Getting off the NavMesh

Navigating in Fully 3D Environments

Daniel Brewer

Digital Extremes

www.warframe.com

Overview

- Define the problem space
- 3D Local Avoidance
- 3D Path Planning
- Results, tips and hints

What is Warframe

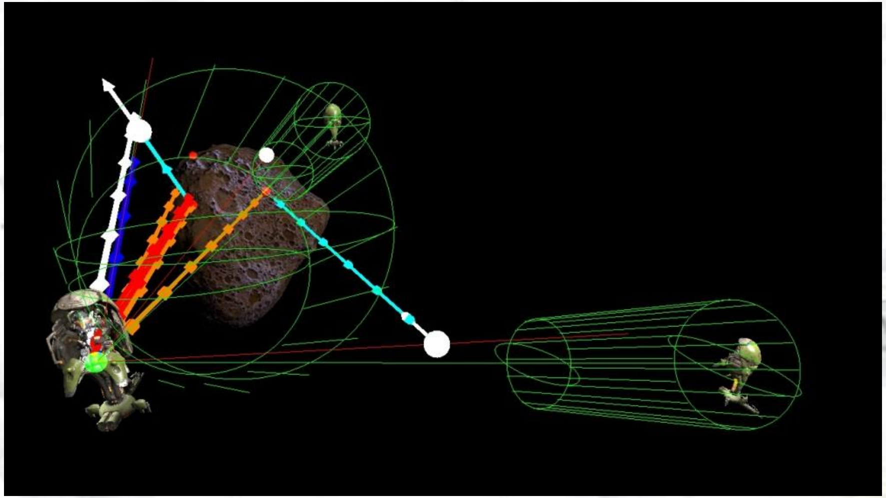
- Free to play
- Online co-op
- Space Ninjas
- Procedural Levels

Problem Space

- Full 3D flight
- Open Space
- Scattered chunks of complex debris
- Procedurally Generated Levels

Avoidance

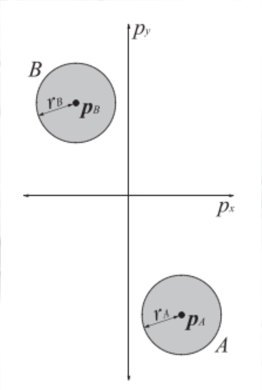
- Velocity Obstacles work well in 2D
- Can be extended to 3D



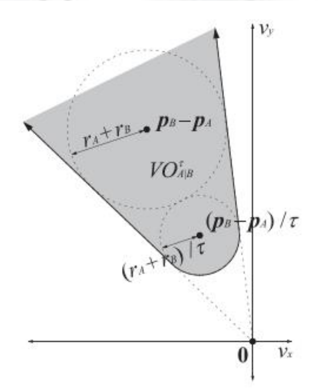
Kythera.ai
Star Citizen's Flight Simulation and Combat AI
http://aigamedev.com/broadcasts/session-star-citizen/

Paper:
Navigating Multiple Simple-Airplanes in 3D Workspace
by Jamie Snape and Dinesh Manocha

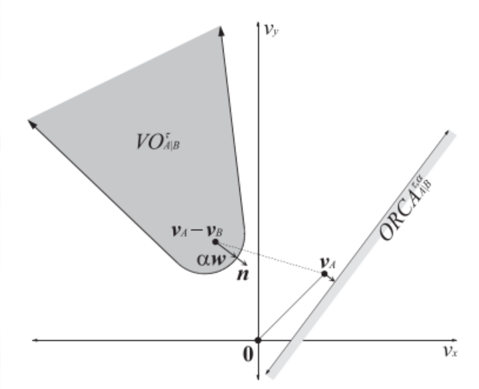
3D ORCA



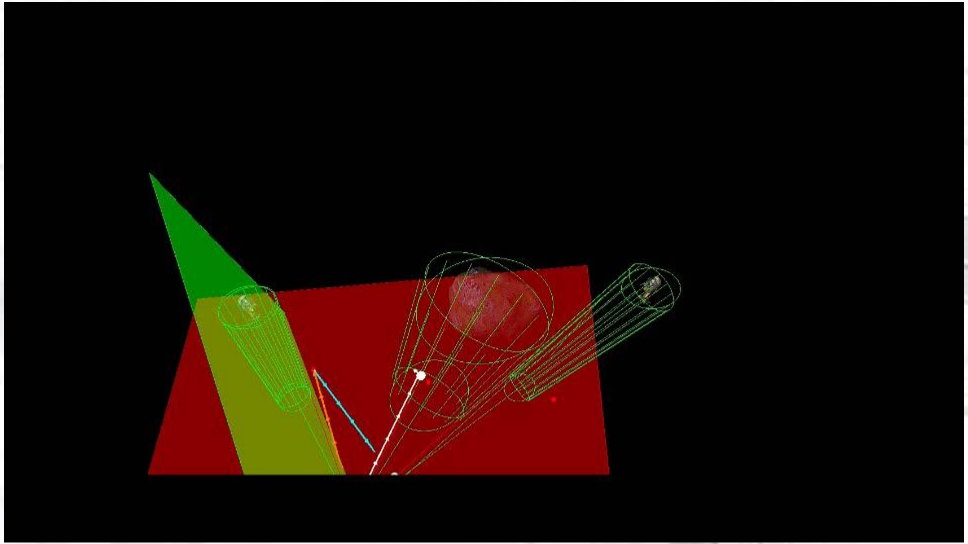
p_y
B
r_B
$\boldsymbol{p}_B$
p_x
r_A
$\boldsymbol{p}_A$
A



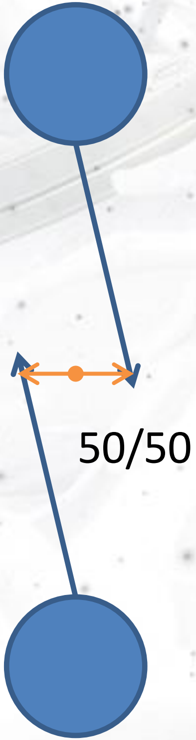
v_y
$r_A + r_B$
$\boldsymbol{p}_B - \boldsymbol{p}_A$
$VO^{\tau}_{A|B}$
$(\boldsymbol{p}_B - \boldsymbol{p}_A)\,/\,\tau$
$(r_A + r_B)\,/\,\tau$
$\mathbf{0}$
v_x

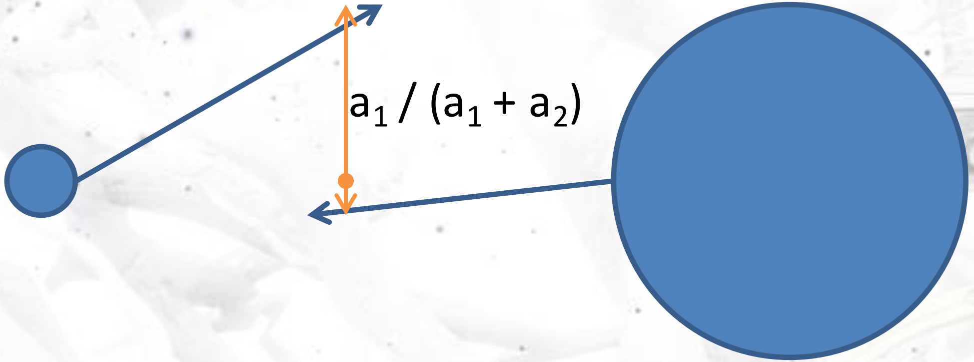


v_y
$VO^{\tau}_{A|B}$
$\boldsymbol{v}_A - \boldsymbol{v}_B$
$\alpha\boldsymbol{w}$
$\boldsymbol{n}$
$\boldsymbol{v}_A$
$ORCA^{\tau,\alpha}_{A|B}$
$\mathbf{0}$
v_x

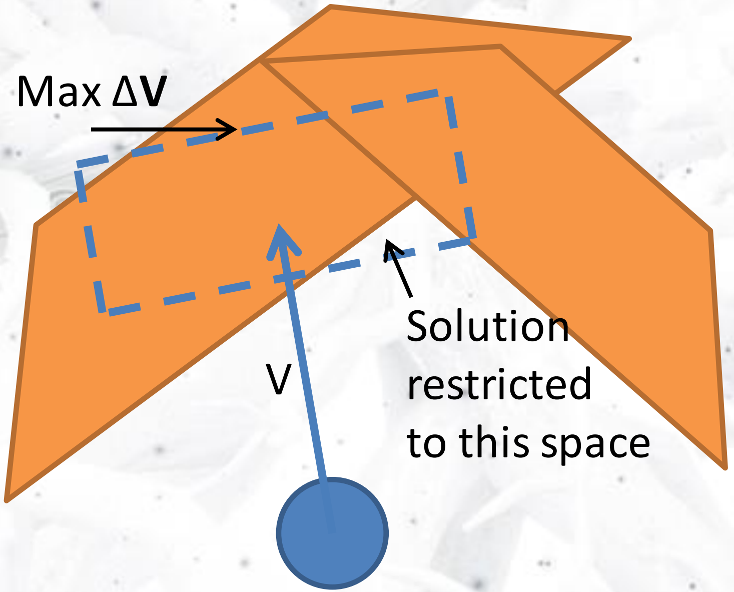


3D ORCA



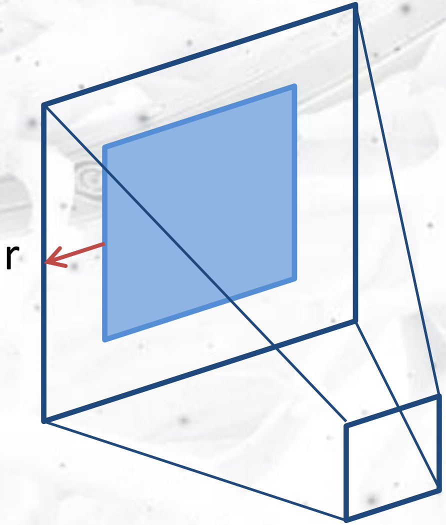


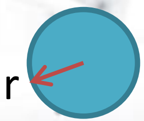
3D ORCA



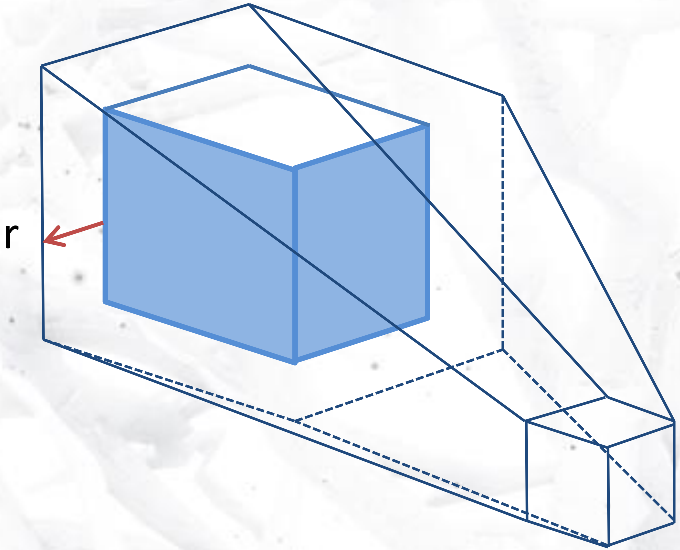
- ORCA assume instant velocity changes
- Real vehicles have momentum and inertia
- Additional constraints for maximum attainable ΔV
- Increase time horizon to find obstacles earlier

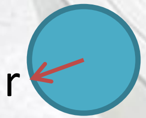
More than spheres





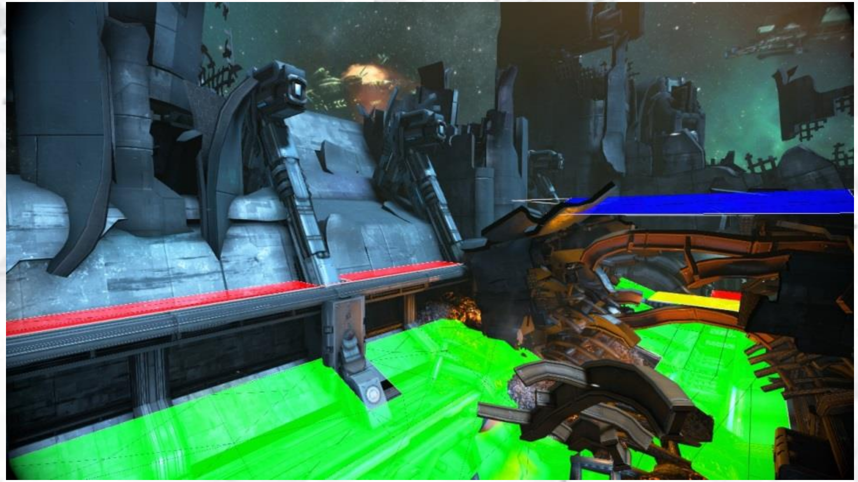
Rectangle
Velocity Obstacle





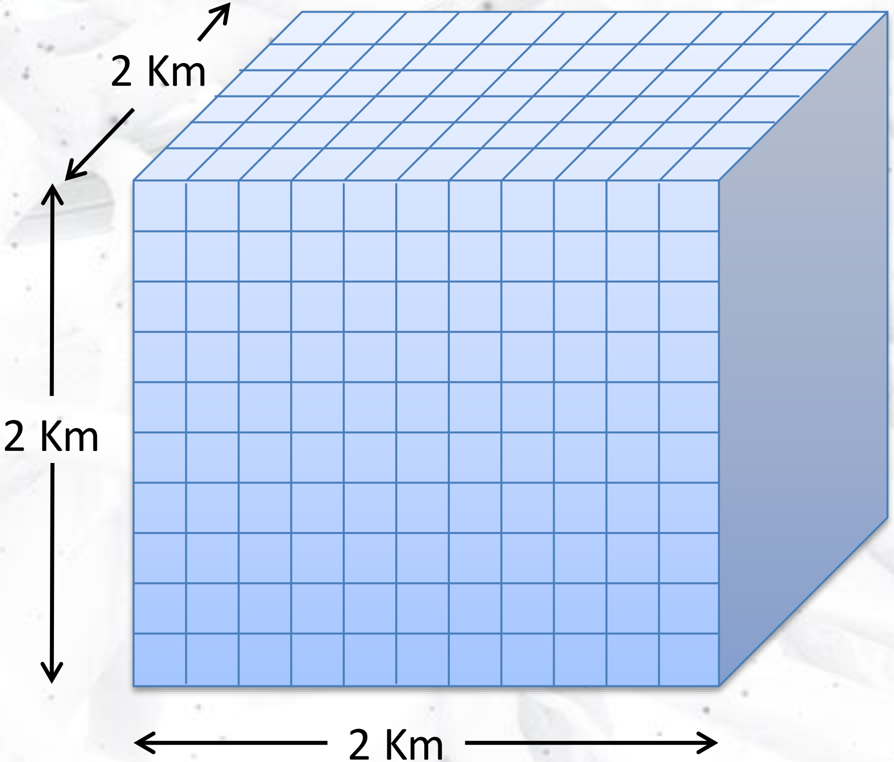
Box
Velocity Obstacle

3D Path Planning



- Multiple Layers of NavMesh
- Connections between layers
- Suitable for constrained environments
- How many layers for 2Km cube?

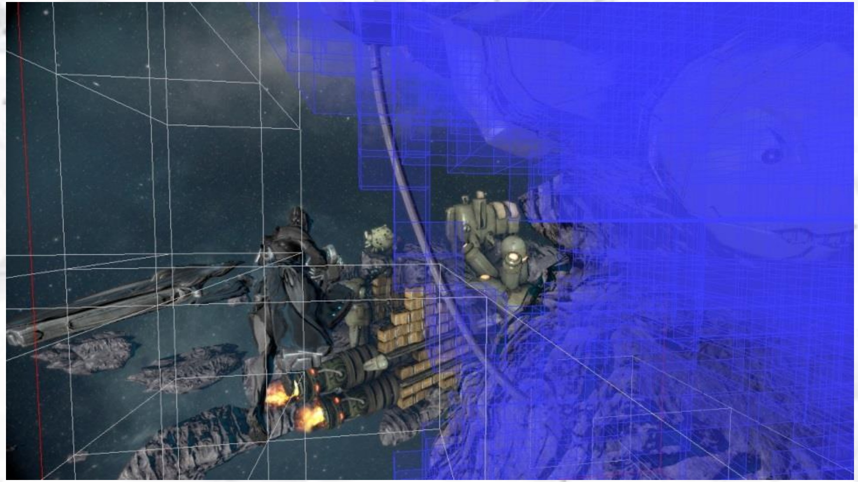
3D Path Planning



- Regular 3D Grids
- Too memory intensive
- 2Km cube at 1m resolution uses 8 Gb!

- Need a compact, adaptive, volumetric representation

Sparse Voxel Octree



- Common Rendering Structure
- Compact
- Neighbour connectivity
- Morton Code Order helps memory access patterns and streaming

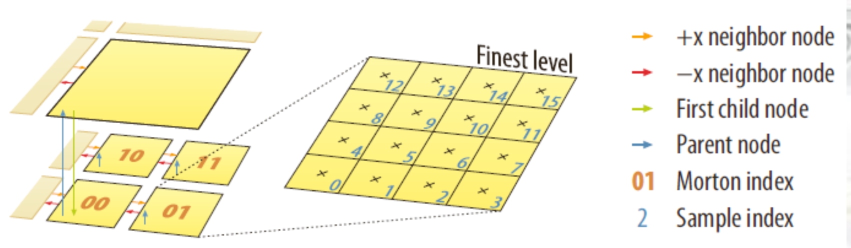
Sparse Voxel Octree

- Construct from bottom up
- Low res rasterization to determine Morton codes for required leafs nodes
- Block allocate all memory
- Fill out parent-child ptrs on way up
- Fill out neighbour ptrs on way down
- Rasterize the final detail leaf nodes
- Detail leafs are 4x4x4 voxel grids packed into 64 bits

Paper:

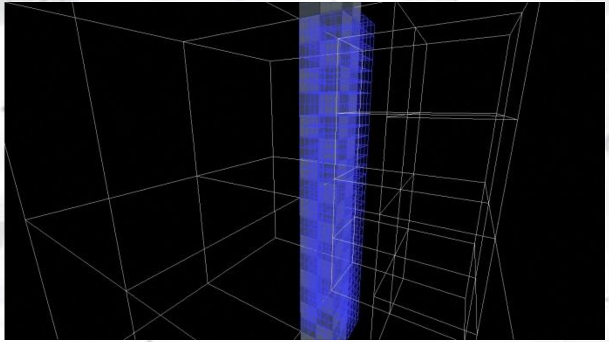
Fast Parallel Surface and Solid Voxelization on GPUs

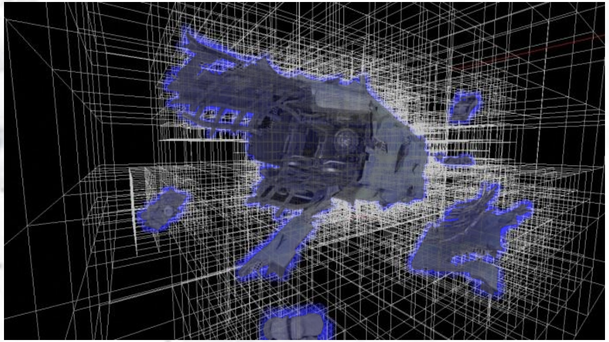
by Michael Schwarz and Hans-Peter Seidel



2D analog of SVO architecture

Sample SVO build stats



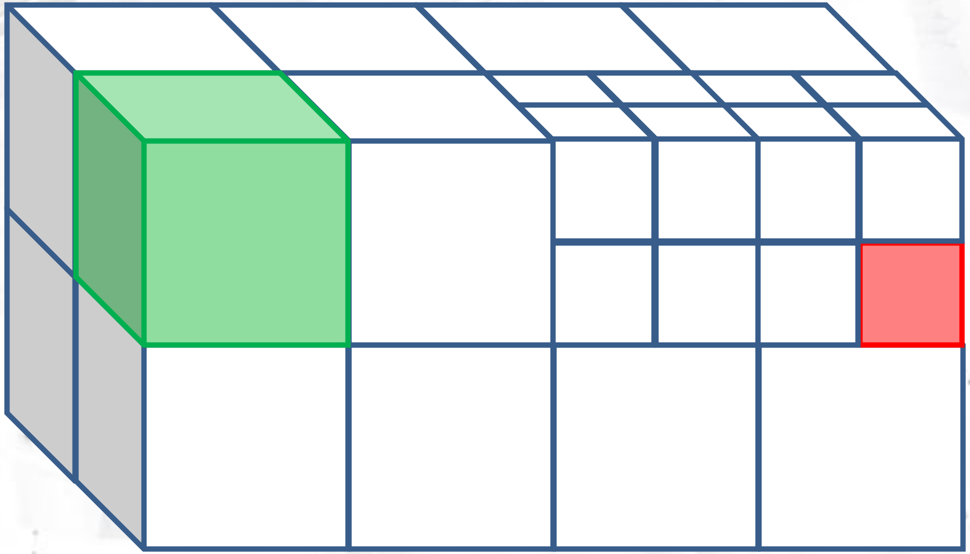


Simple Test
32 x 32 x 32
8 collision polygons
32,768 regular grid nodes
3 octree layers
56 oct nodes, 32 lead nodes
2,104 pathfind nodes
2,944 bytes

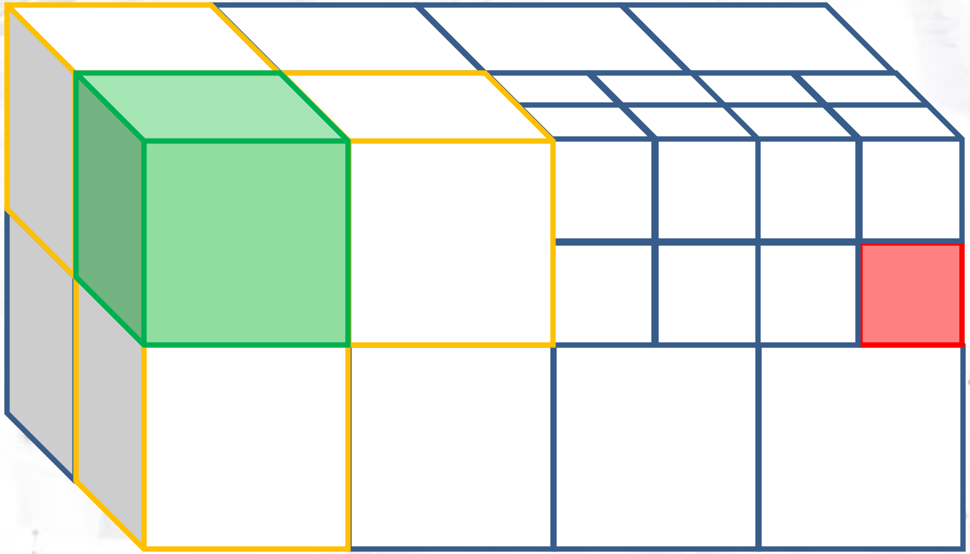
Complex Test
128 x 160 x 96
40,435 collision polygons
1,966,080 regular grid nodes
6 octree layers
5,680 oct nodes, 4,224 leaf nodes
276,016 pathfind nodes
306,432 bytes

Typical Level
1024 x 1024 x 1024
481,417 collision polygons
1,073,741,824 regular grid nodes!
8 octree layers
43,648 oct nodes 30,800 leaf nodes
2,014,848 pathfind nodes
2,398,960 bytes

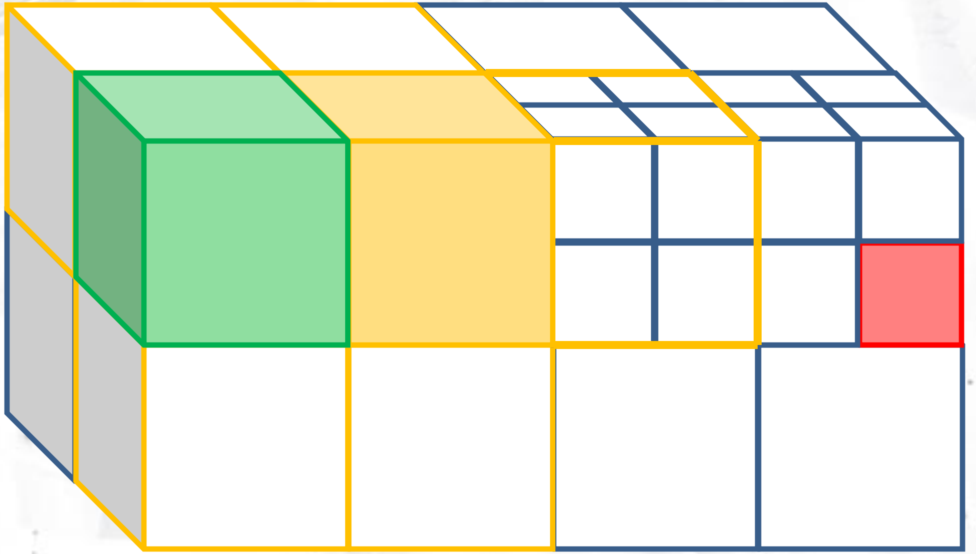
Pathfinding on SVOs



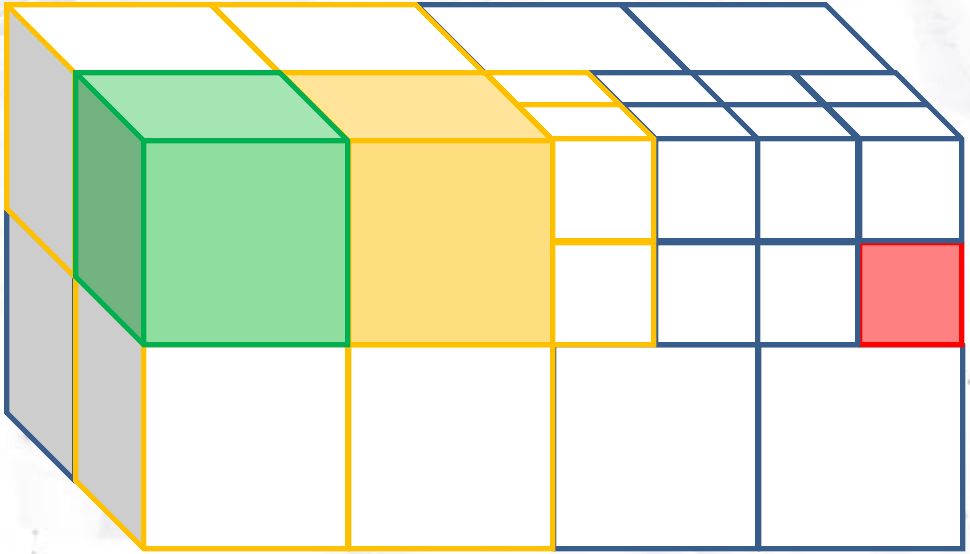
Pathfinding on SVOs



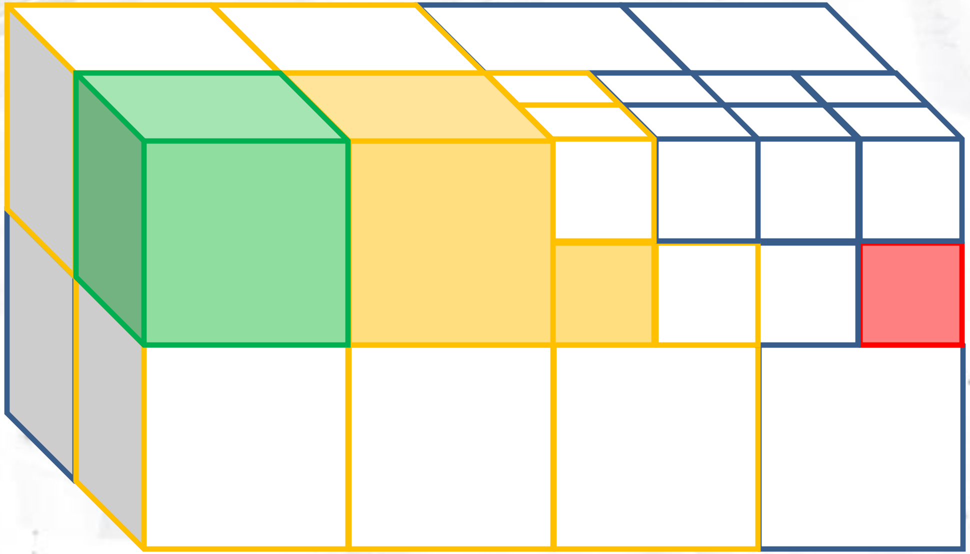
Pathfinding on SVOs



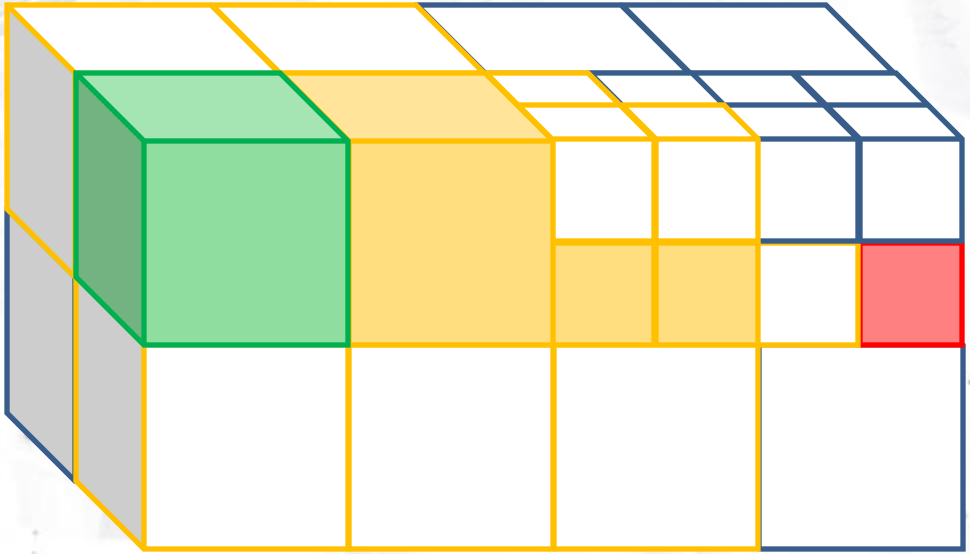
Pathfinding on SVOs



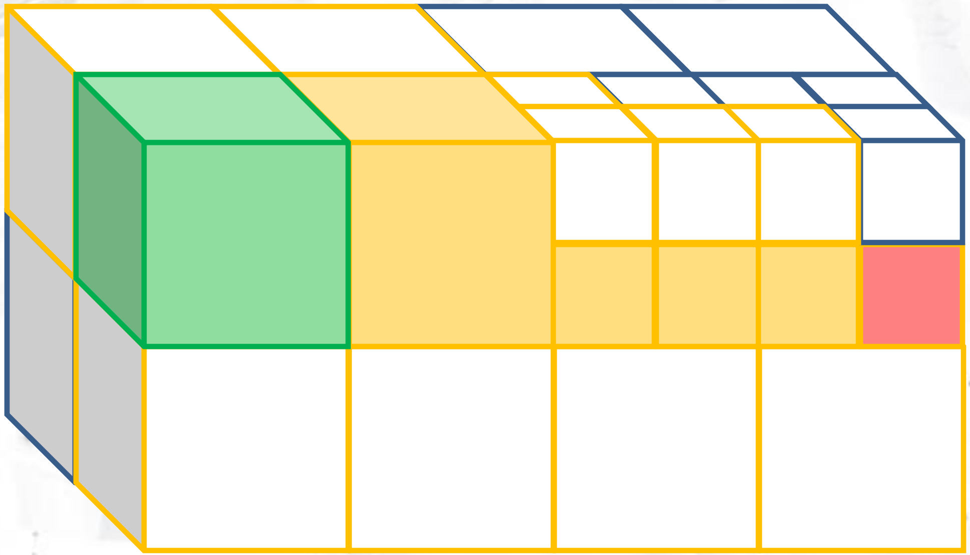
Pathfinding on SVOs



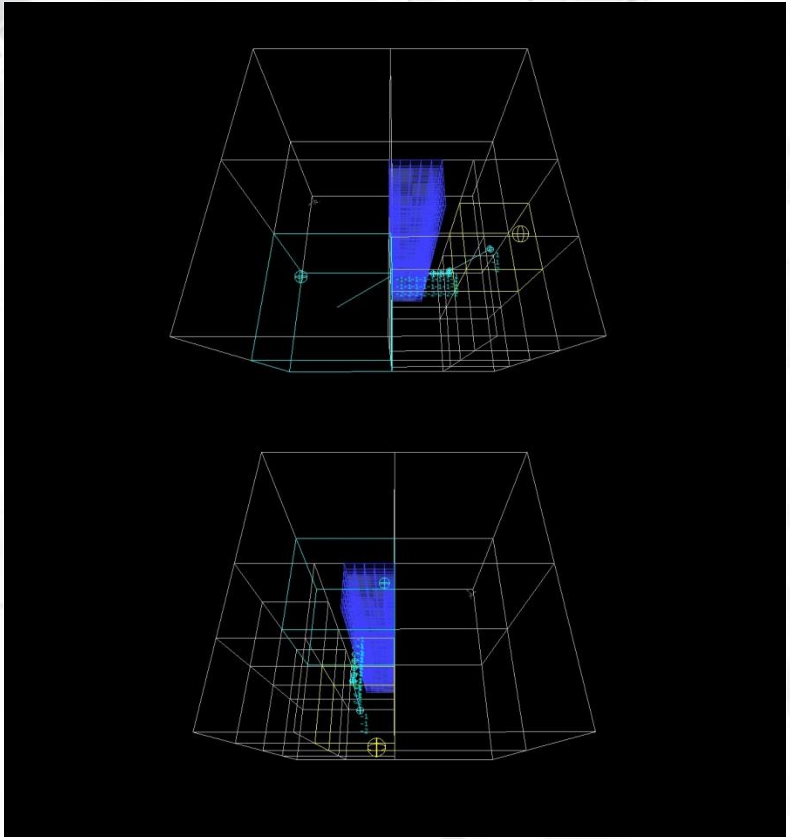
Pathfinding on SVOs

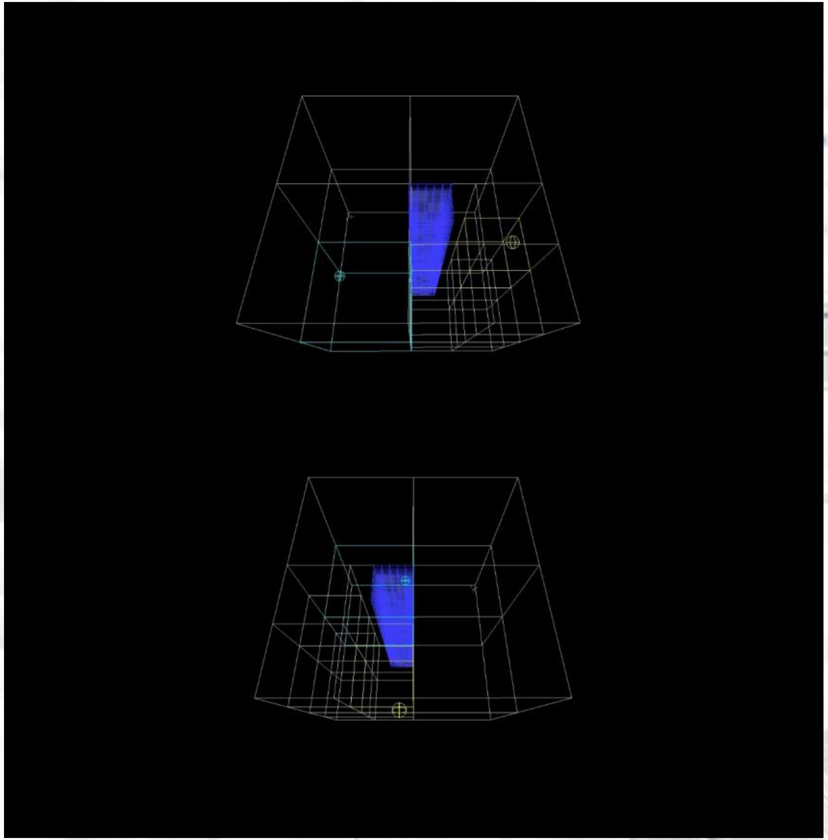


Pathfinding on SVOs



A* on SVOs

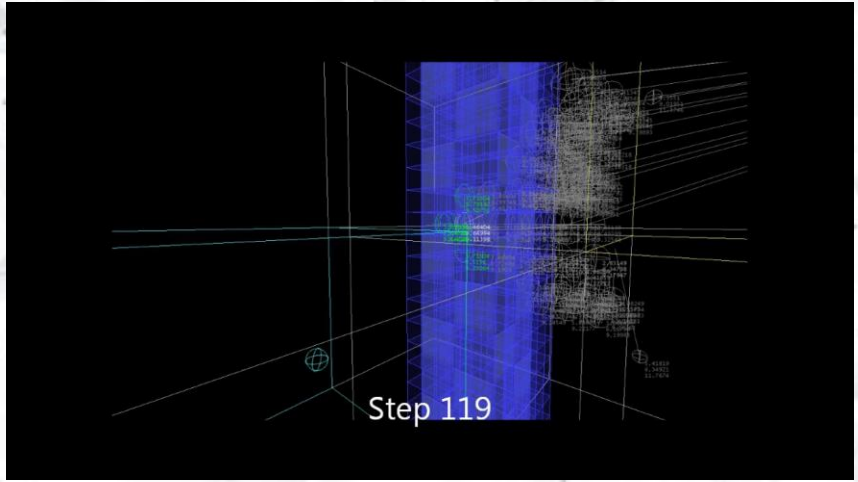


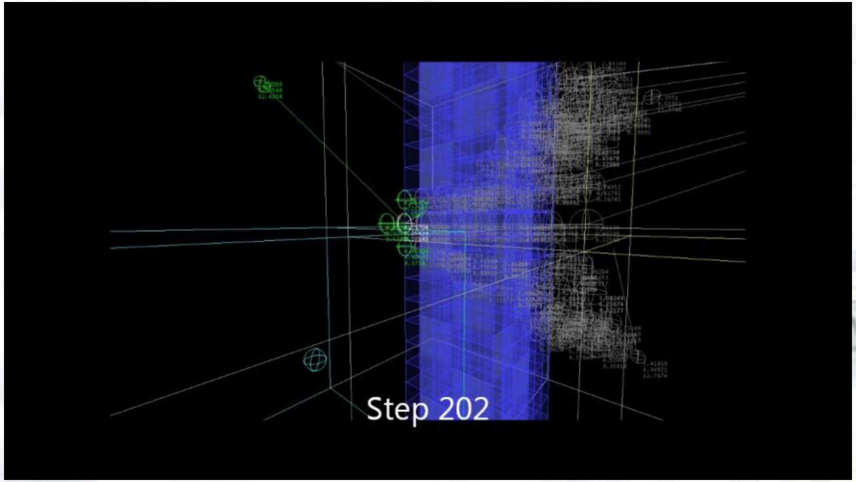


Top Tip!

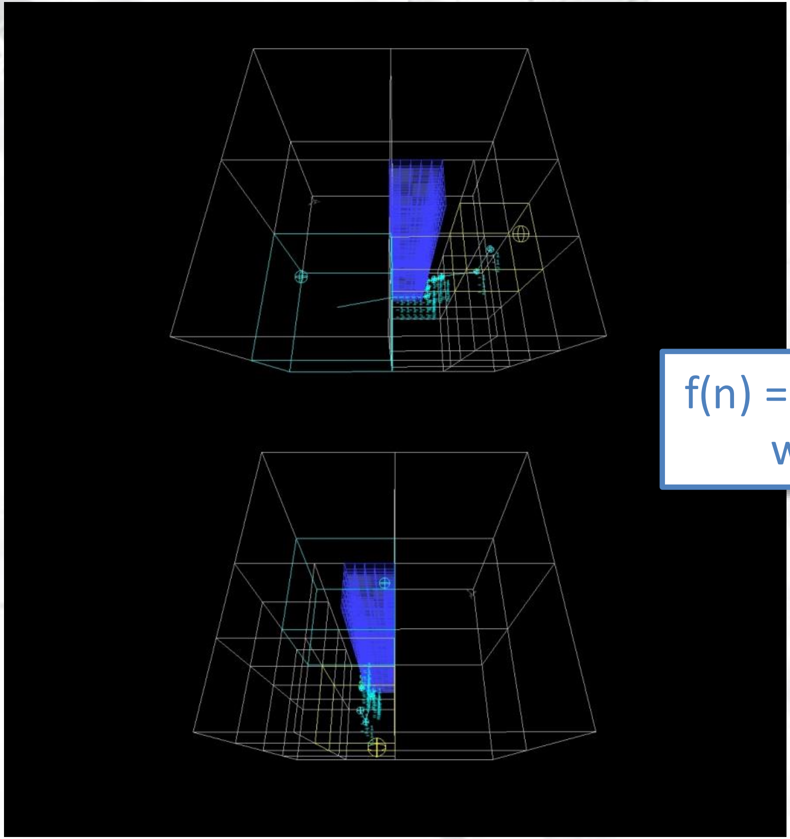
Always set up debug visualization of all the stages of your algorithm!

Leap Ahead and Catch Up

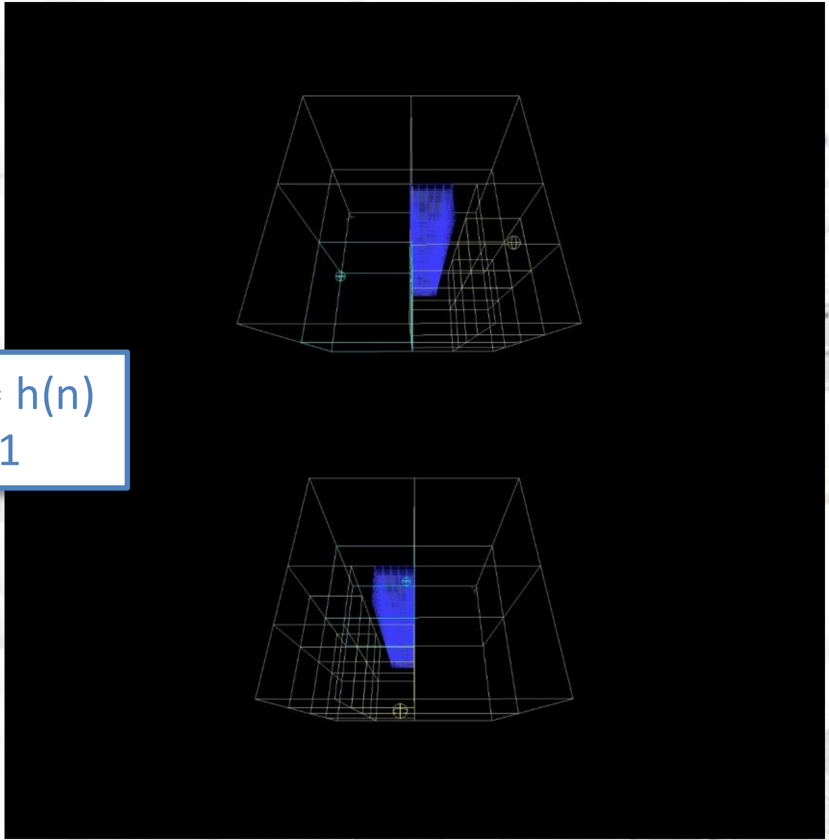




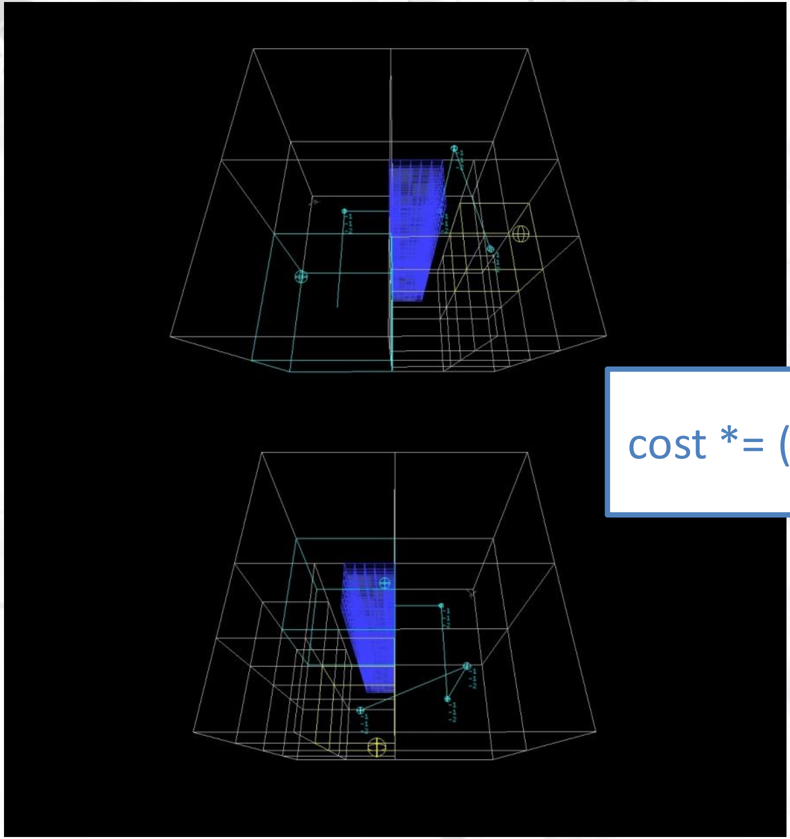
Greedy A*

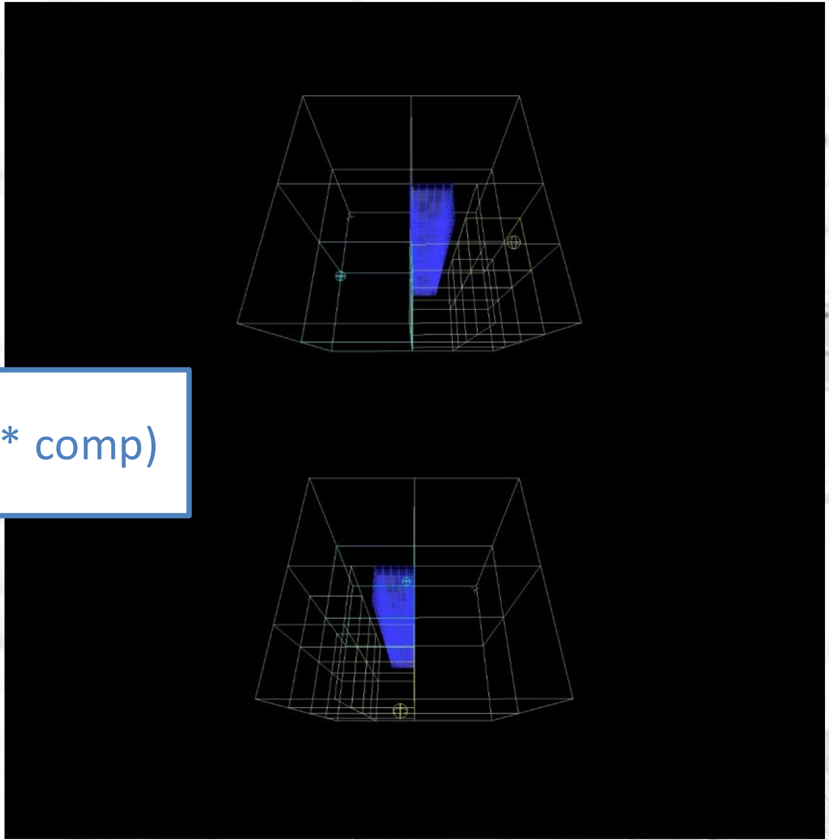


$$f(n) = g(n) + w * h(n)$$
where $w > 1$



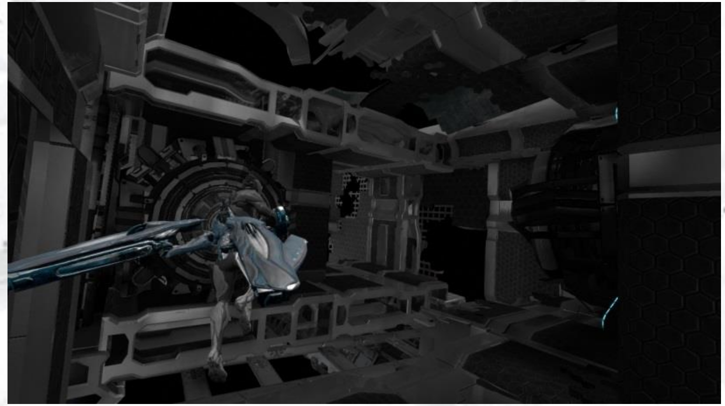
Node Size Compensation

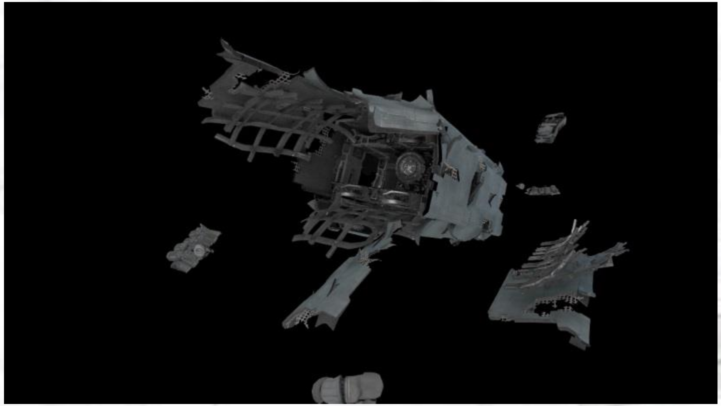


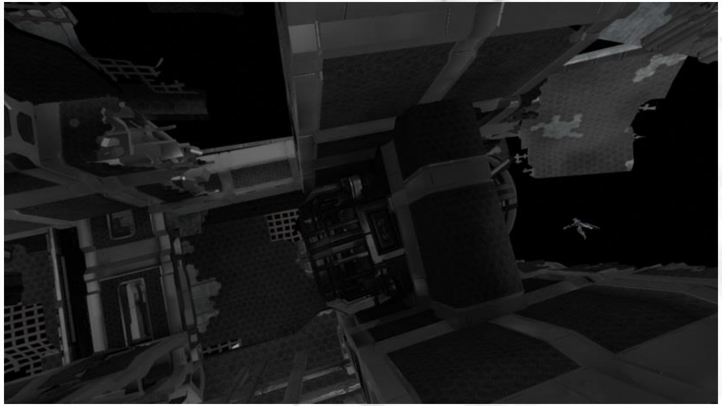


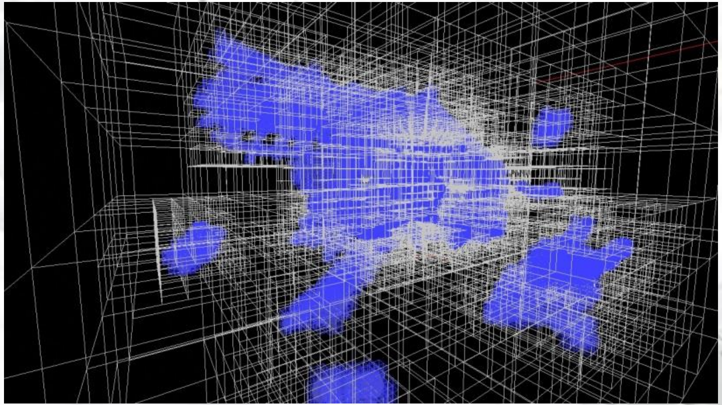
cost *= (1.0f - size * comp)

Complex Case

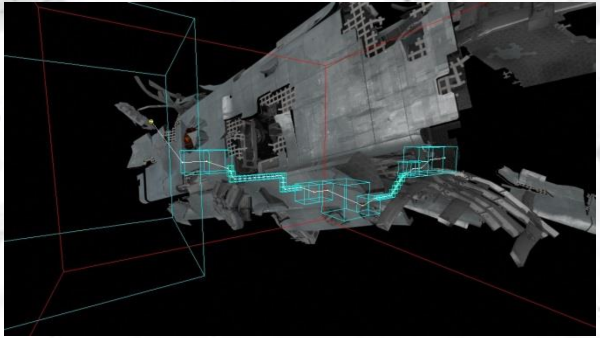




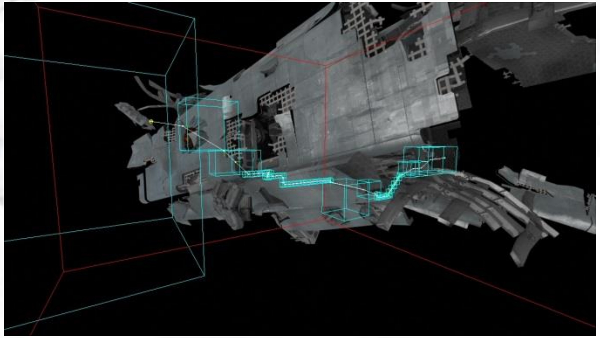




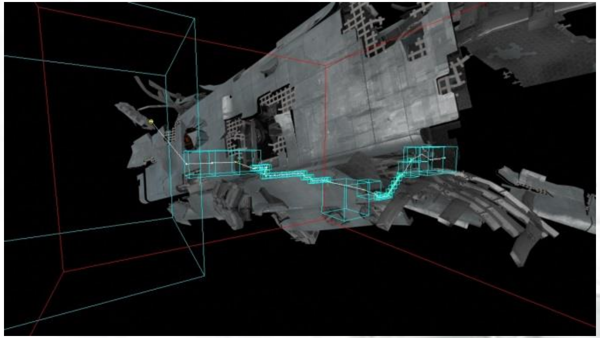
Complex Case



A*
Node Centers
Straight Line Distance
32,916 iterations
50 path steps

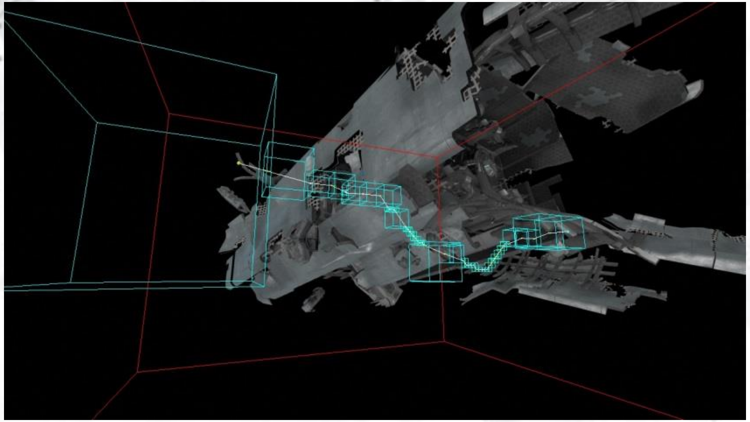


A*
Face Centers
Manhattan Distance
10,692 iterations
57 path steps

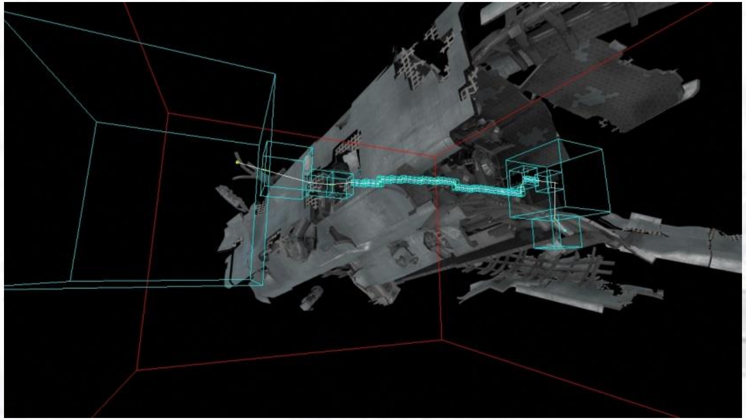


Greedy A*
Face Centers
Manhattan Distance
3,378 iterations
58 path steps

Complex Case

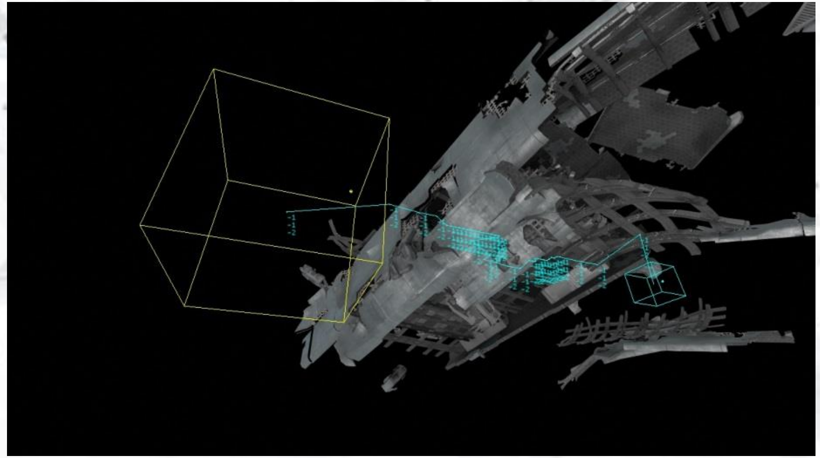


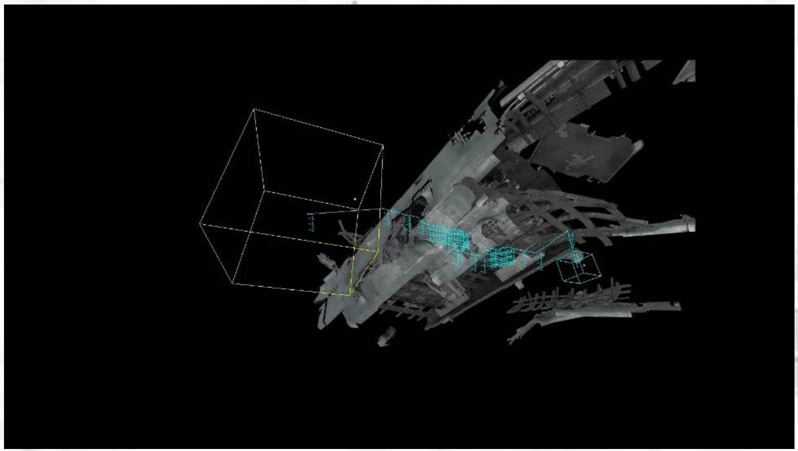
Greedy A*
Face Centers
Manhattan Distance
Size Compensation
2,425 iterations
49 path steps



Greedy A*
Node Centers
Straight Line Distance
Size Compensation
1625 iterations
59 path steps

Complex Case

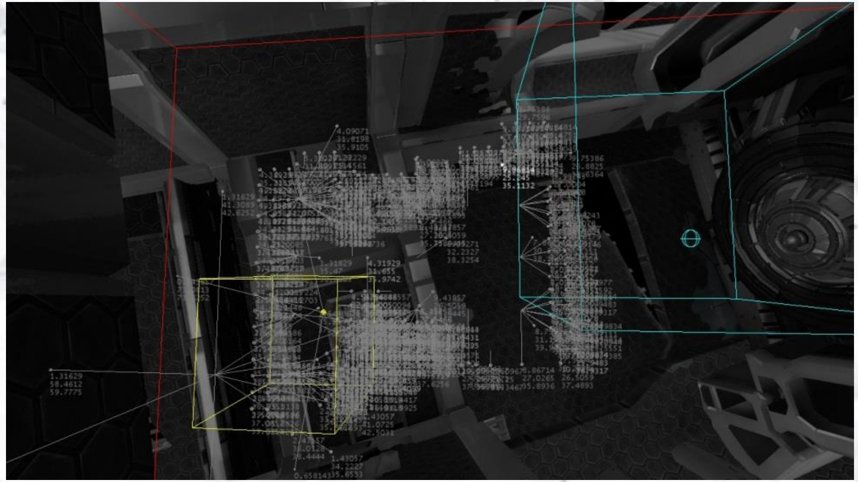


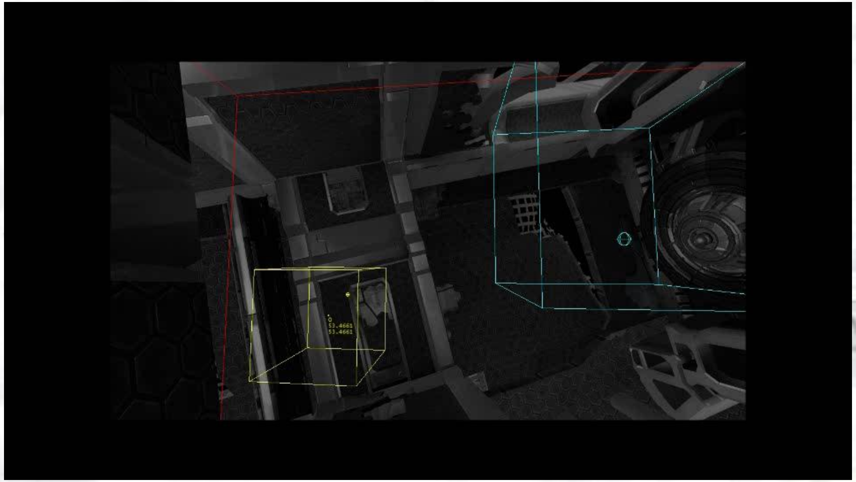


Greedy A*
Node Centers
Straight Line Distance
Size Compensation
Unit Node Cost

213 iterations!
42 path steps

Doughnut of Doom!





Future Optimizations

- JPA* style jump points
- O^3 searching for jump points may not be quicker than optimized A* search
- Hierarchical Search
- Possible memory bloat to store passable flags between each face of SVO nodes

Thank You!

daniel.brewer@digitalextremes.com

NINJAS PLAY FREE AT

WWW.WARFRAME.COM

AI GAME PROGRAMMERS GUILD

DIGITAL EXTREMES