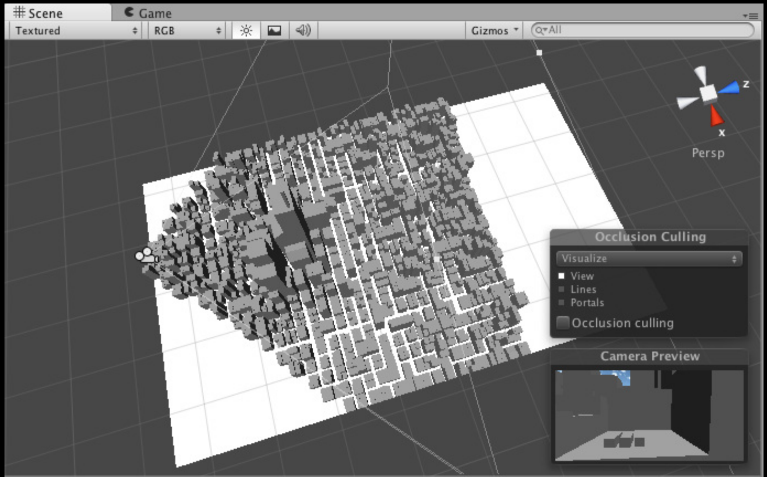
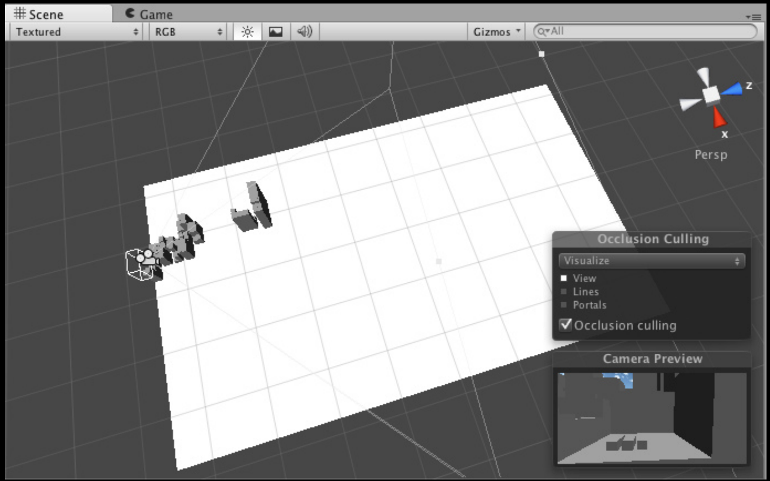
Real Time Occlusion Culling

Ilya Seletsky

Advisor: Zoe Wood



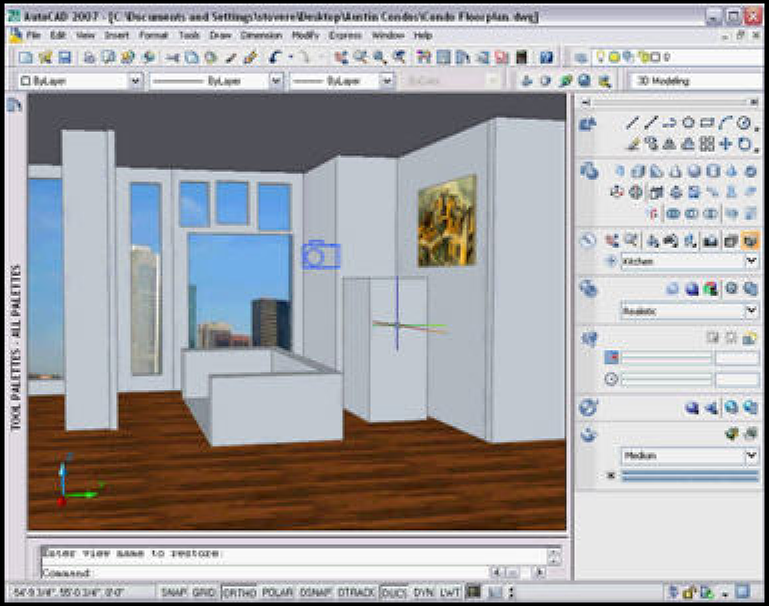
☐ Occlusion culling



☑ Occlusion culling

Real Time Graphics

-30 FPS (33.33 ms per frame)
-60 FPS (16.66 ms per frame)
-Useful for games, CAD applications, etc...

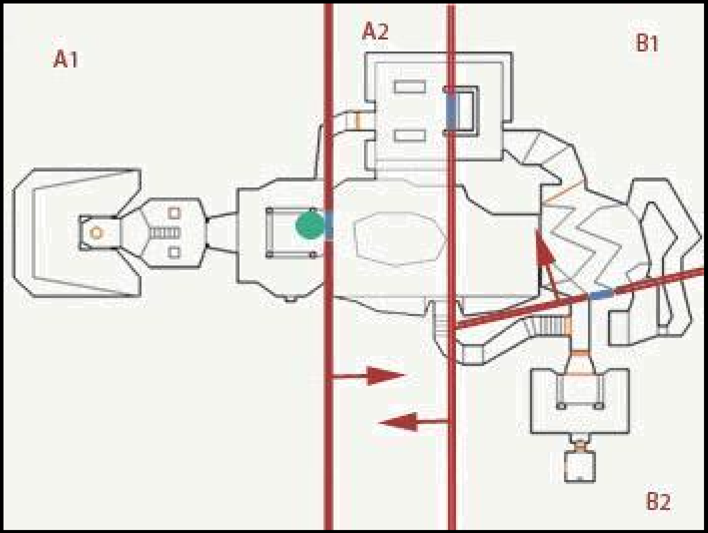

AutoCAD 2007 - [C:\Documents and Settings\stovere\Desktop\Austin Condos\Condo Floorplan.dwg]
File Edit View Insert Format Tools Draw Dimension Modify Express Window Help
ByLayer
ByLayer
ByLayer
3D Modeling
TOOL PALETTES - ALL PALETTES
Kitchen
Realistic
Medium
SNAP GRID ORTHO POLAR OSNAP OTRACK DUCS DYN LWT

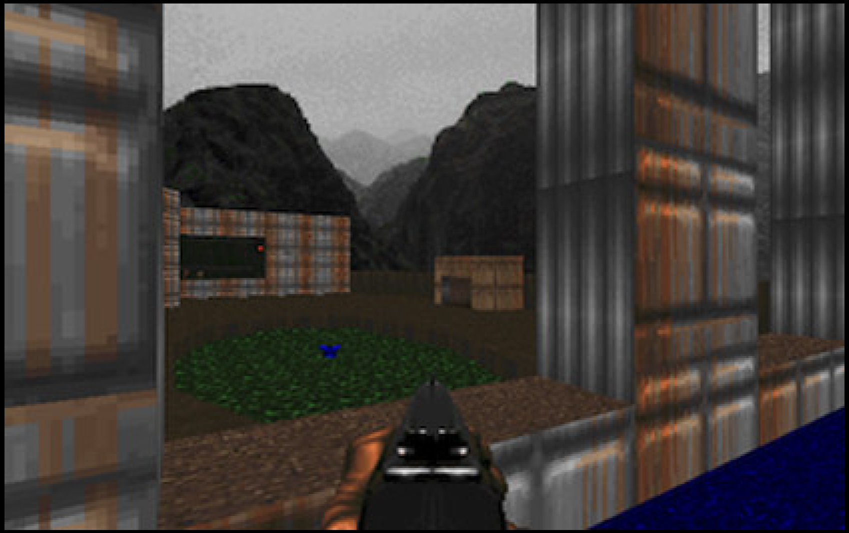
```
Enter view name to restore:
Command:
```

Occlusion Culling

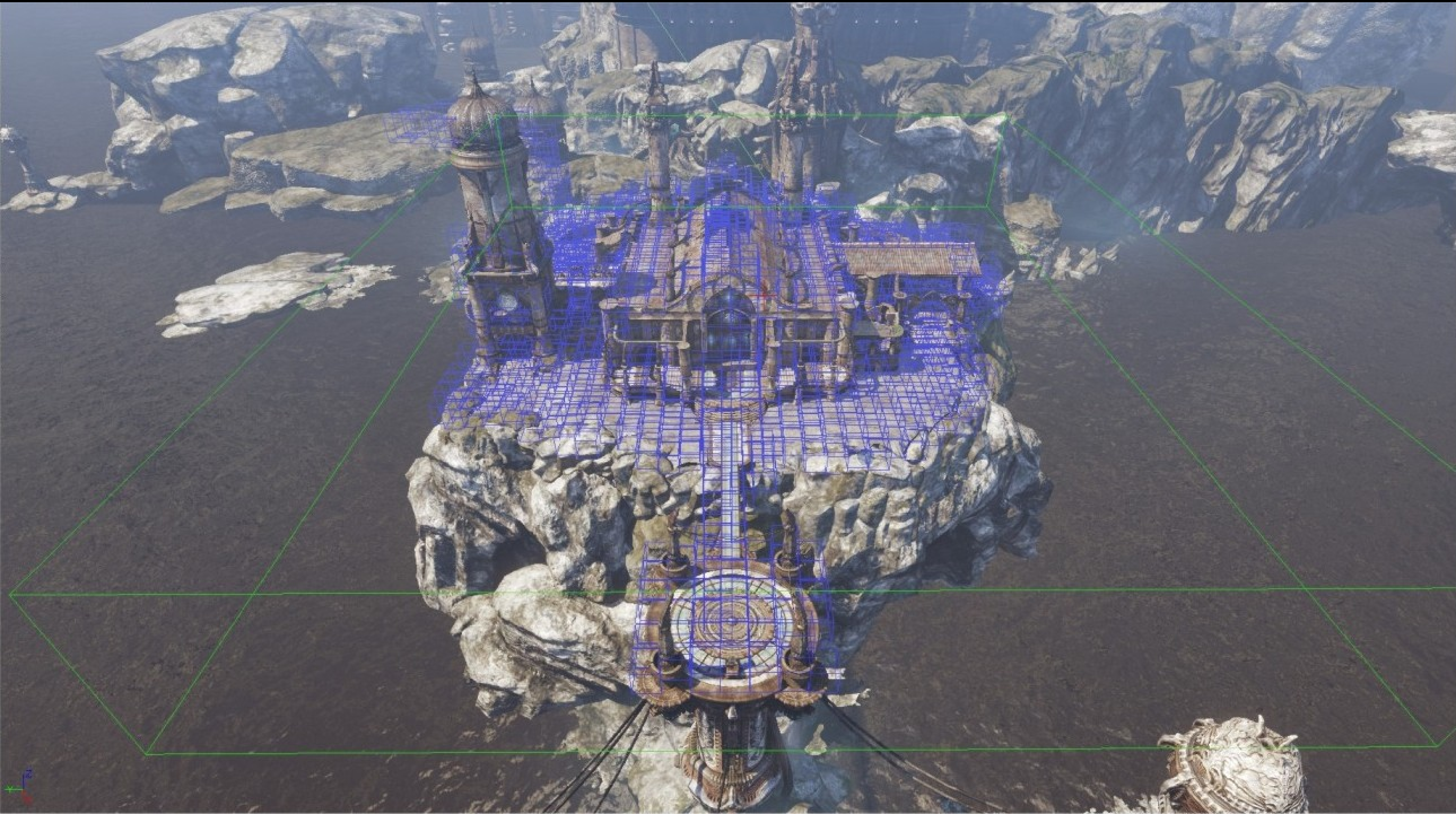
-Figure out what not to draw
-Back in the day was statically prebaked

BSP Trees Keeping DooM running at 35 FPS on 66Mhz 486 CPU and 8MB RAM





Unreal Engine 3 Prebaked Visibility Grid

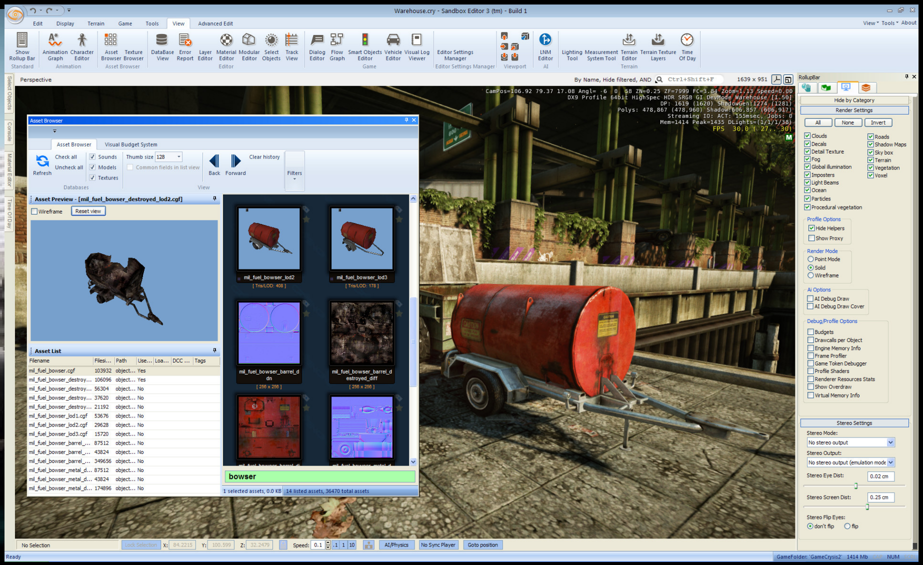


Real Time Occlusion Culling

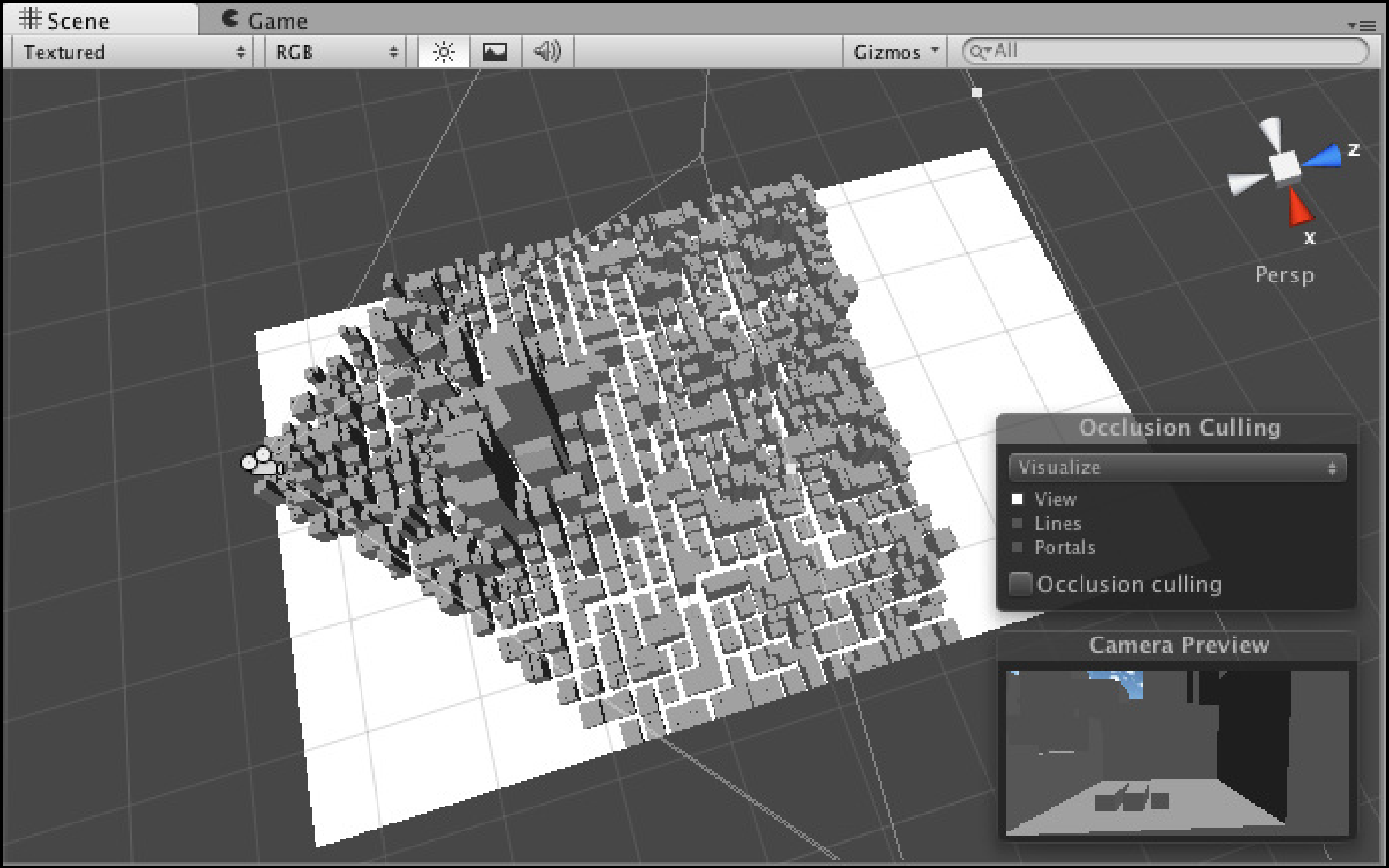
-No prebaking allows for more dynamic and flexible environments

Buildings are going down. Where's your prebaked static visibility data now!!!!

Editing a level in real time. Where's your prebaked static visibility data now!!!!



View Frustum Culling Alone



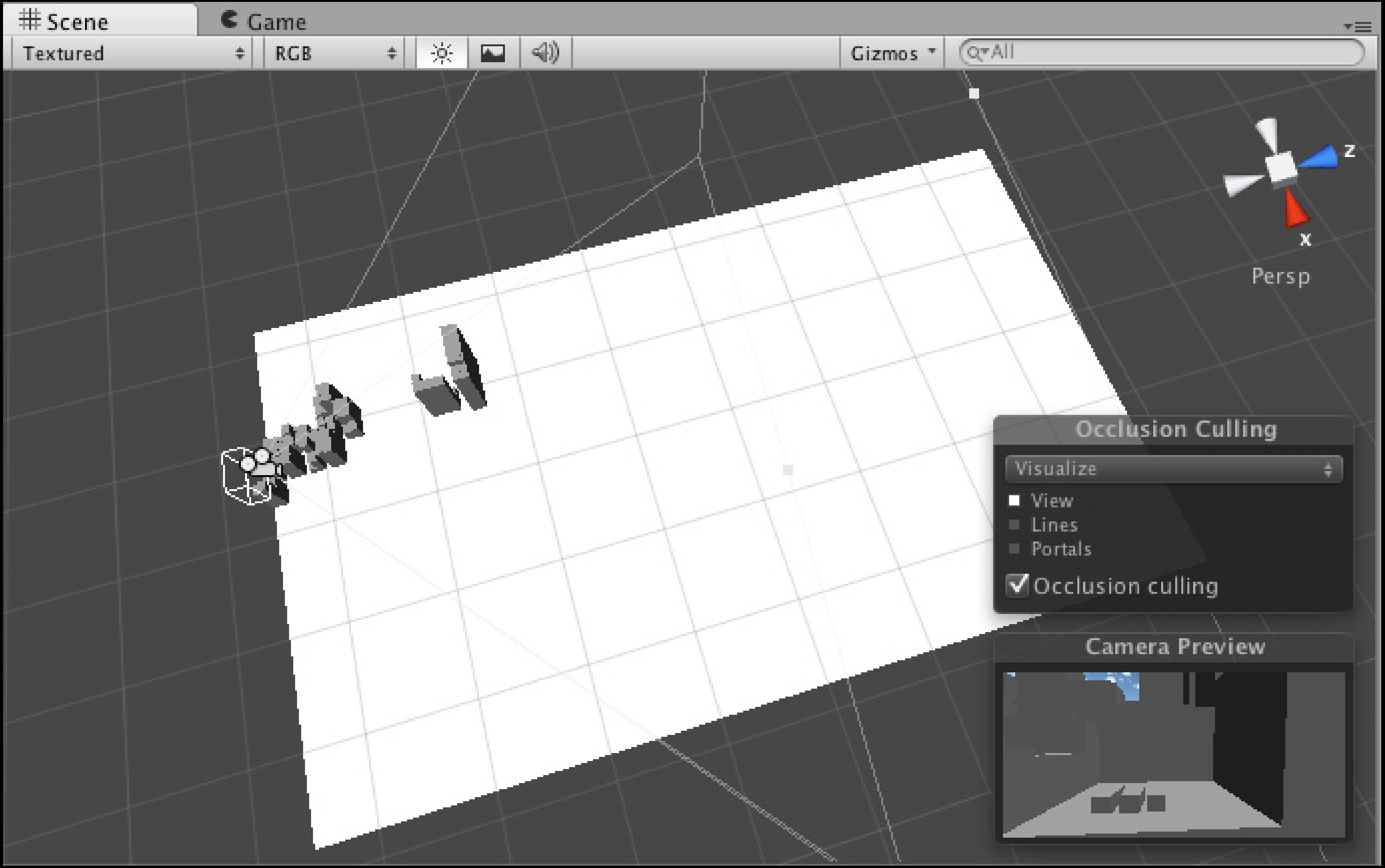
Occlusion Culling

Visualize

■ View
□ Lines
□ Portals

□ Occlusion culling

VFC+Occlusion Culling



Occlusion Culling

Visualize

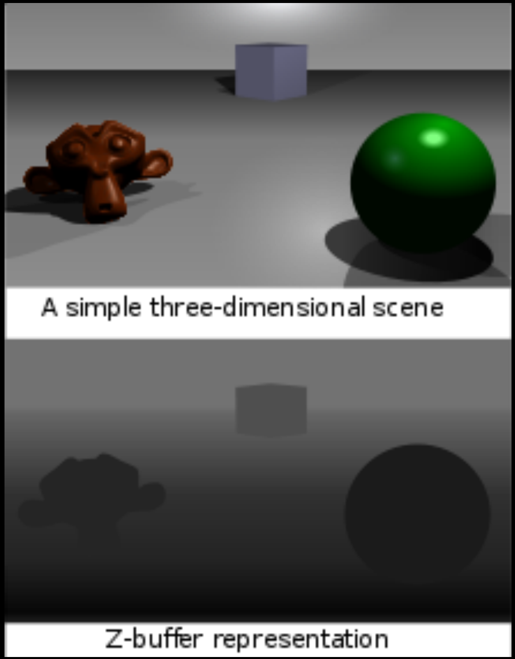
■ View
■ Lines
■ Portals

☑ Occlusion culling

Camera Preview

Depth Test

- Lowest level of occlusion culling
- Z Buffer
- About to draw a pixel
  - Skip if this pixel is already behind what's currently drawn
- Opaque Objects Only

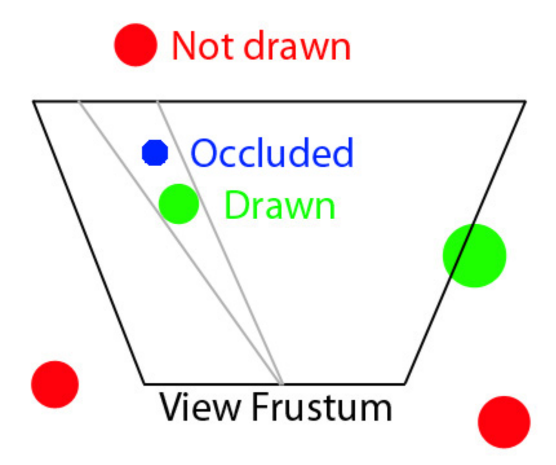


A simple three-dimensional scene

Z-buffer representation

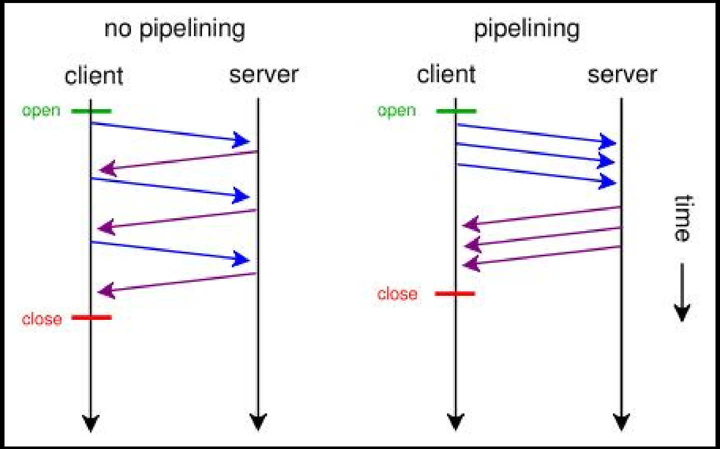
Hardware Occlusion Query

- Begin Query
- Draw Object
- End Query
- Retrieve how many pixels passed depth test
  - Know if object is visible
  - Know what level of detail to use



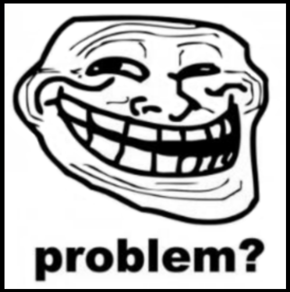
Hardware Occlusion Query

- CPU issuing asynchronous calls to GPU
  - Stalls
- Batch occlusion queries for best results
  - Don't draw object 0, retreive result, draw object 1, retreive result
  - Draw x number of objects
  - Retrieve x number of results



Hardware Occlusion Query

- Use query results next frame
- Objects will pop in a frame late
- Human eye usually won't notice at interactive FPS
    - 33.33 ms at 30 FPS
    - Some modern games do this and you didn't even notice



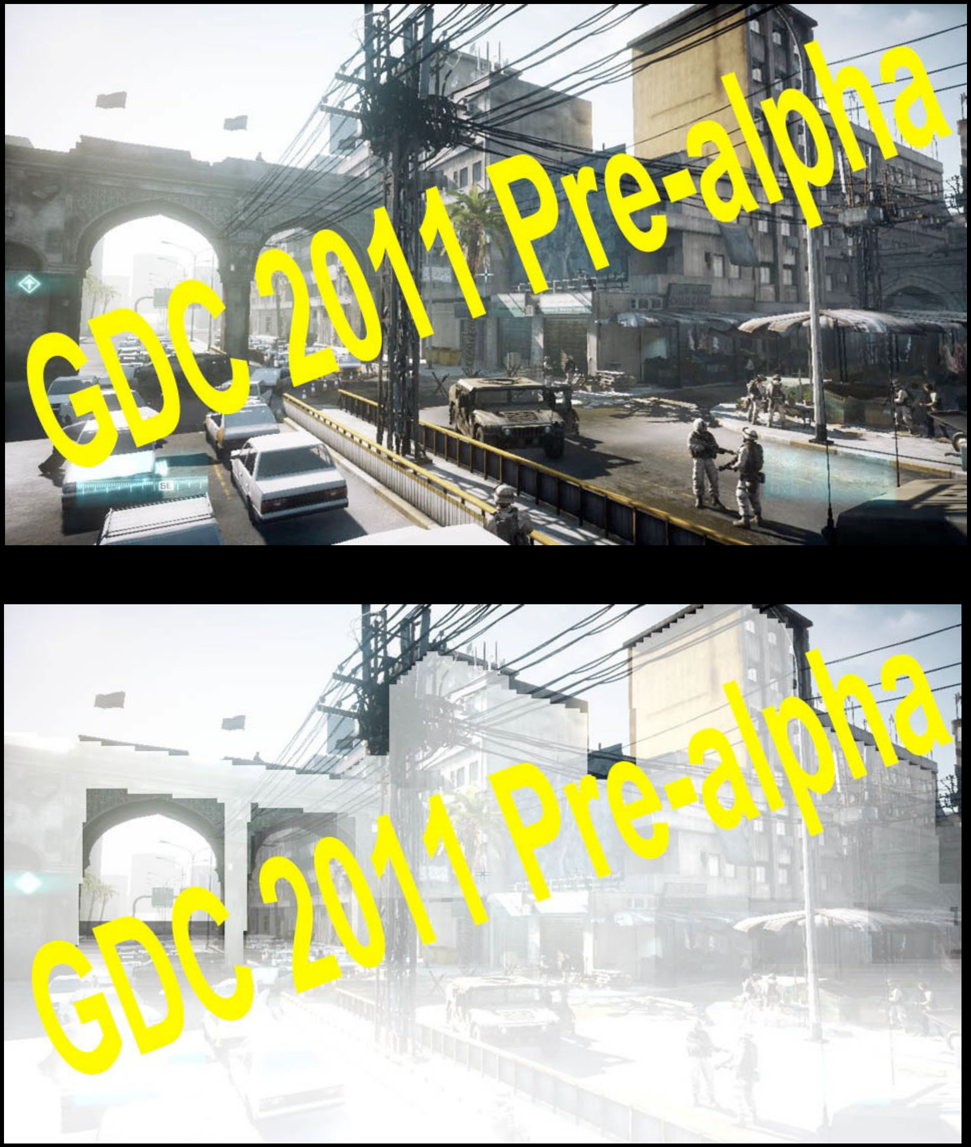
You will now pay close attention and try to notice it in all your games...

Hardware Occlusion Query

- Render objects front to back
- Use results next frame
- First Test
  - Disable depth write and color write
  - Render simple box
- Get results back
  - If passed last frame render actual object
  - If not, go back to step 1
- Get results back
  - If passed last frame render actual object
  - If not, go back to step 1

Software Occlusion Culling

- Avoids hardware query drawbacks
  - No CPU stall waiting for GPU results
  - Use results same frame
- Software rasterize simple geometry for large occluders
- Test objects against software buffer with simple box
- Used in Cryengine and Battlefield 3



From Dice's Presentation: Culling the Battlefield

Why try Hardware?

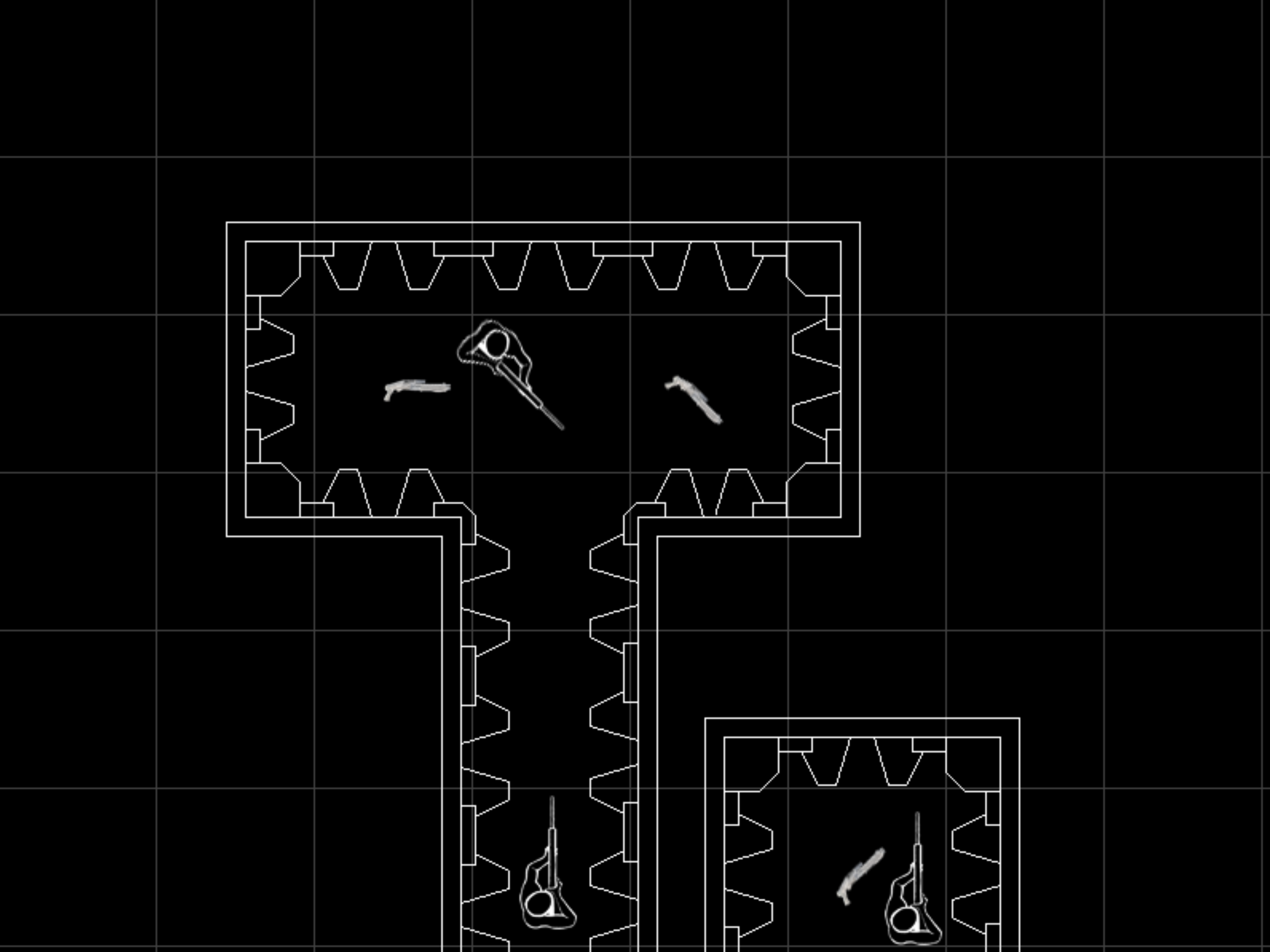
- GPU can draw lots of geometry EASY

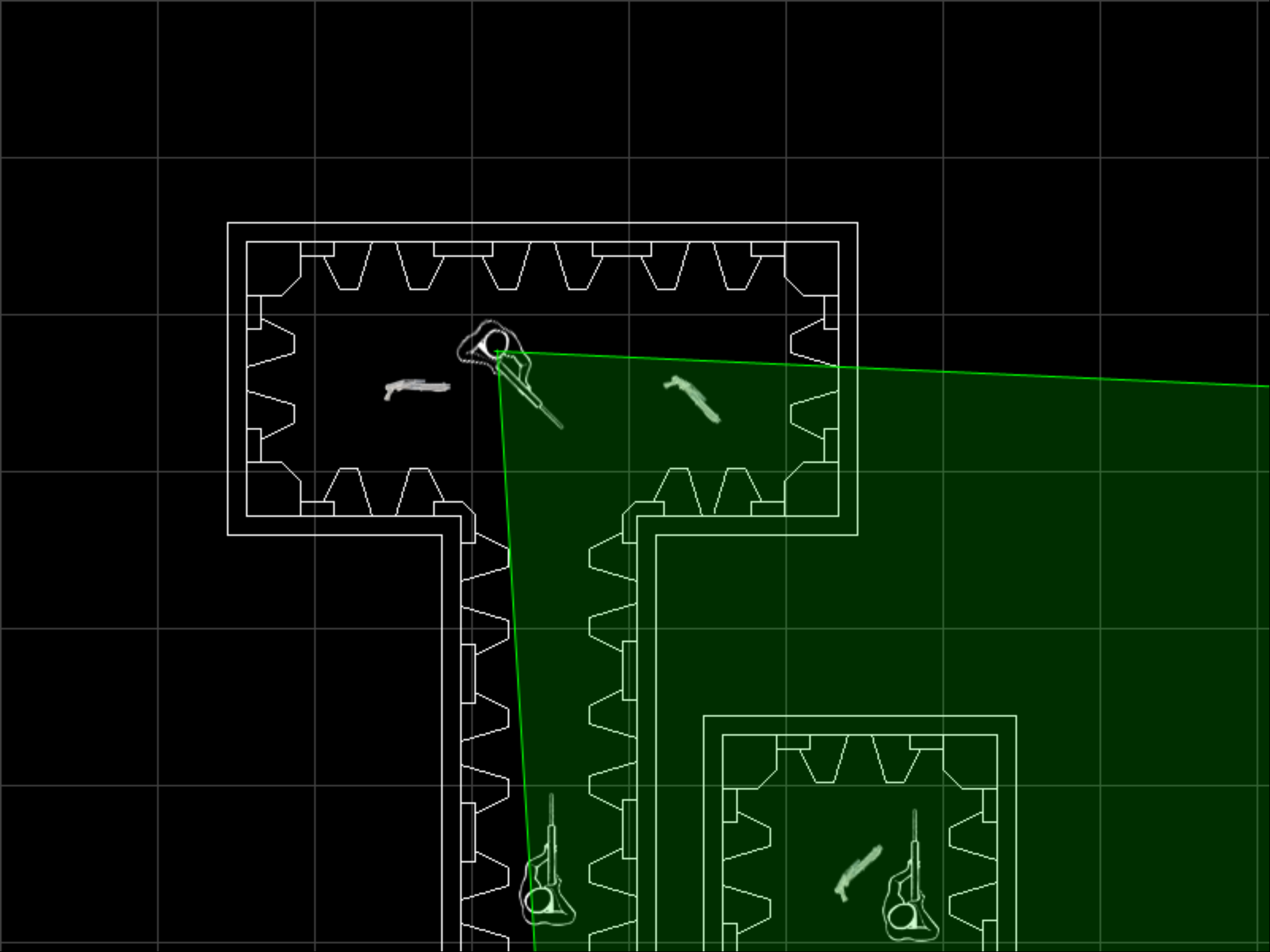
So why do occlusion culling instead of drawing EVERYTHING!!!!

- Large draw distance = MAAANY objects
- Bottleneck is transferring data to GPU
  - Transfer simple box model once
  - Render MAAAANY boxes
  - Later transfer detailed data only for visible objects
    - Geometry
    - Textures
    - Running complex shaders....

My implementation so far

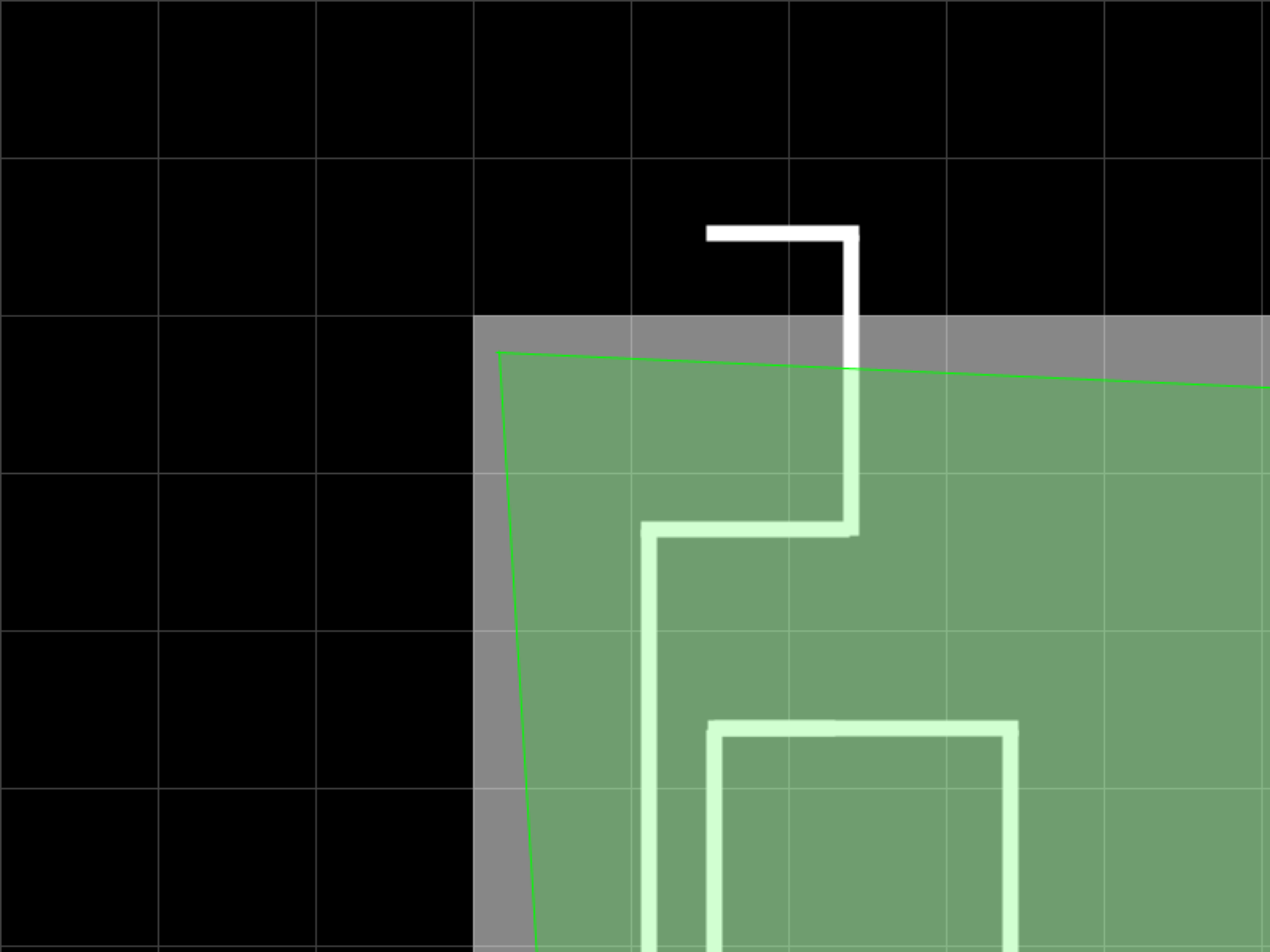
- 3D Uniform Grid Scene
  - Hierarchical structure like Octtree wouldn't work well for this
- 2 Passes in one frame
  - First pass
    - Figure out visible 3D uniform grid cells
  - Second pass
    - Draw objects





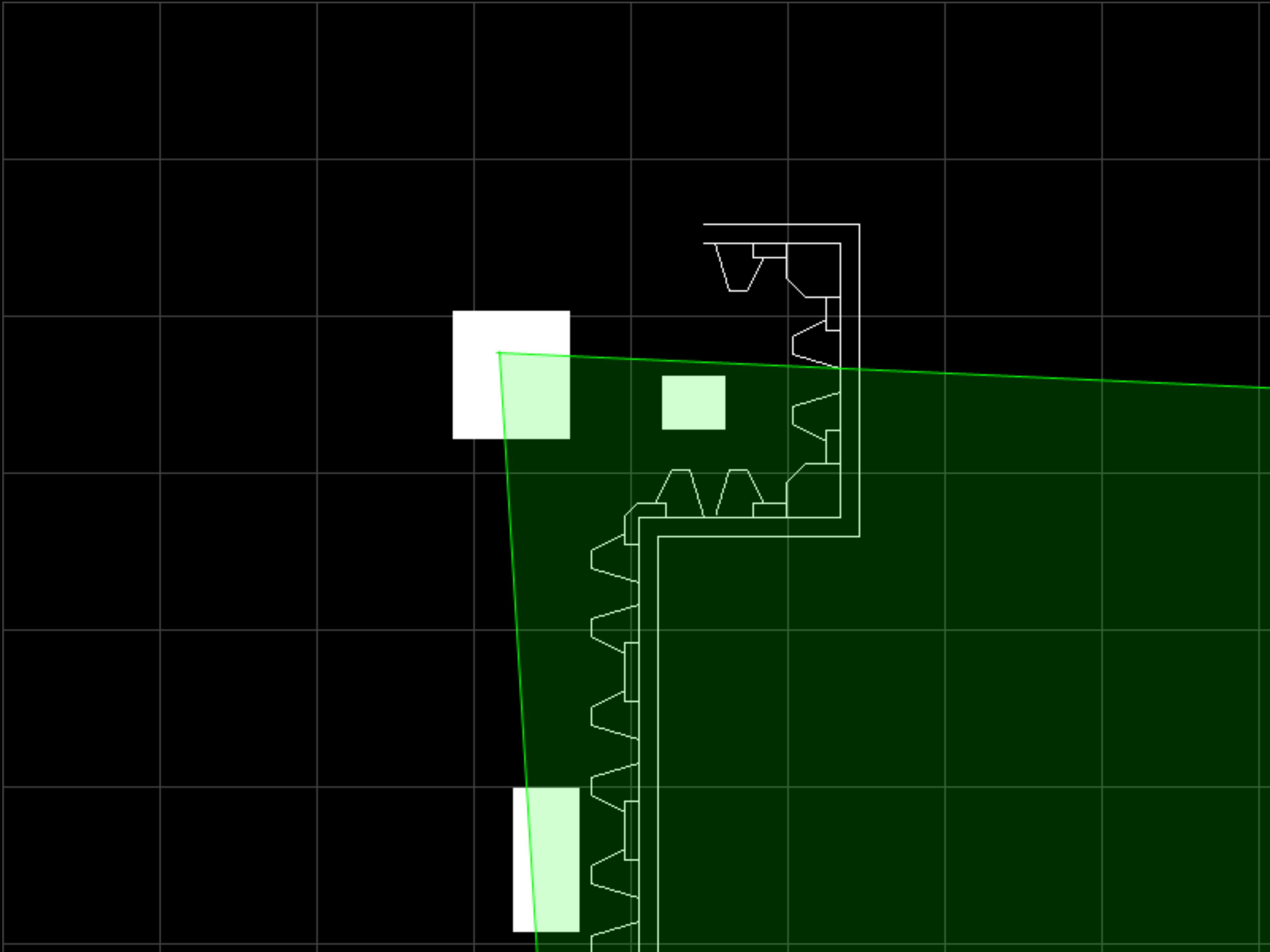
First Pass

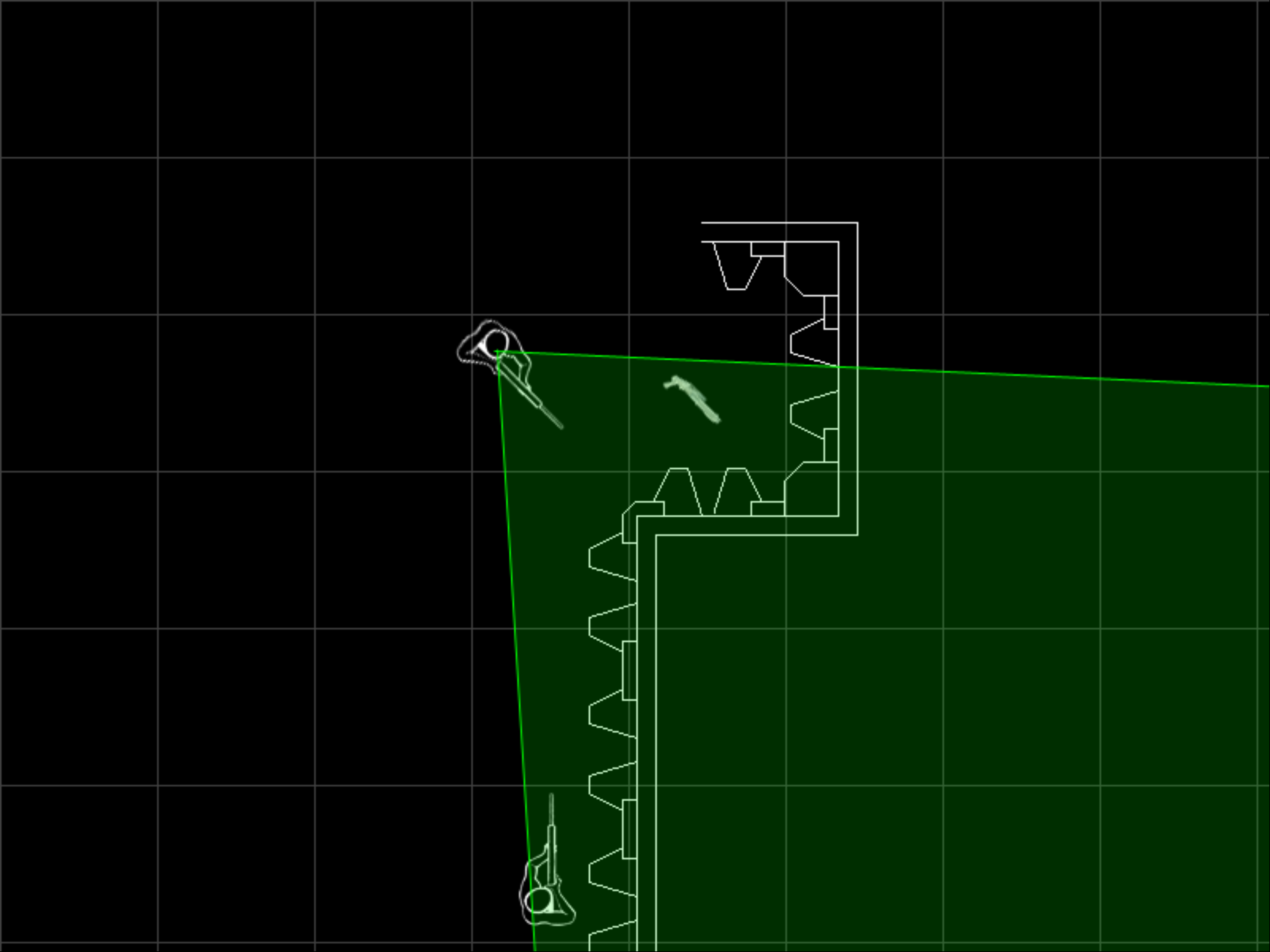
- Render large occluders to populate depth buffer
  - Simplified geometry that's fully contained by visual geometry
  - Color write off, depth write on
- Figure out visible 3D uniform grid cells with occlusion queries
  - Color write and depth write off, query only
- Use result in same frame
  - Might be inefficient and cause a stall, I'll figure out if this is the case later



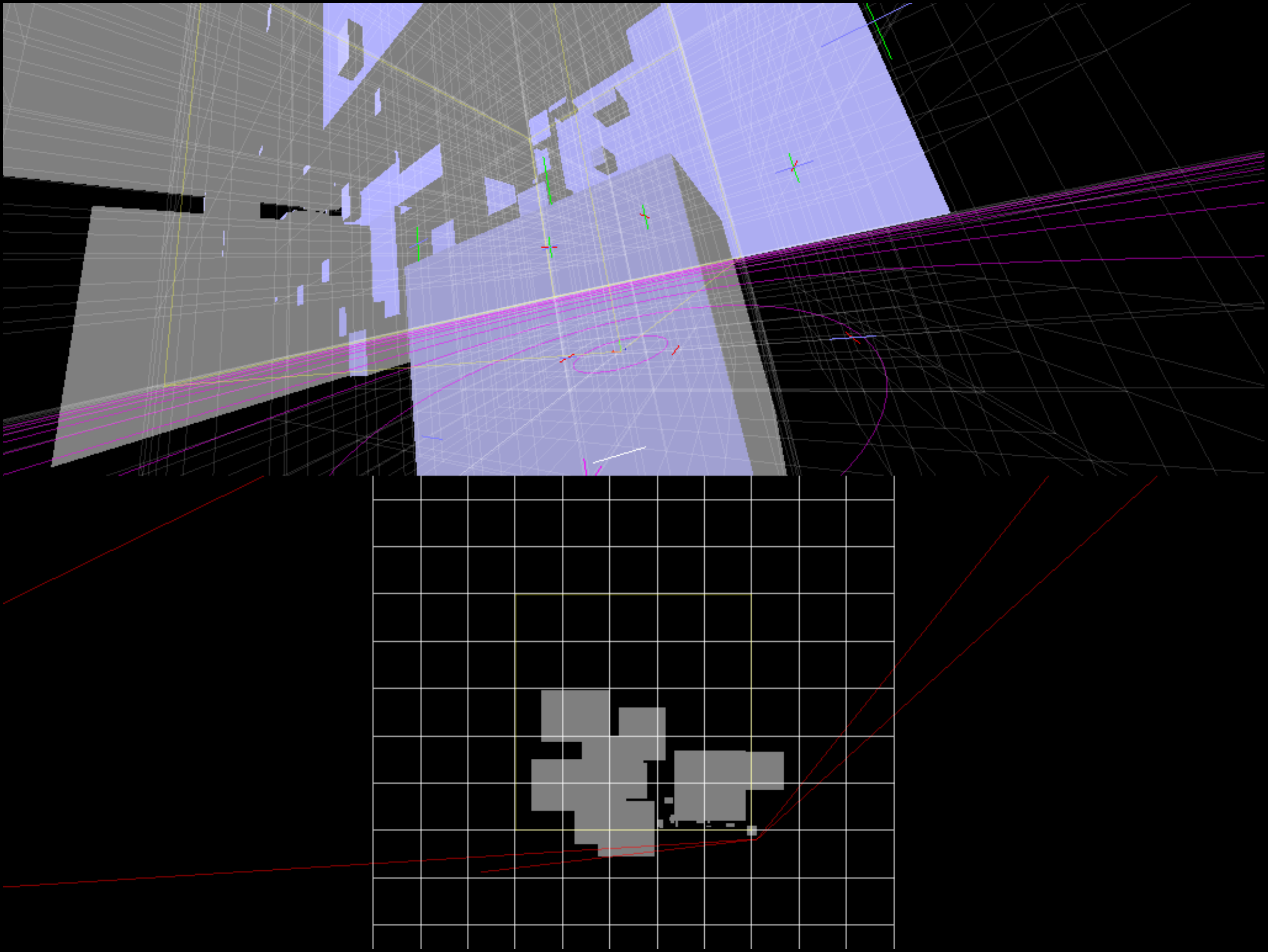
Second Pass

- Draw objects in the visible 3D uniform grid cells and do traditional occlusion culling queries
- Use those results in later frames





Quick Demo



Questions?