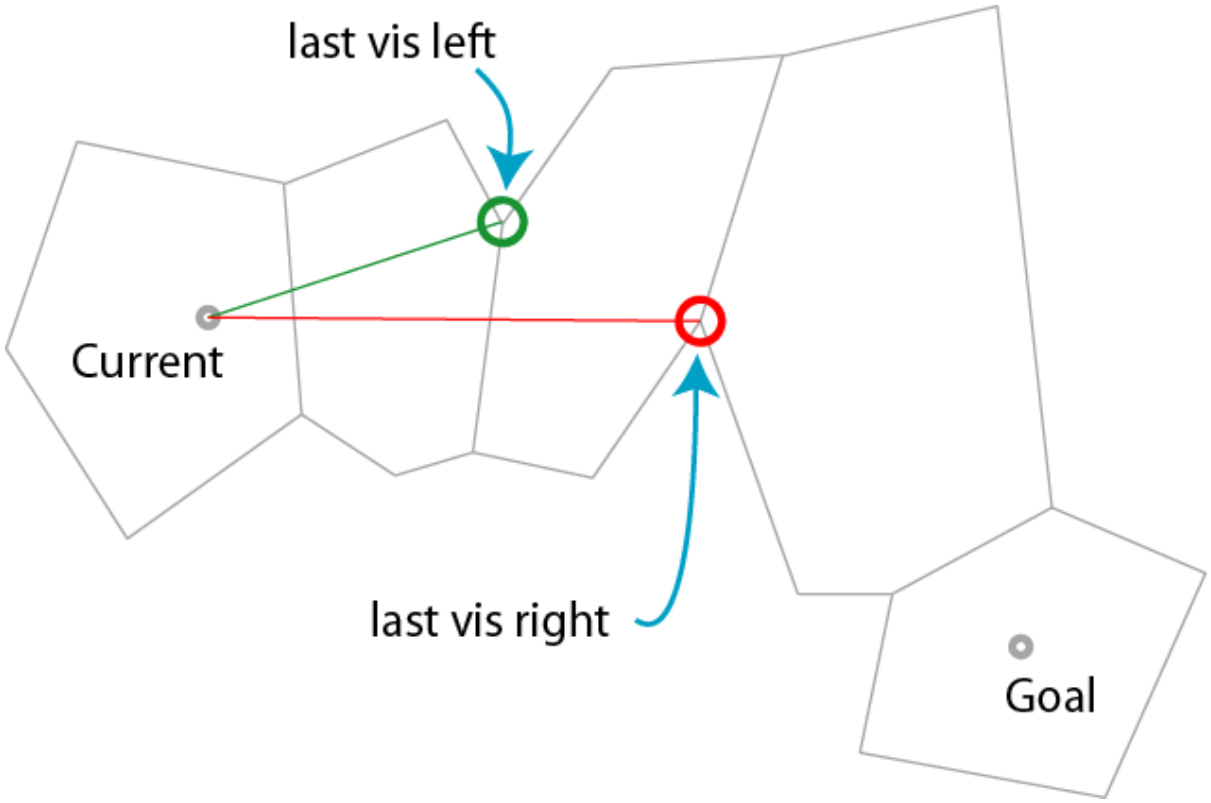
Second Iteration, Left Side



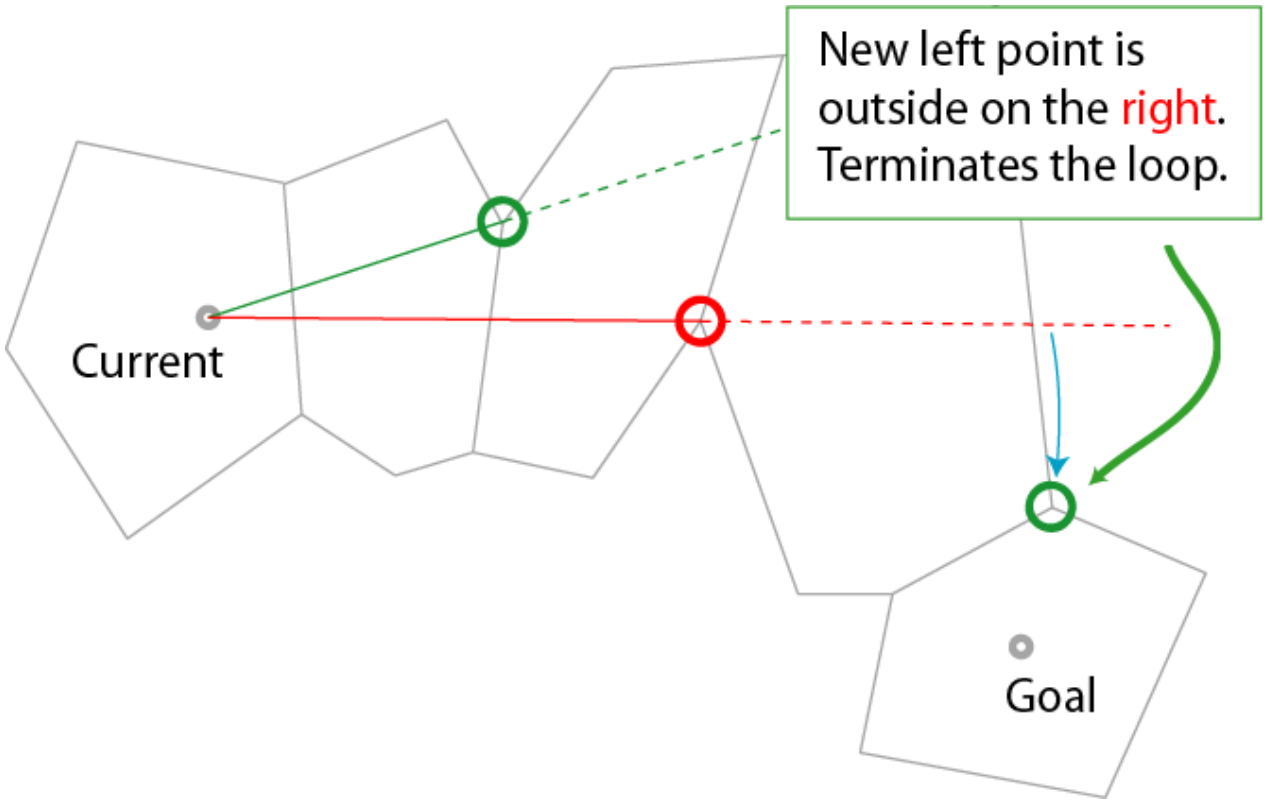
Second Iteration, Right Side



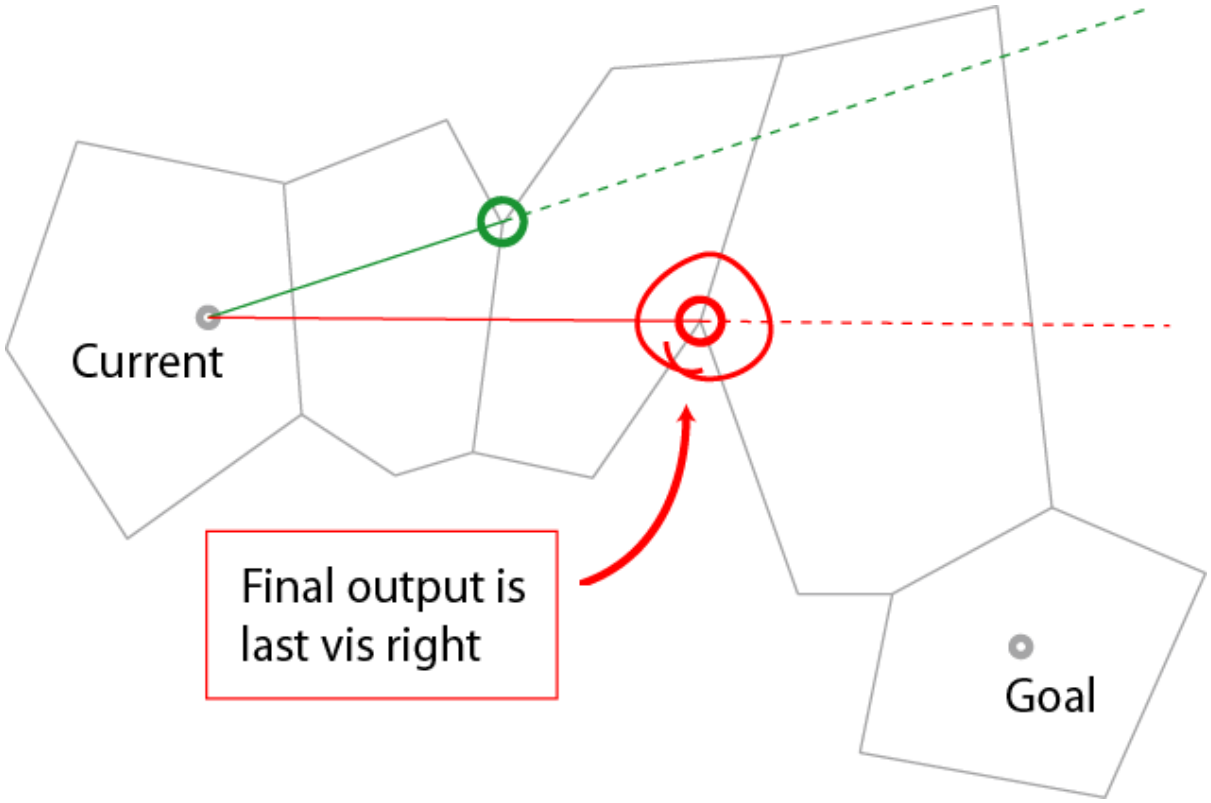
Second Iteration Complete



Third Iteration, Left Side



Final Output



"Next Corner" Summary

- Easy to compute from convex graph
- < 2 cross-products per iteration
- Can limit iterations
- Never explicitly create min-dist path
- Fluid path following even with large disturbances

Conclusions

- Automated build of convex area graph
  - efficiently represents the usable free space
  - operates on polygon soup mesh
  - settable enemy size and shape
- Dynamic obstacles update graph
  - no speed overhead once updated
- Fluid AI navigation using the graph
  - Many useful queries performed rapidly

Special Thanks

- Meilin Wong
- Hong Park and David Modiano
- Crystal Dynamics

- dmiles@navpower.com
- www.navpower.com