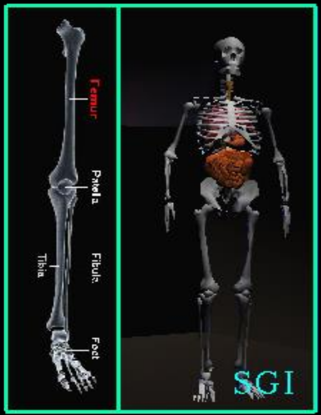
Interactive Visualization of Complex Scenes

COS 426 Spring 2015
Princeton University

Interactive Visualization

Render images with interactive control of viewpoint

Mechanical CAD

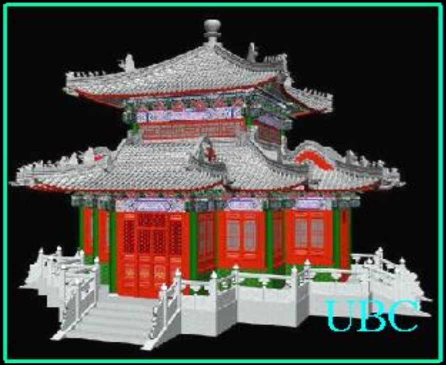


Medicine

Driving Simulation

Entertainment

Architectural CAD



Education

Interactive Visualization Goals

Realism

- Realistic enough to convey information

Frame rate:

- >10-60 frames per second

Latency

- <10-500 milliseconds response delay

Computation

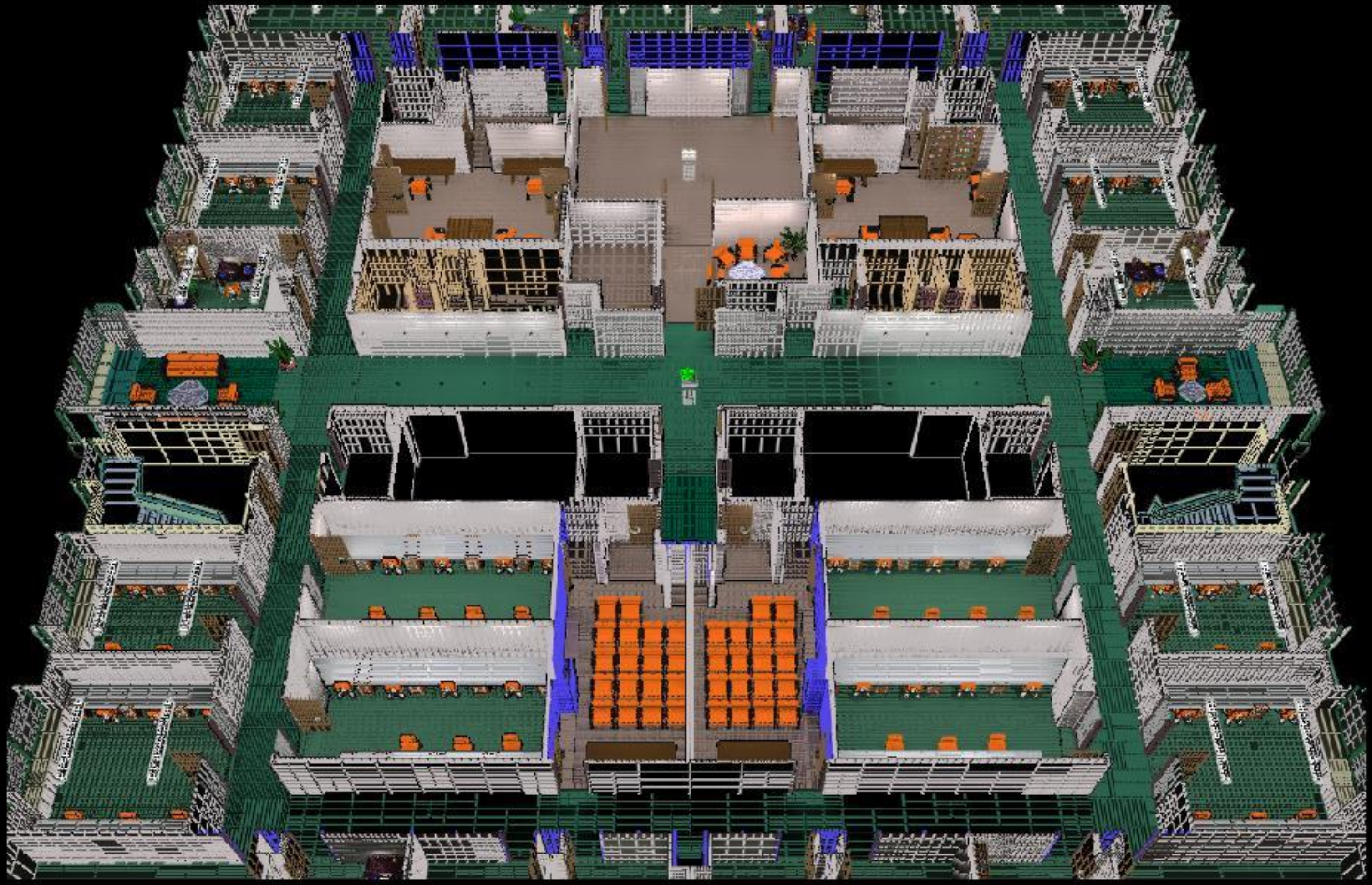
- Fast preprocessing
- Fast startup
- Low storage
- etc.

Scene Complexity

Examples:

- Automobile: ~20,000 parts
- Boeing Airplane: ~2,000,000 parts
- Aircraft Carrier: ~20,000,000 parts
- Sculptures: ~200,000,000,000 samples
- Outdoor Environments

Architectural Models

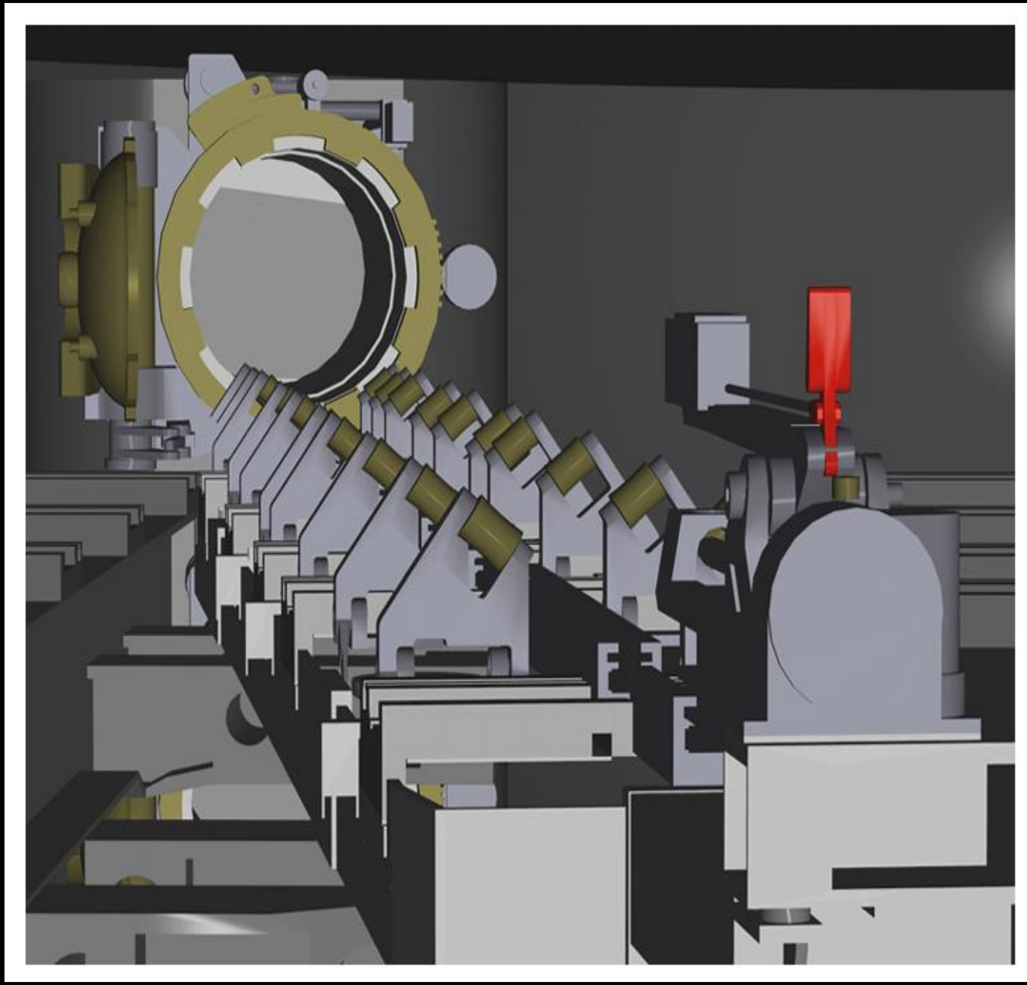


Soda Hall Model (7.6M polygons)

Structural Engineering Models

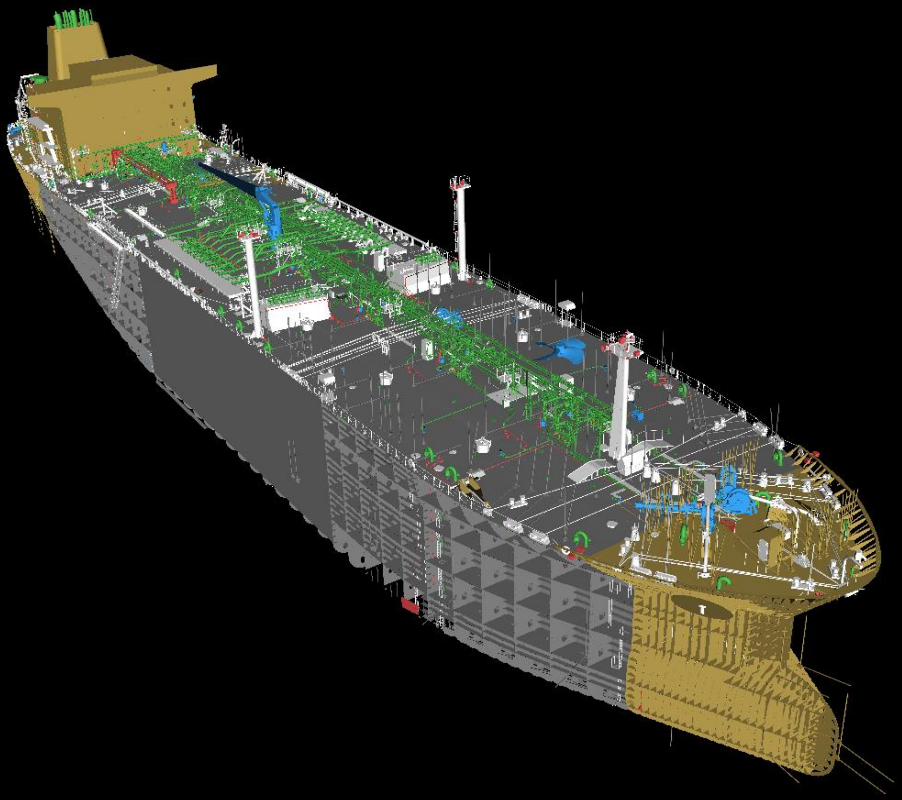
Coal-Fired Powerplant: 15 million triangles

Mechanical CAD Models



Submarine Torpedo Room (850K polygons)

Mechanical CAD Models



82 million triangles; 126,000 objects

Newport News Shipbuilding

Video Games

Portal

Rendering Acceleration Techniques

Visibility Culling

- Backface culling, view-frustum culling, occlusion culling, ...

Detail Elision

- Levels of detail, multiresolution, ...

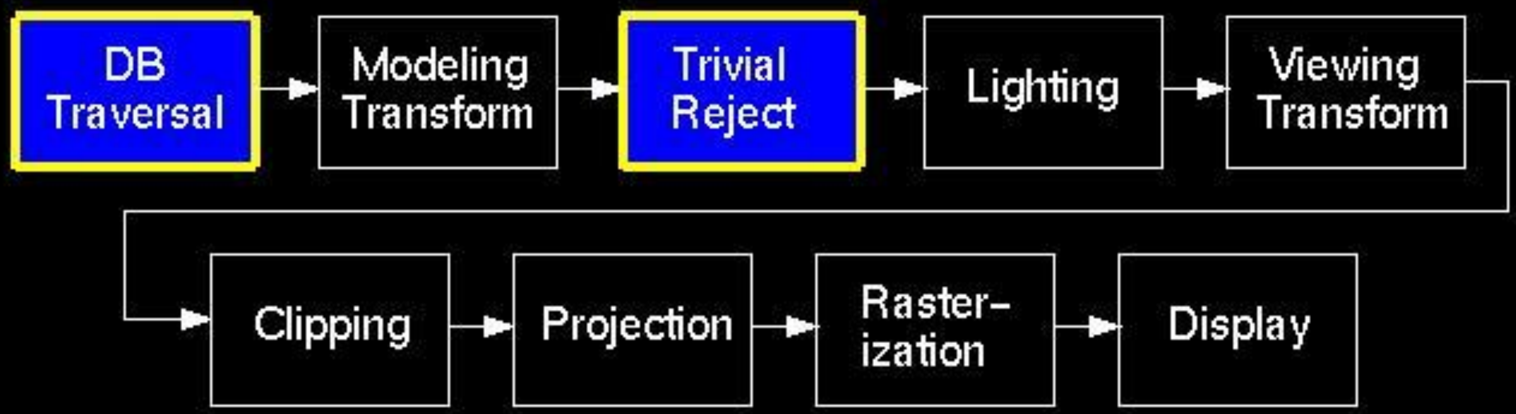
Images

- Textures, billboards, imposters, ...

Visibility Culling

Quickly eliminate large portions of the scene that will not be visible in the final image

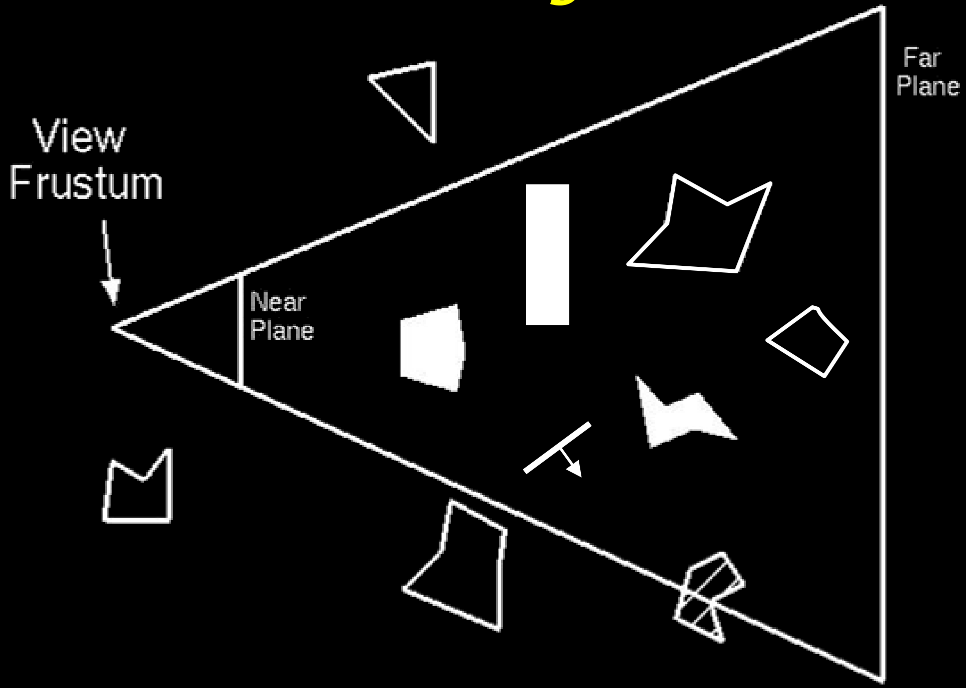
- Not the exact visibility solution, but a quick and conservative test to reject primitives that are not visible
  - Trivially reject stuff that is obviously not seen
  - Use Z-buffer and clipping for the exact solution



Visibility Culling

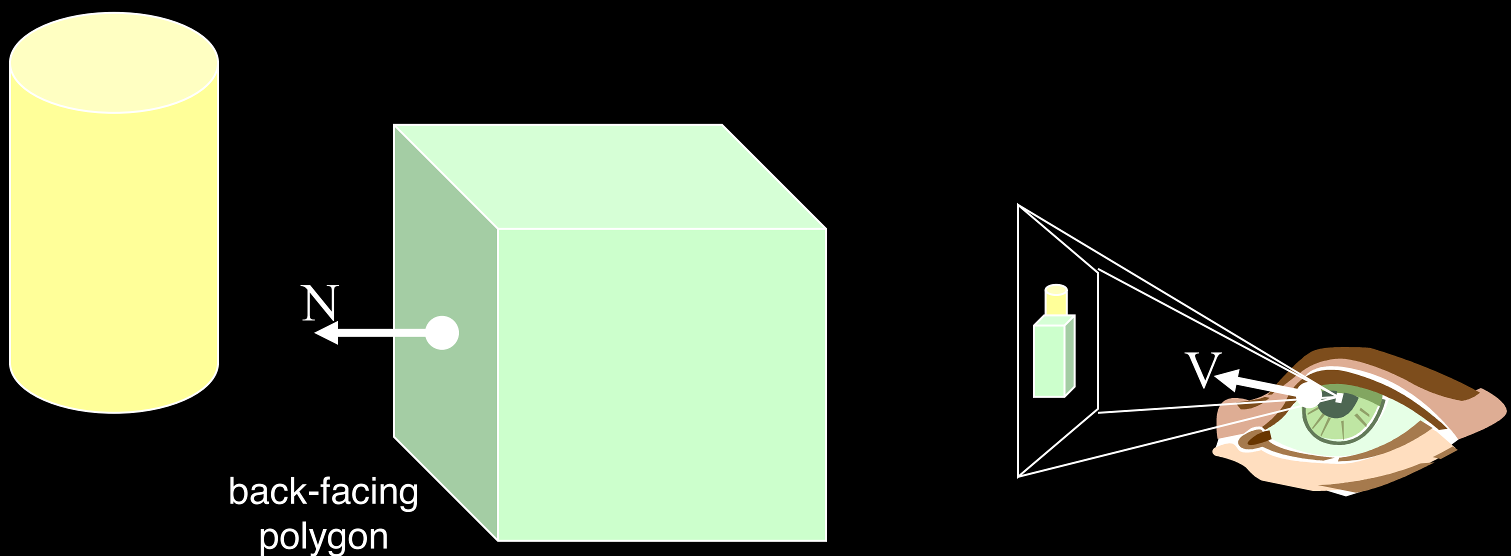
Basic idea: don't render what can't be seen

- Facing away from camera: *backface culling*
- Off-screen: *view-frustum culling*
- Occluded by other objects: *occlusion culling*



Back-Face Culling

Do not draw polygons facing backwards with respect to camera



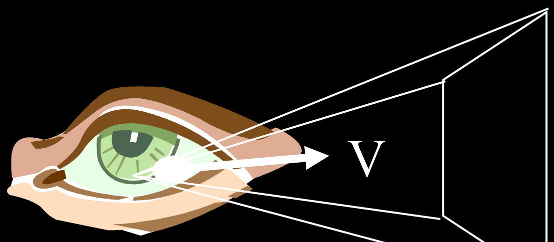
A polygon is backfacing if

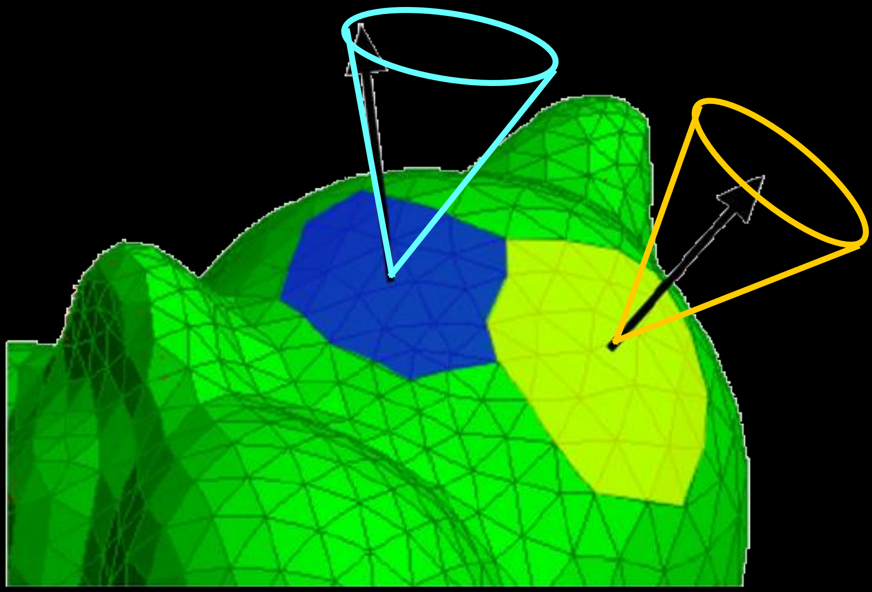
$$V \bullet N > 0$$

Back-Face Culling

Avoid testing every face separately

- Cluster faces
- Precompute range of normals for each cluster
- Check cluster before testing every face

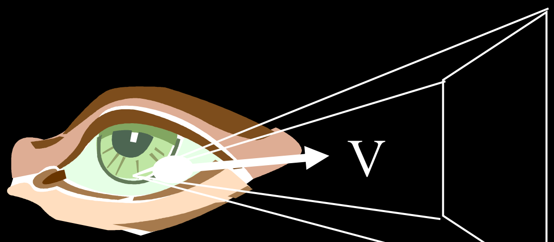


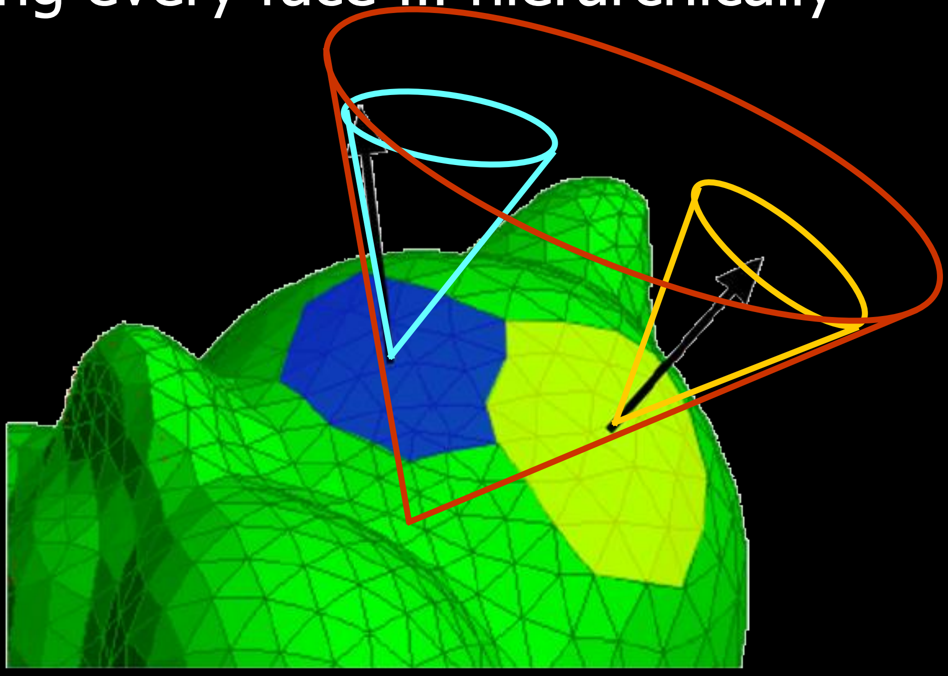


Hierarchical Back-Face Culling

Avoid testing every face separately

- Cluster faces hierarchically
- Precompute range of normals for each cluster
- Check cluster before testing every face … hierarchically

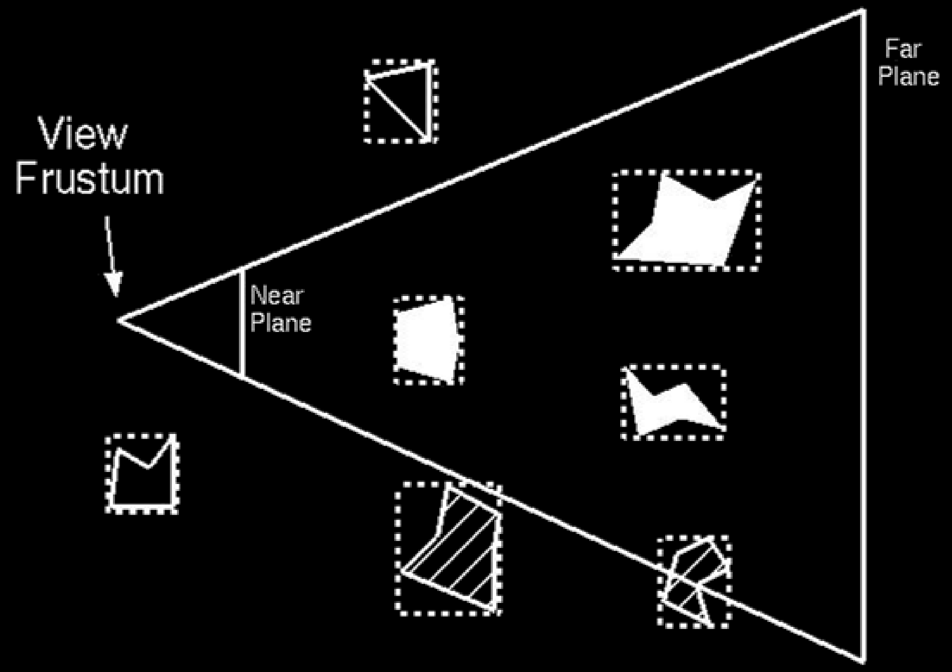




View-Frustum Culling

Don't draw primitives outside the view frustum

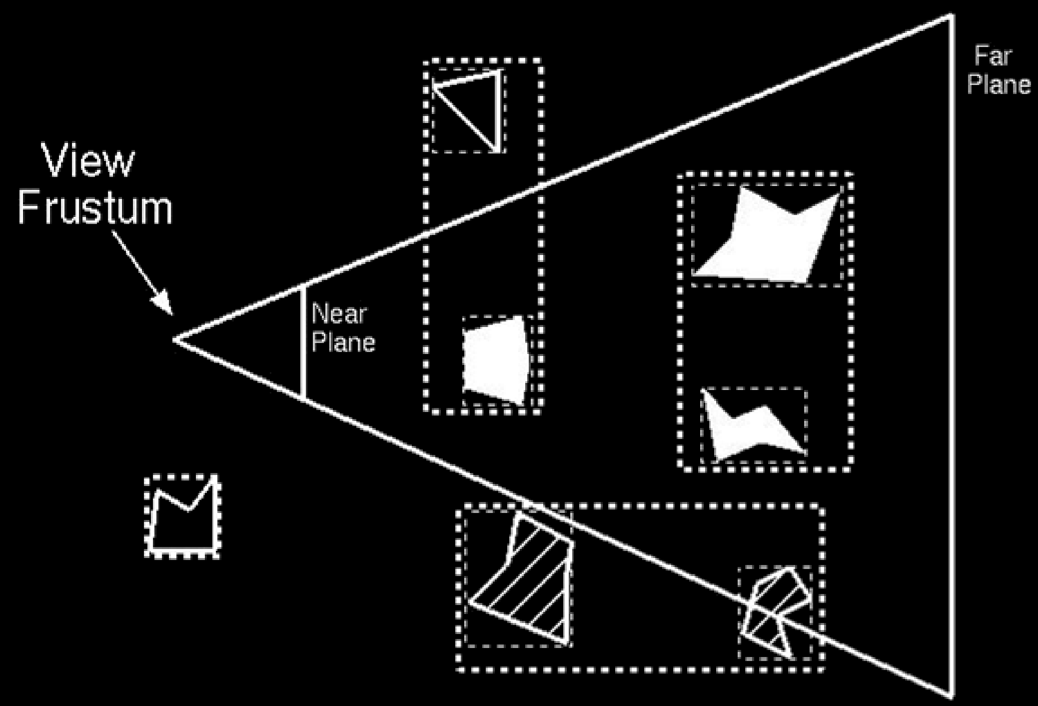
- Organize primitives into clumps
- Before rendering the primitives in a clump, test their bounding volume against the view frustum



View-Frustum Culling

Hierarchical bounding volumes

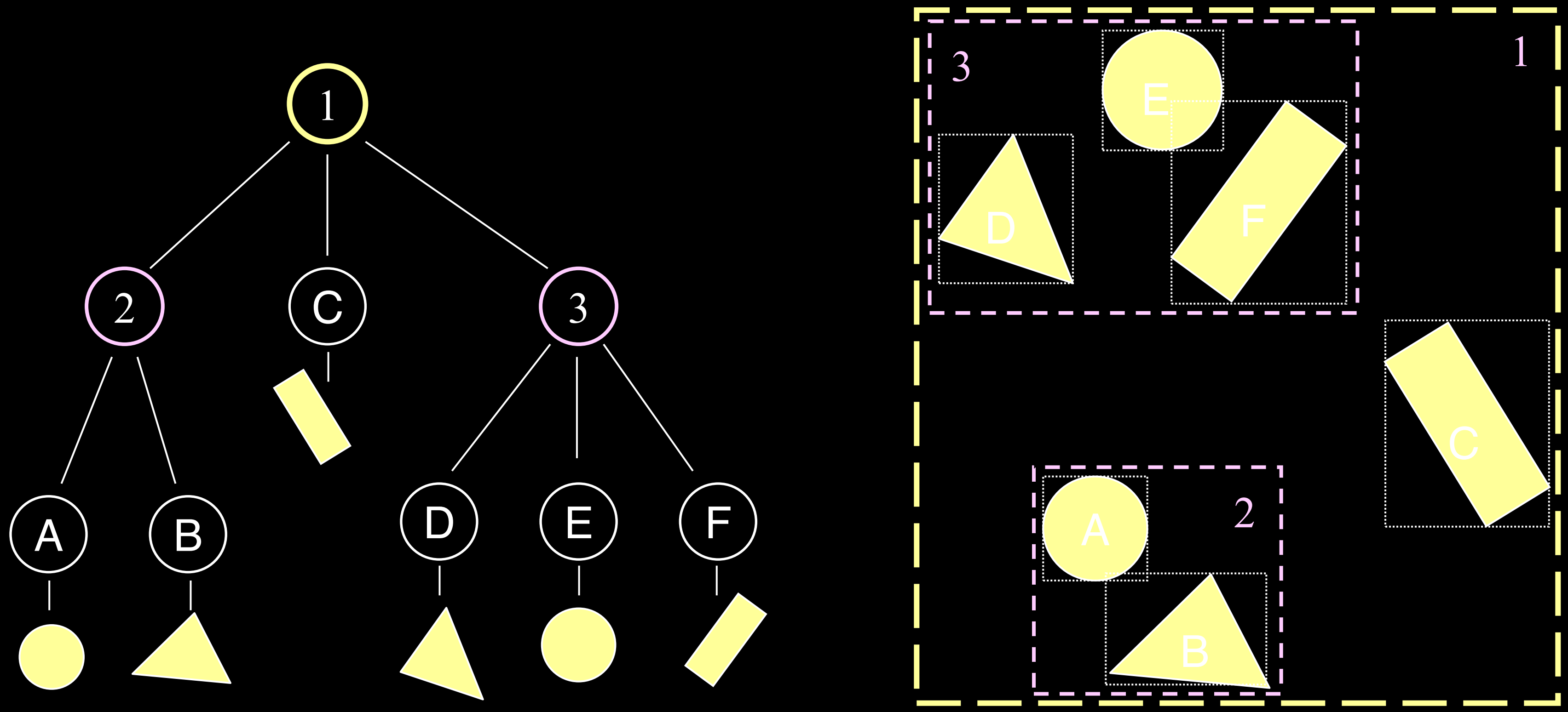
- If a clump is entirely outside or entirely inside view frustum, no need to test its children



Hierarchical View-Frustum Culling

Hierarchical bounding volumes

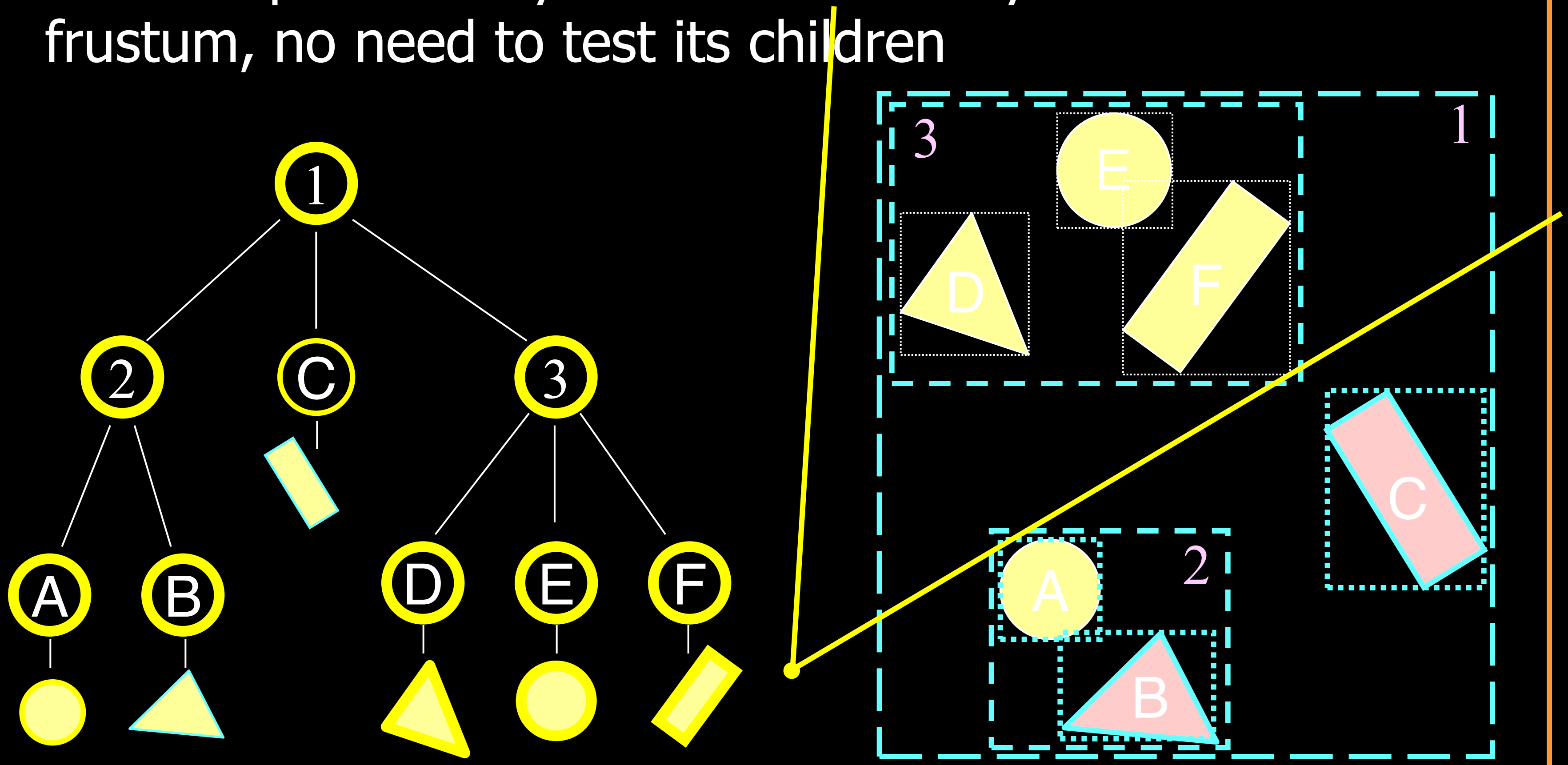
- If a clump is entirely outside or entirely inside view frustum, no need to test its children



Hierarchical View Frustum Culling

Hierarchical bounding volumes

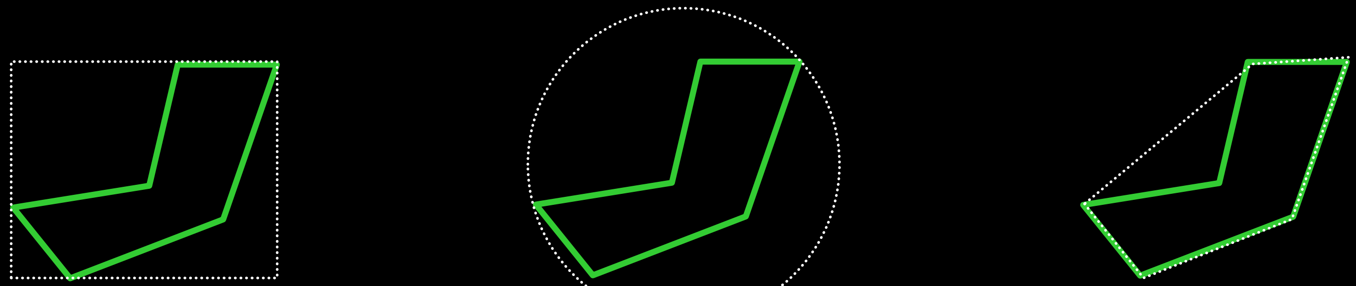
- If a clump is entirely outside or entirely inside view frustum, no need to test its children



Hierarchical View Frustum Culling

What shape should the bounding volumes be?

- Spheres and axis-aligned bounding boxes:
  - Simple to calculate/test
  - May be poor approximation
- Convex hulls:
  - More complex to calculate/test
  - Tighter approximation

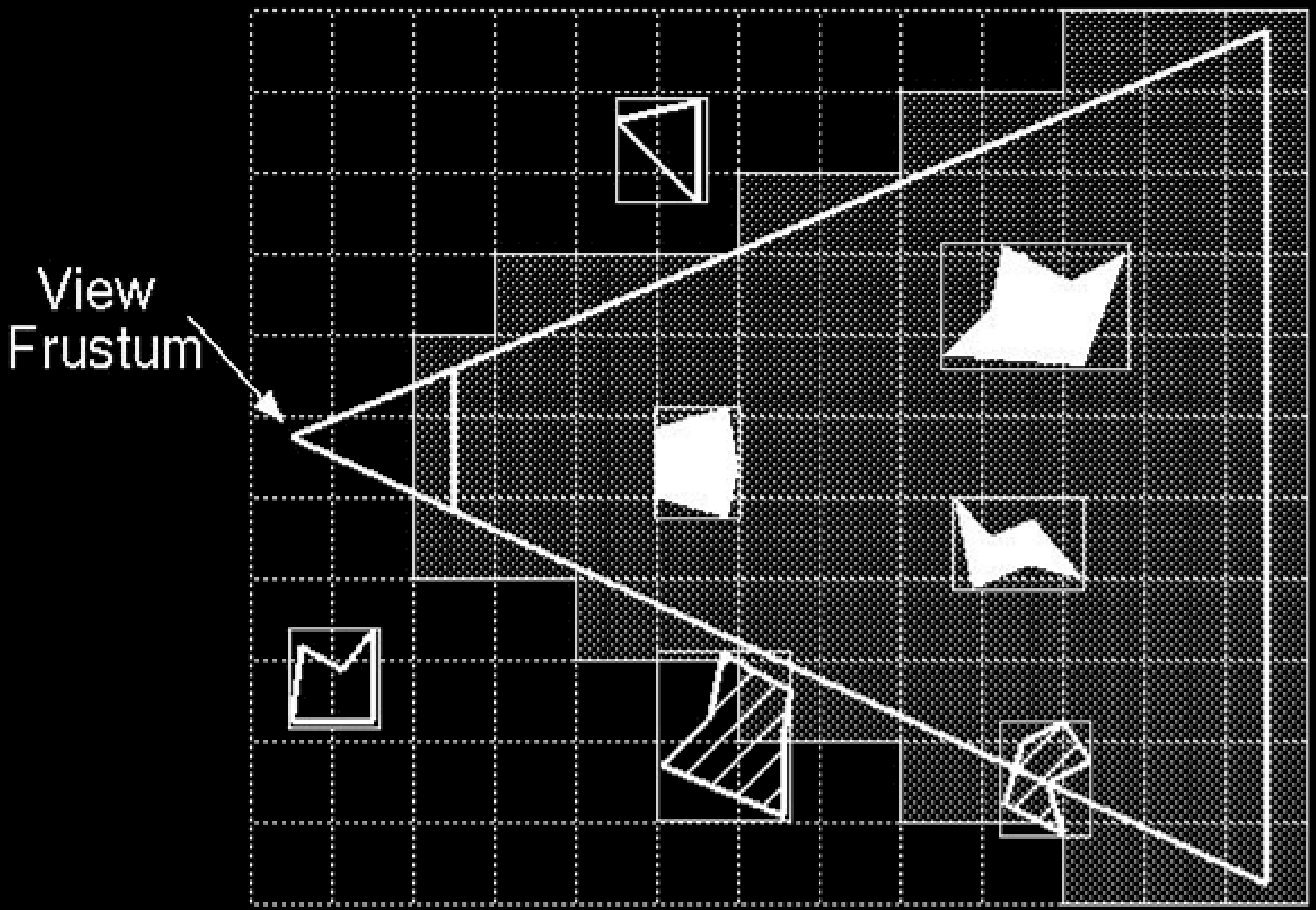


Bounding Box (Axis-Aligned)

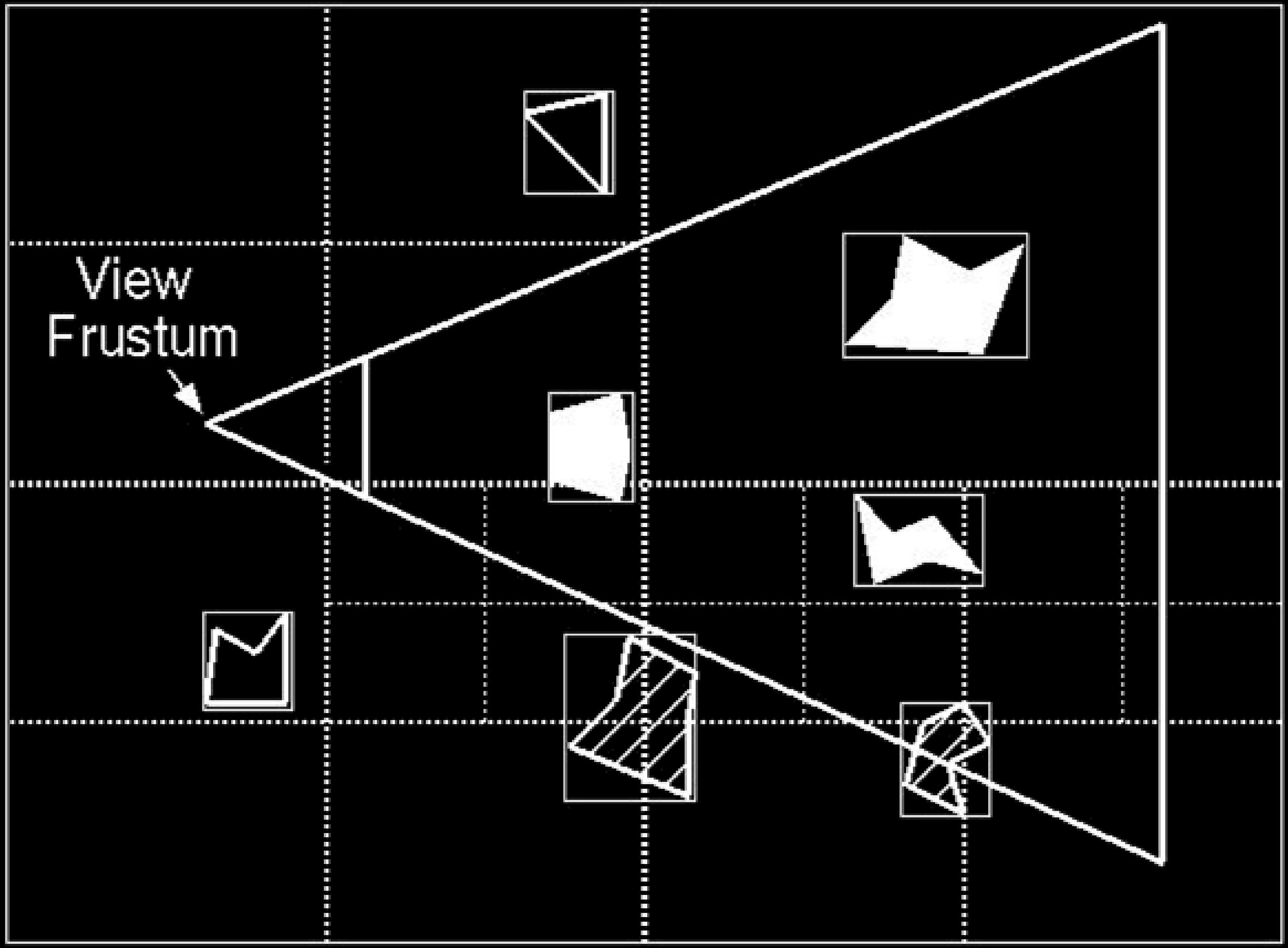
Bounding Sphere

Convex Hull

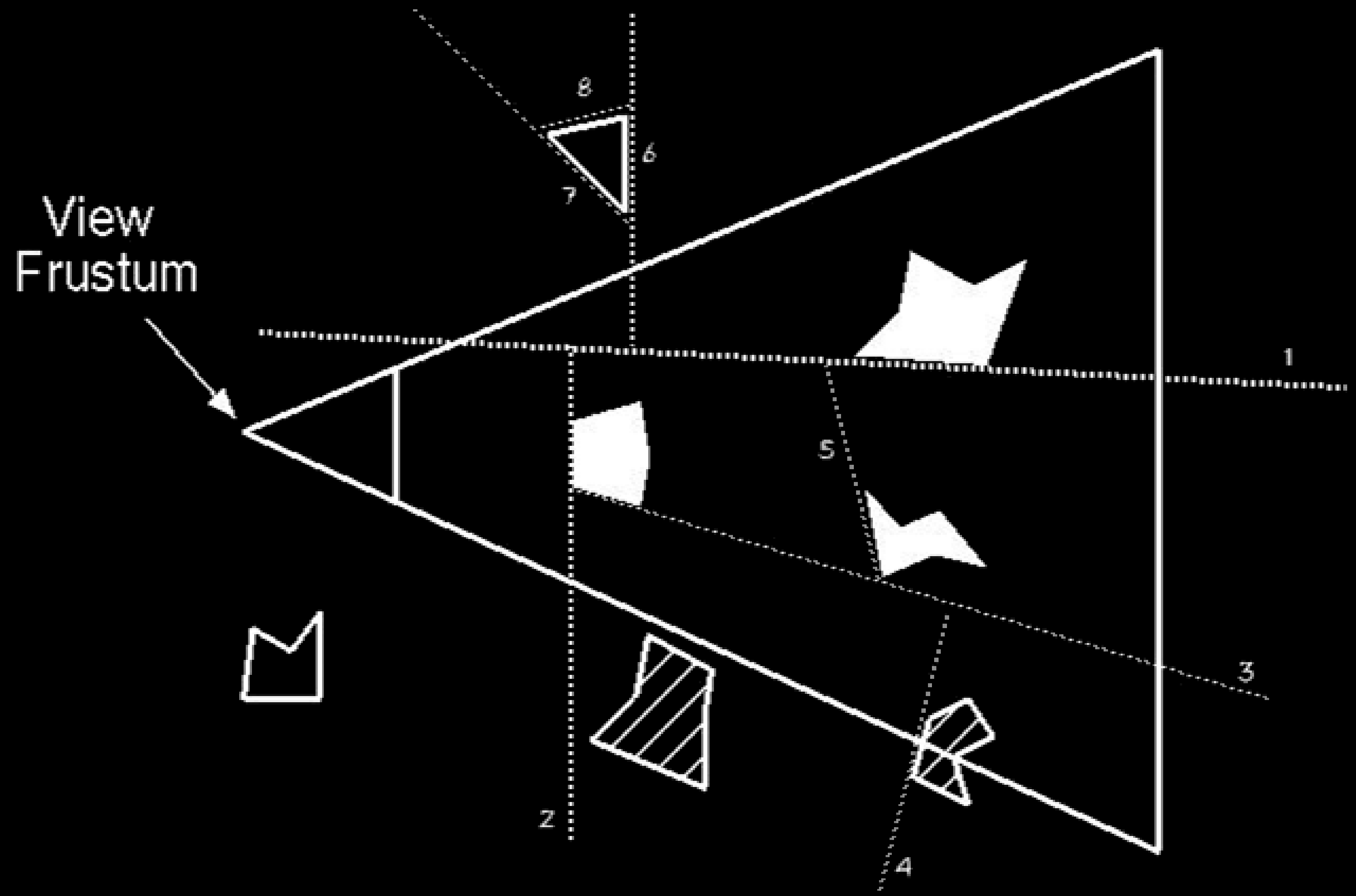
Uniform Grid Subdivision



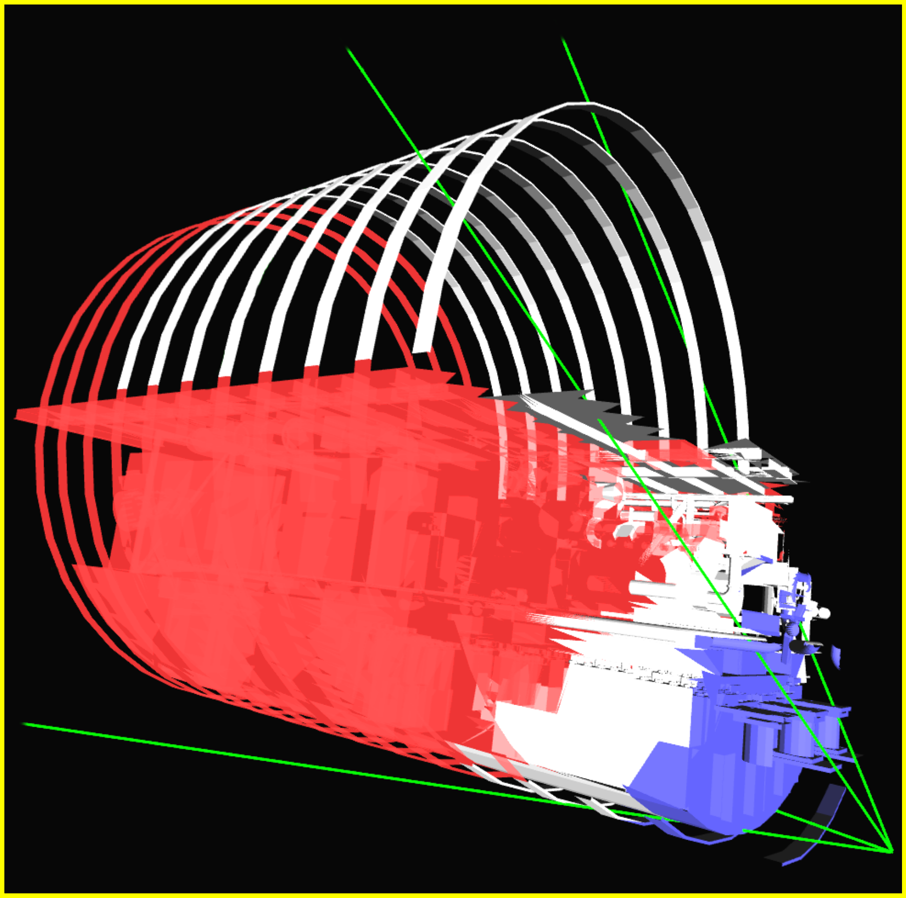
Octree Subdivision



BSP Subdivision



Occlusion Culling



Blue parts: occluders
Red parts: occludees

Occlusion Culling

Object-precision

- Cells and portals
- Shadow volumes

Image-precision

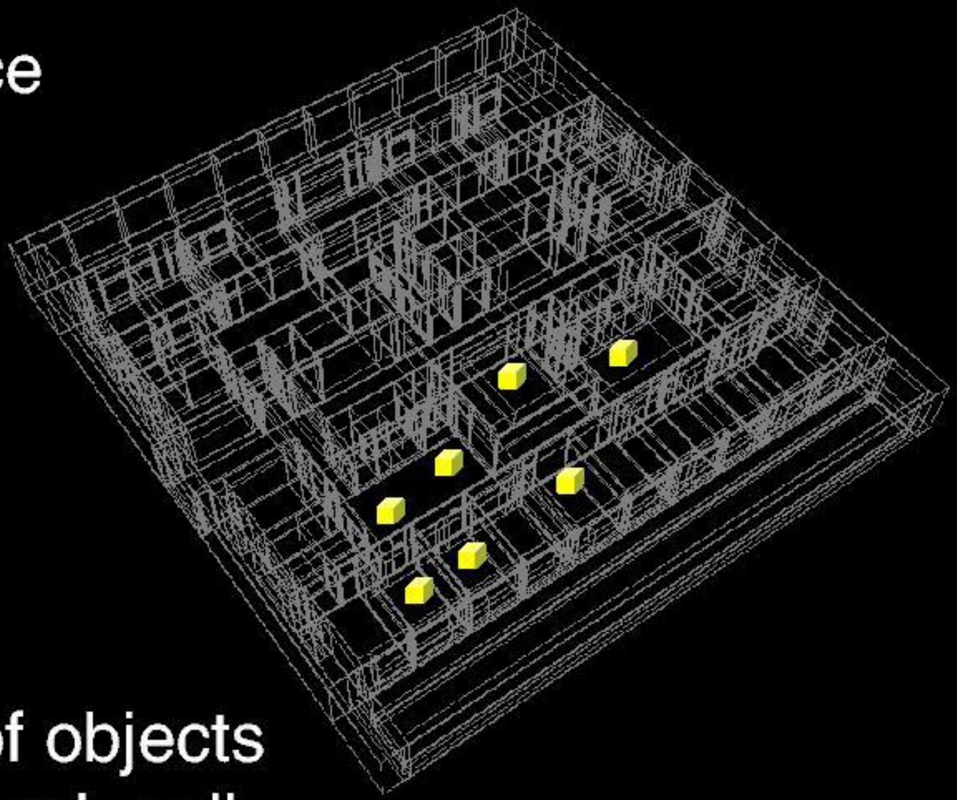
- OpenGL occlusion test
- Hierarchical Z-buffer
- Hierarchical occlusion maps

Cells and Portals

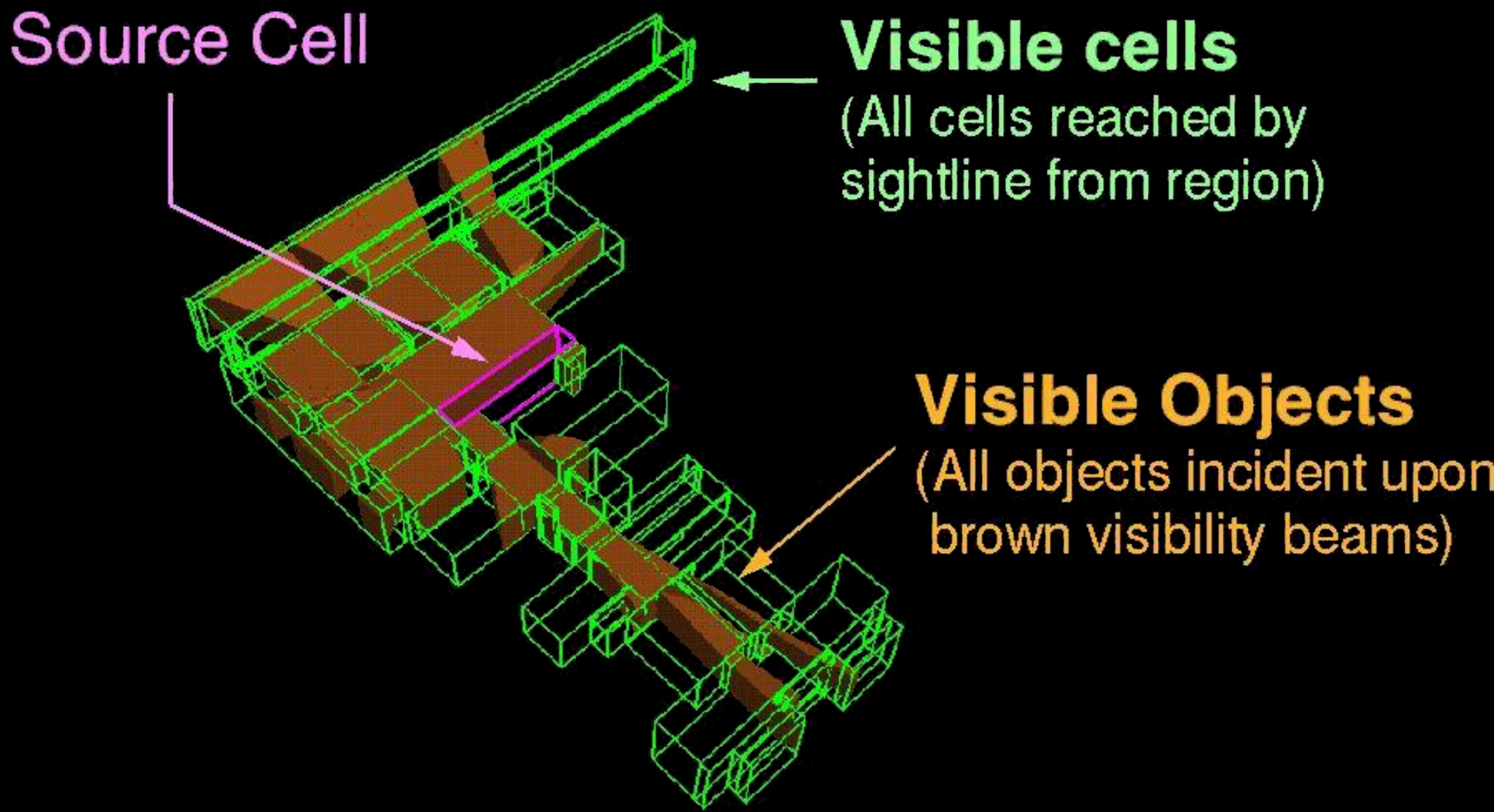
Subdivide space
into cells.

Cluster polygons
into objects.

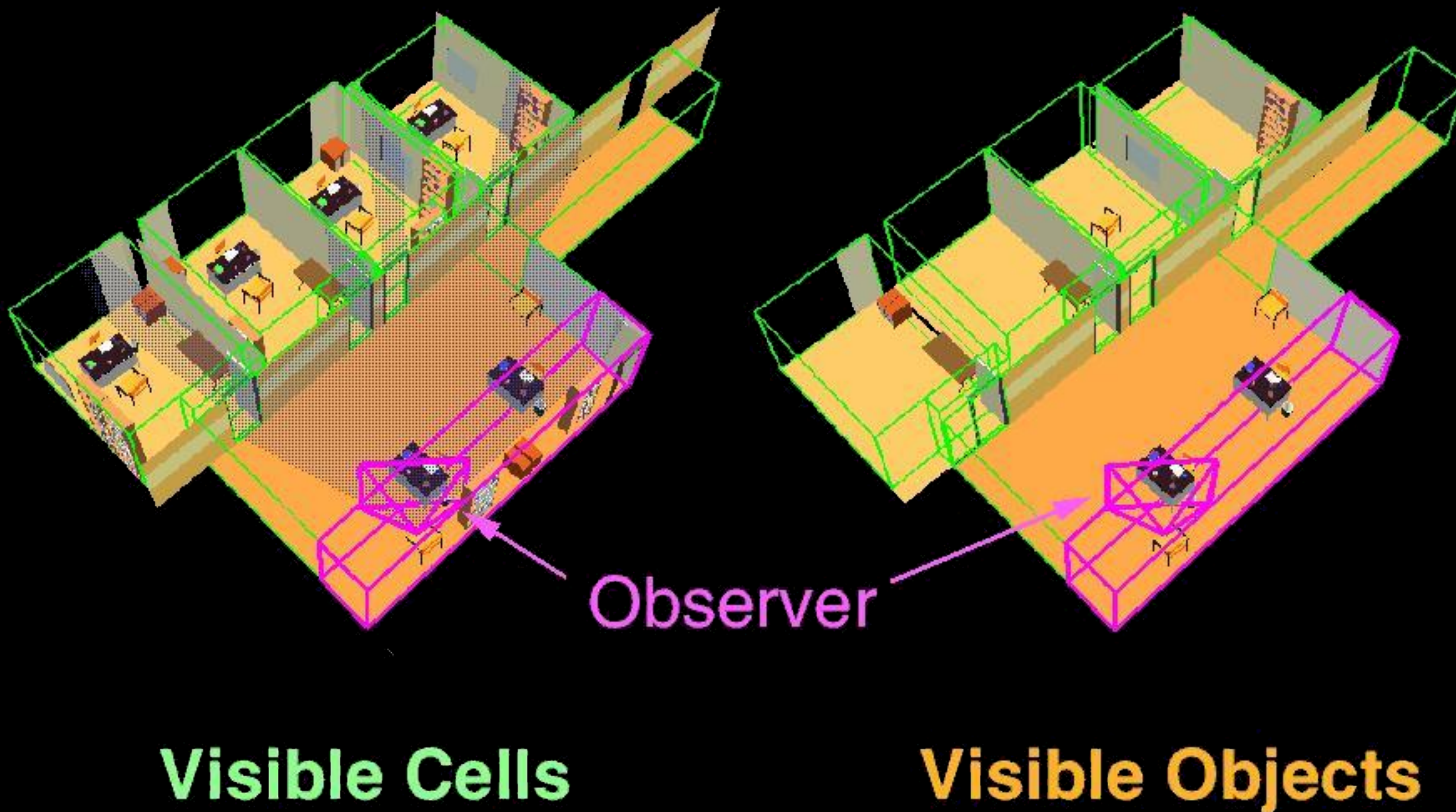
Build an index of objects
incident upon each cell.



Cells and Portals



Cells and Portals



Occlusion Culling

Object-precision

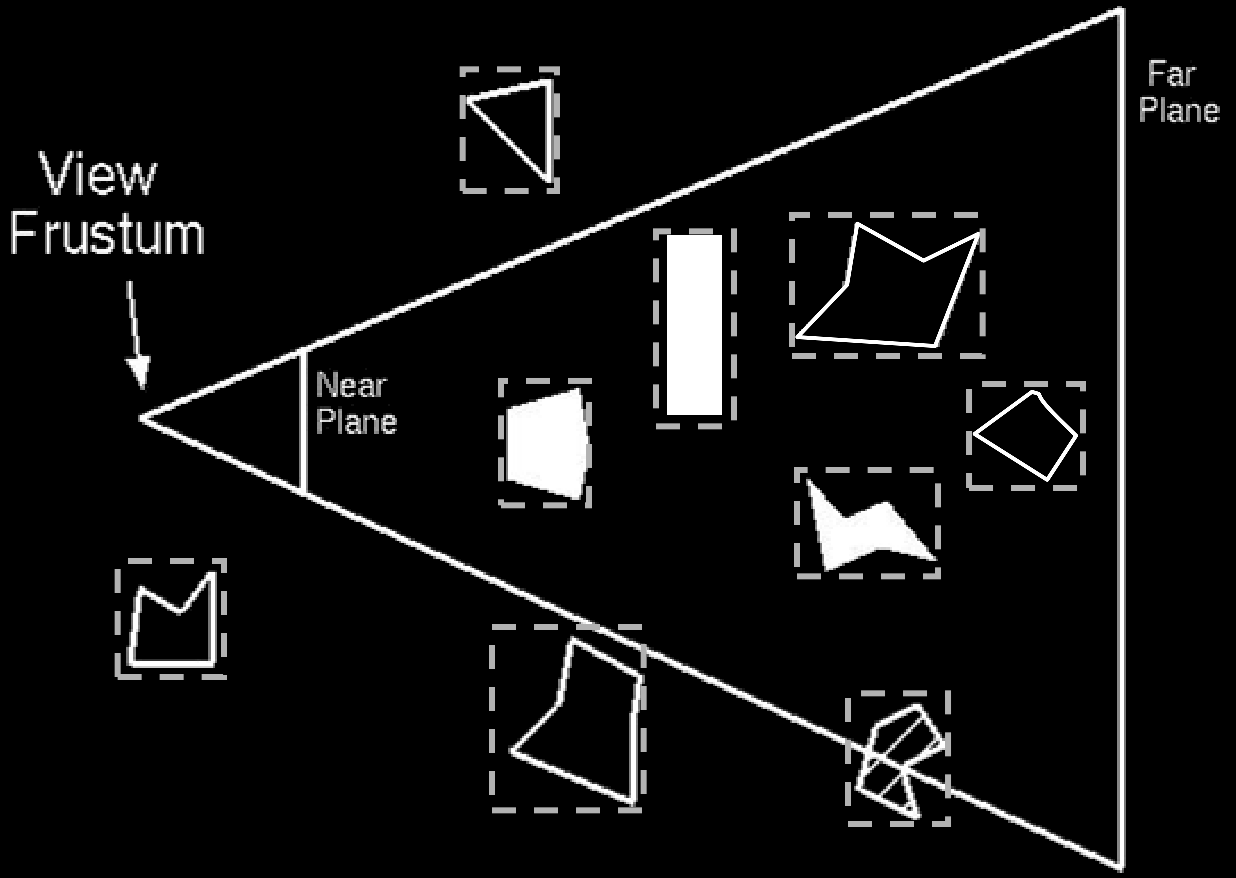
- Cells and portals
- Shadow volumes

Image-precision

- OpenGL occlusion test
- Hierarchical Z-buffer
- Hierarchical occlusion maps

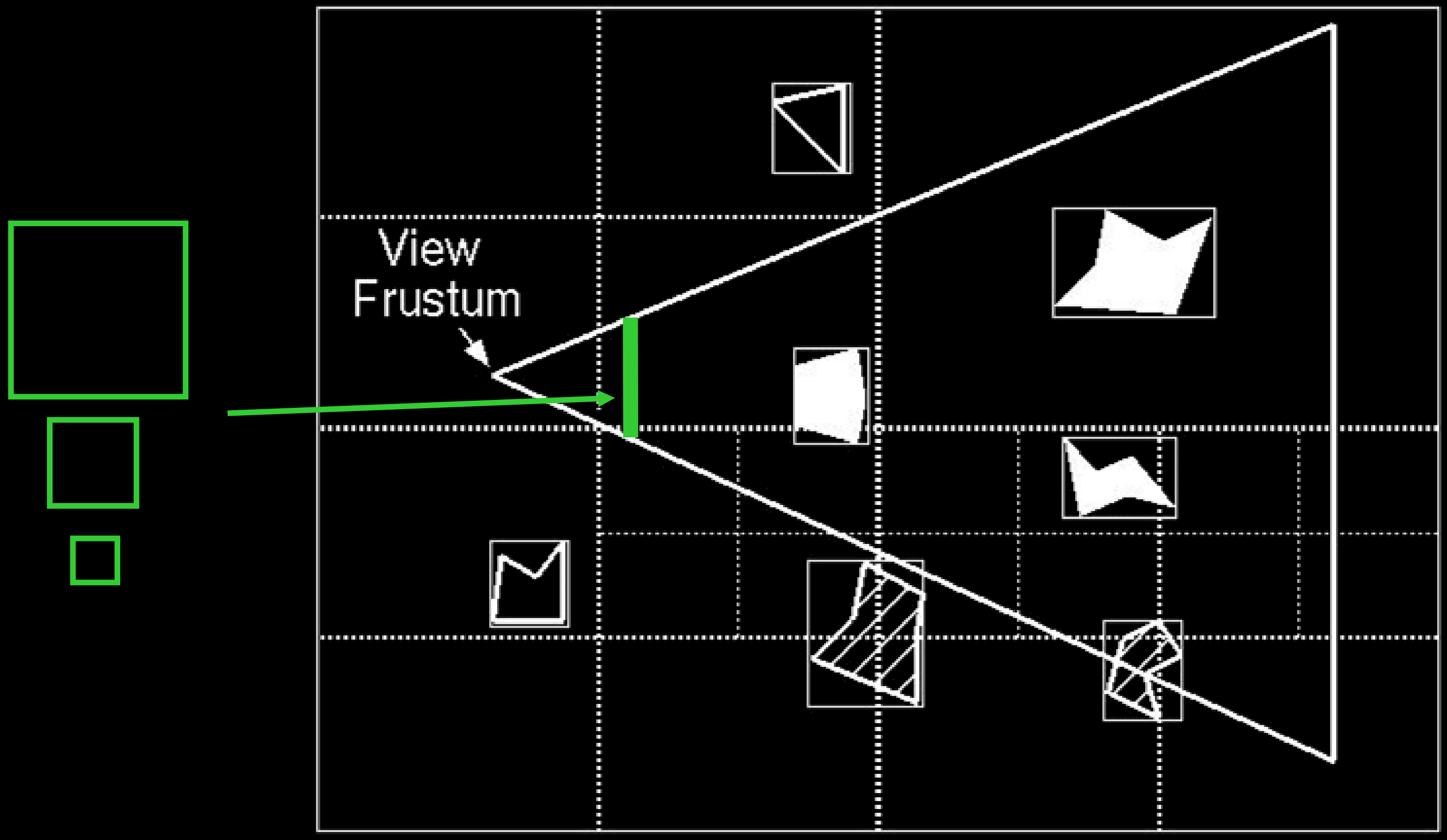
OpenGL Occlusion Test

Hardware returns how many z-buffer tests pass



Hierarchical Z-Buffer

Store z-buffer as pyramid and test depth hierarchically



Rendering Acceleration Techniques

Visibility Culling

- Backface culling, view-frustum culling, occlusion culling, …

Detail Elision

- Levels of detail, multiresolution, …

Images

- Textures, billboards, imposters, …

Levels of Detail

Triangles
: 41,855
27,970
20,922
12,939
8,385
4,766

courtesy of Division and Viewpoint

Levels of Detail

Pre-process

- Generate discrete set of independent levels of detail

Run-time

- Select level of detail according to viewpoint

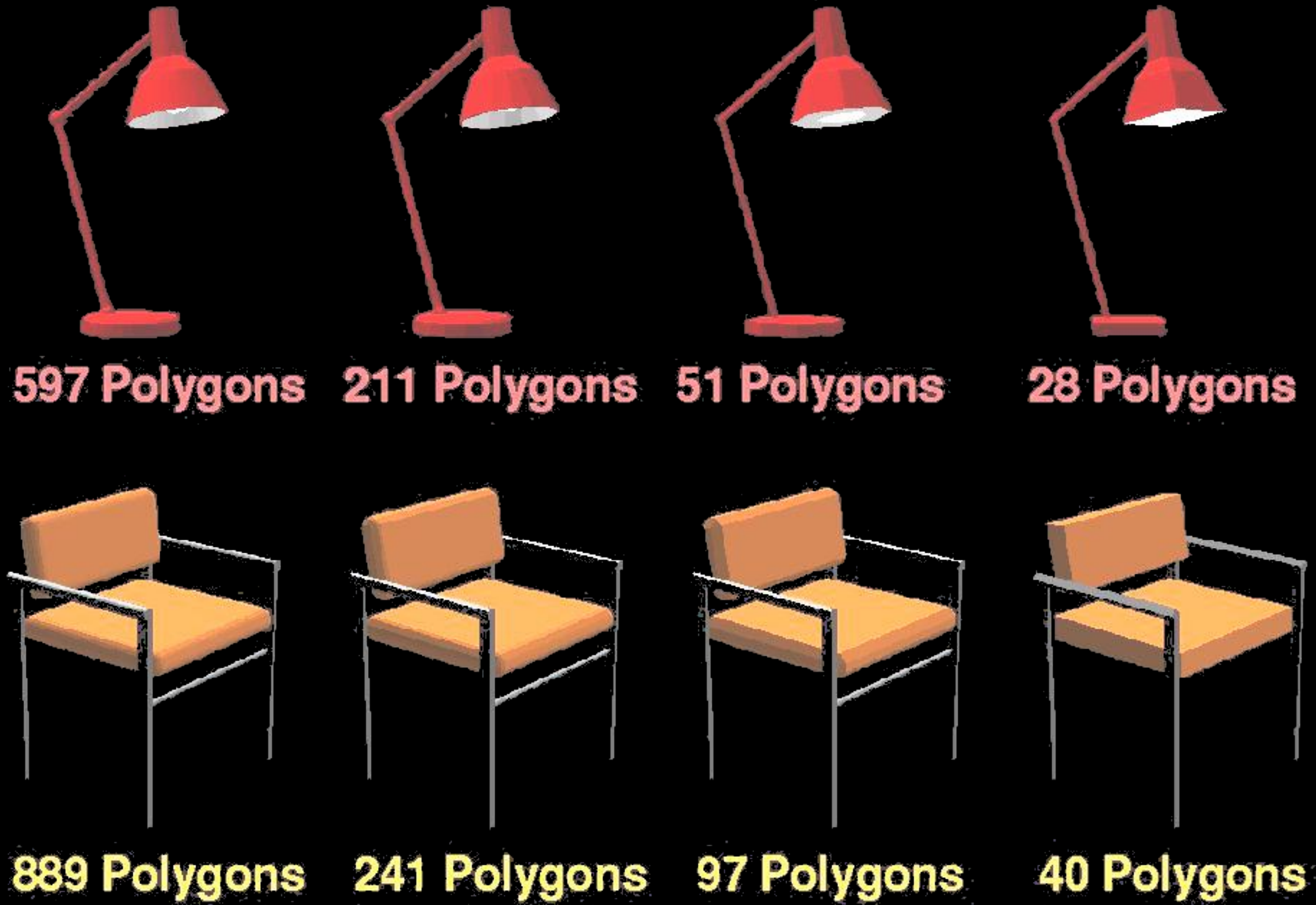
Advantages

- Fairly efficient storage (2x original)
- No significant run-time overhead

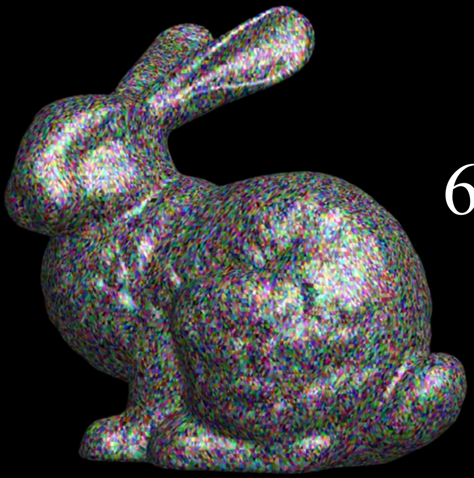
Disadvantages

- Requires per-object simplification
- Not good for spatially large objects

Levels of Detail



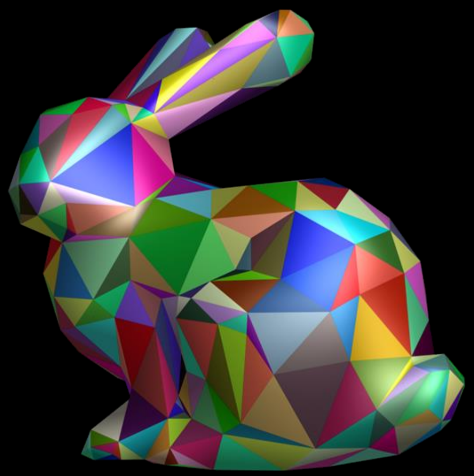
Levels of Detail



69 k tris

11 k tris

2 k tris

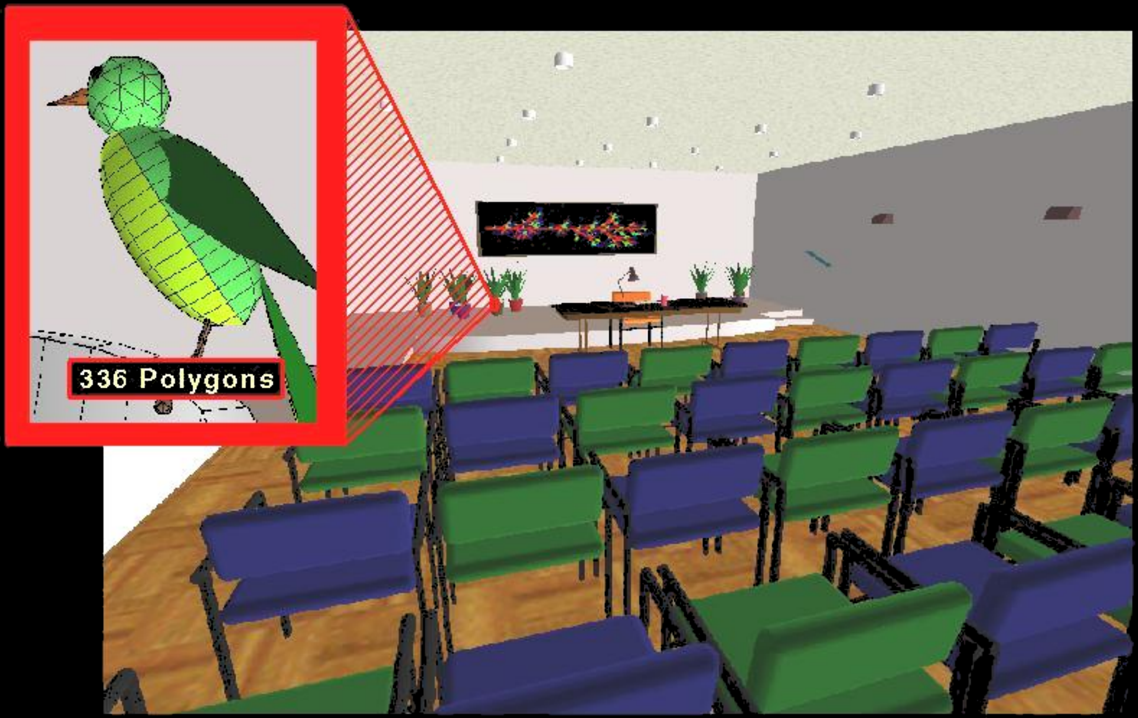


575 tris

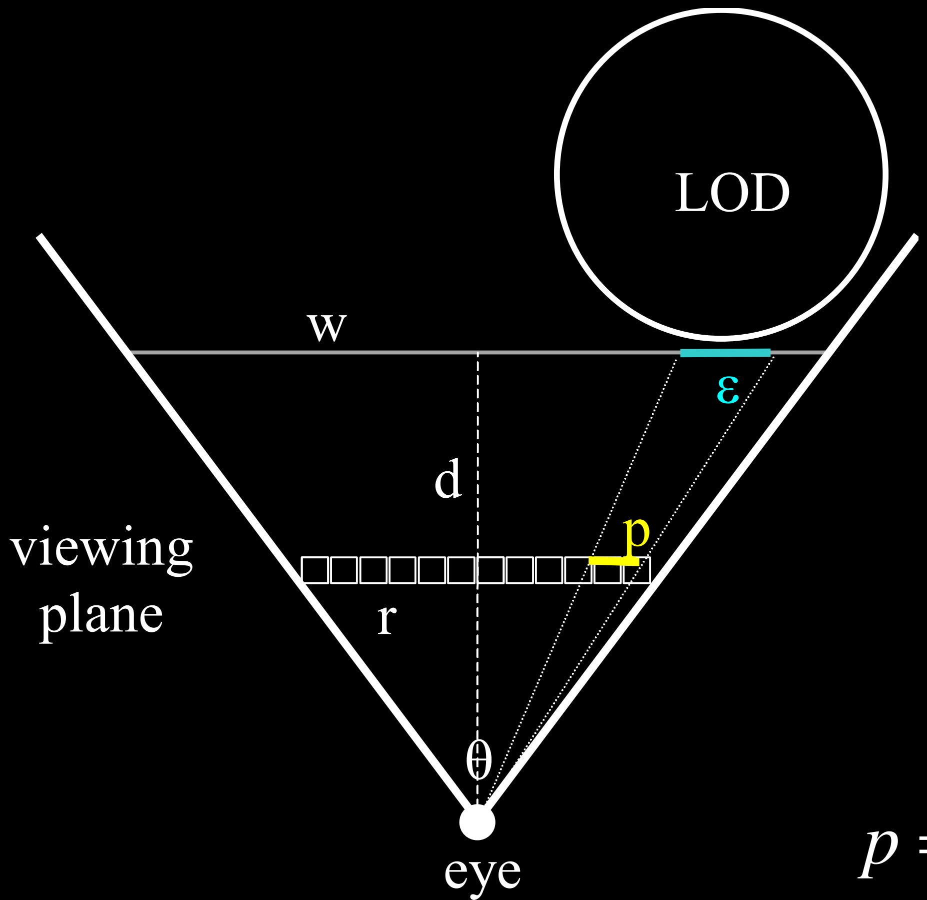
Selecting Levels of Detail

Two possibilities:

- Guarantee quality, maximize frame rate
- Guarantee frame rate, maximize quality

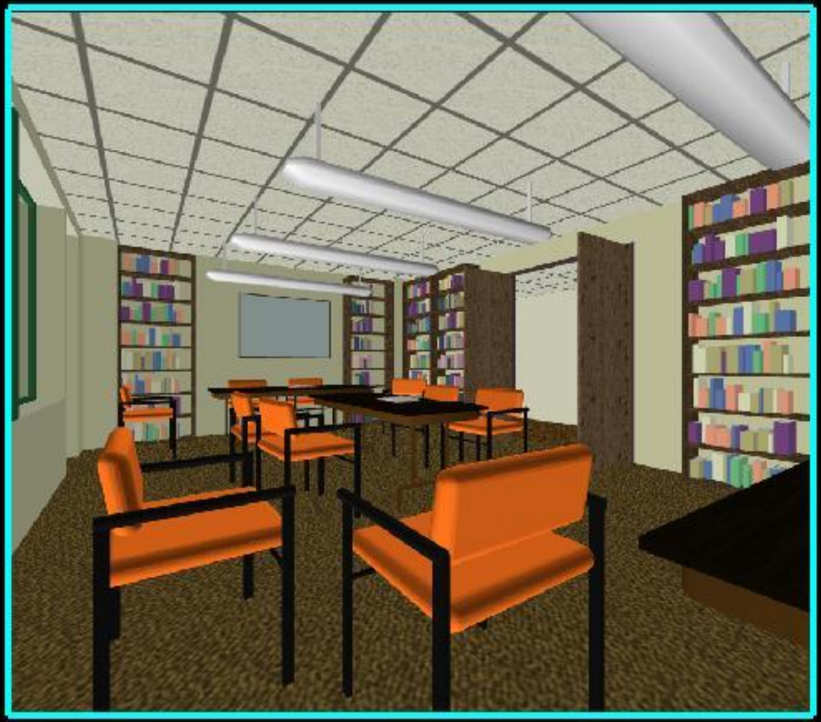


Selecting Levels of Detail



$$p = \frac{\varepsilon r}{w} = \frac{\varepsilon r}{2d\tan(\theta/2)}$$

Guaranteeing Frame Rate

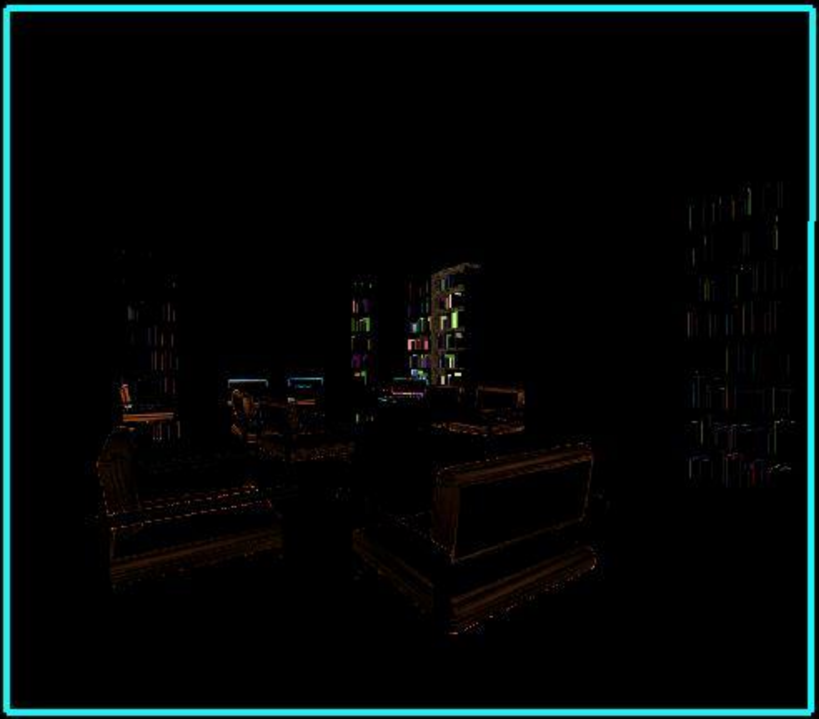


No Detail Elision
0.22 Seconds
(19,881 Polygons)

Optimization Detail Elision
0.05 Seconds
(3,568 Polygons)

Guaranteeing Frame Rate

Objects Shaded by LOD
(Higher LODs appear darker)



Pixel–by–Pixel Differences
(Larger differences appear brighter)

Multiresolution Meshes

Pre-process

- Generate tree of simplification operations

Run-time

- Refine/coarsen current model according to viewpoint

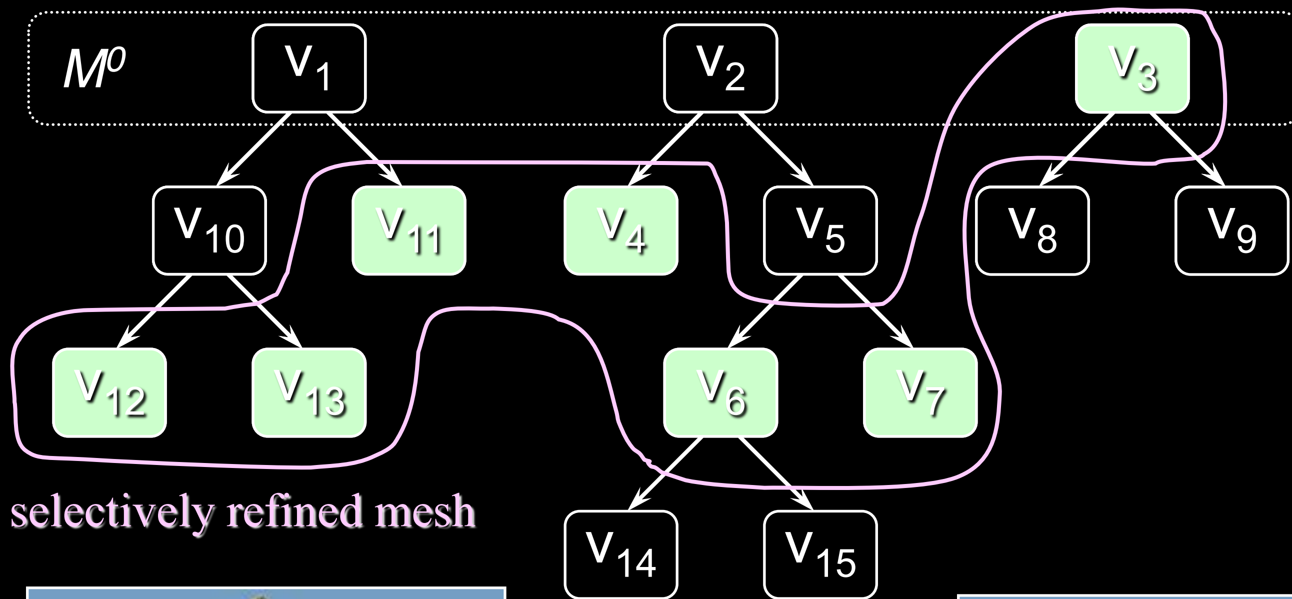
Advantages

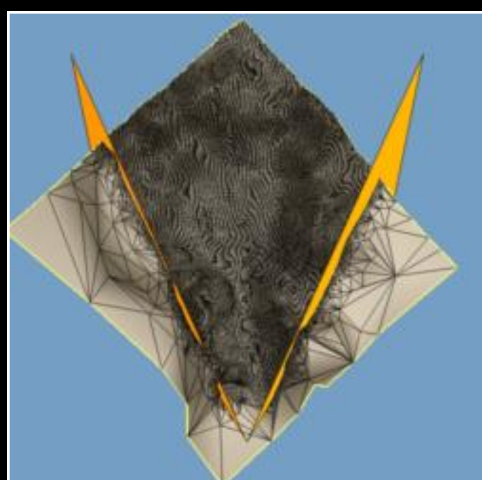
- Allows finer control of tessellation

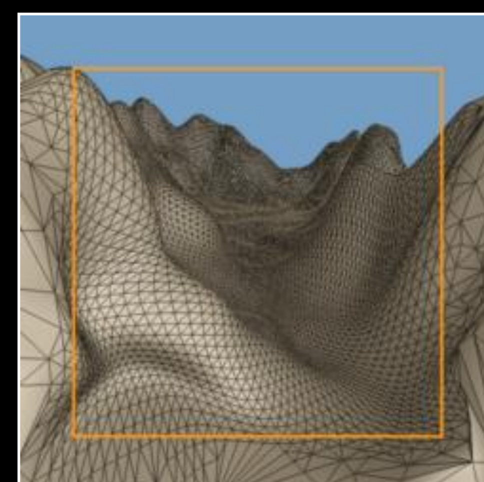
Disadvantages

- More run-time computation and complexity
- Difficult for retained-mode graphics

Multiresolution Meshes







Rendering Acceleration Techniques

Visibility Culling

- Backface culling, view-frustum culling, occlusion culling, …

Detail Elision

- Levels of detail, multiresolution, …

Images

- Textures, billboards, imposters, …

Imposters

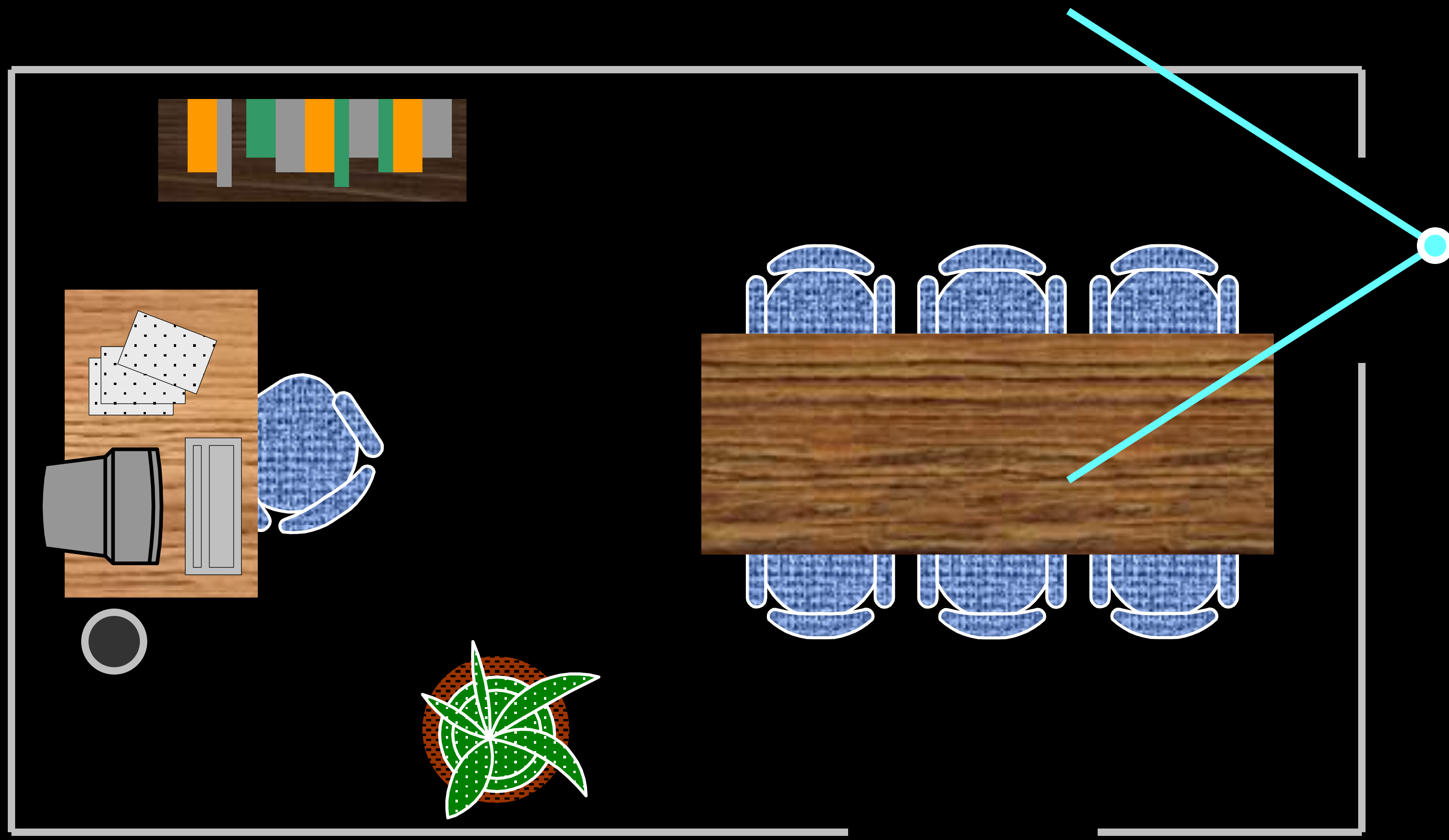
Algorithm

- Select subset of model
- Create image of the subset
- Cull subset and replace with image

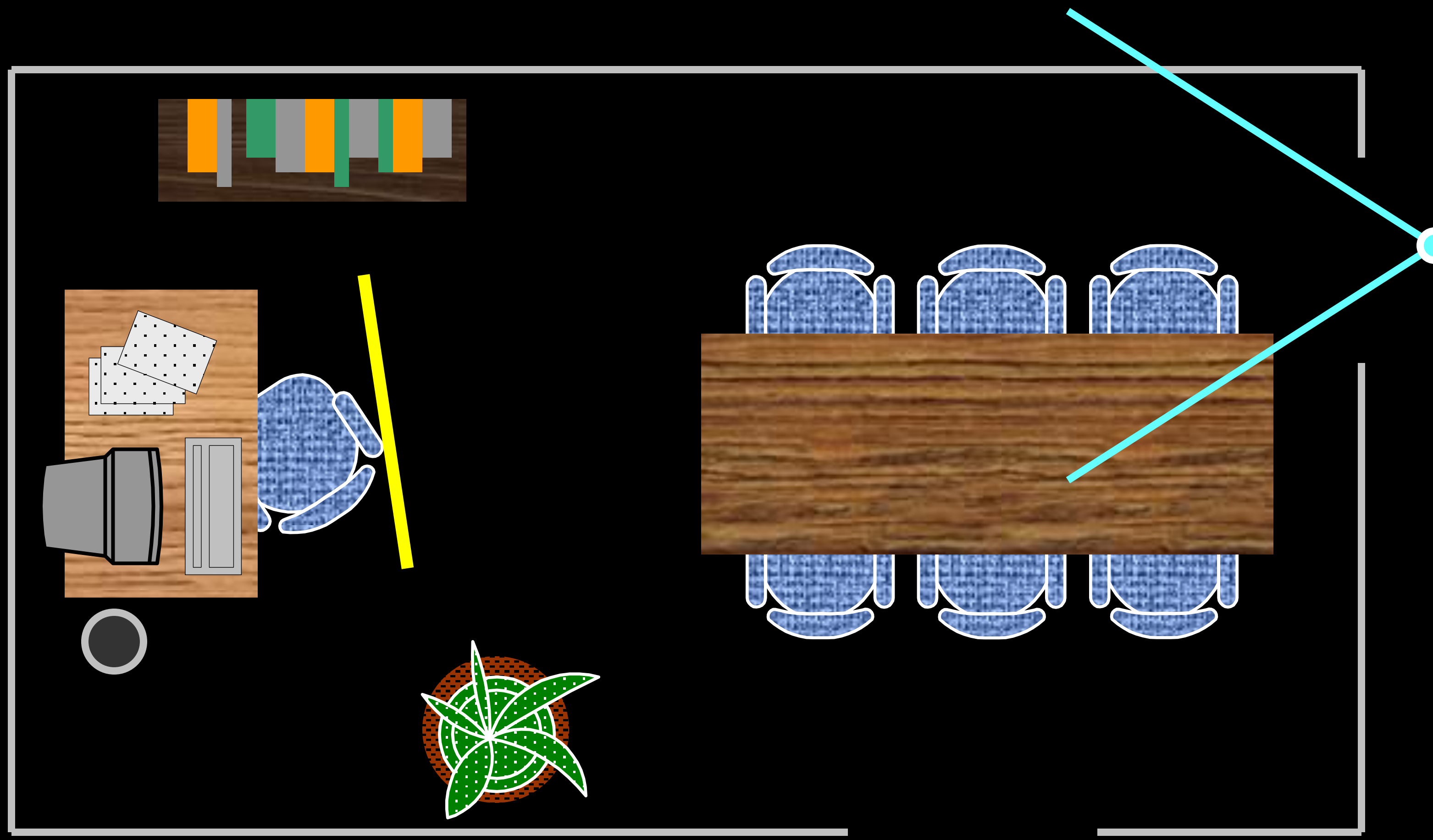
Why?

- Image displayed in (approx.) constant time
- Image reused for several frames

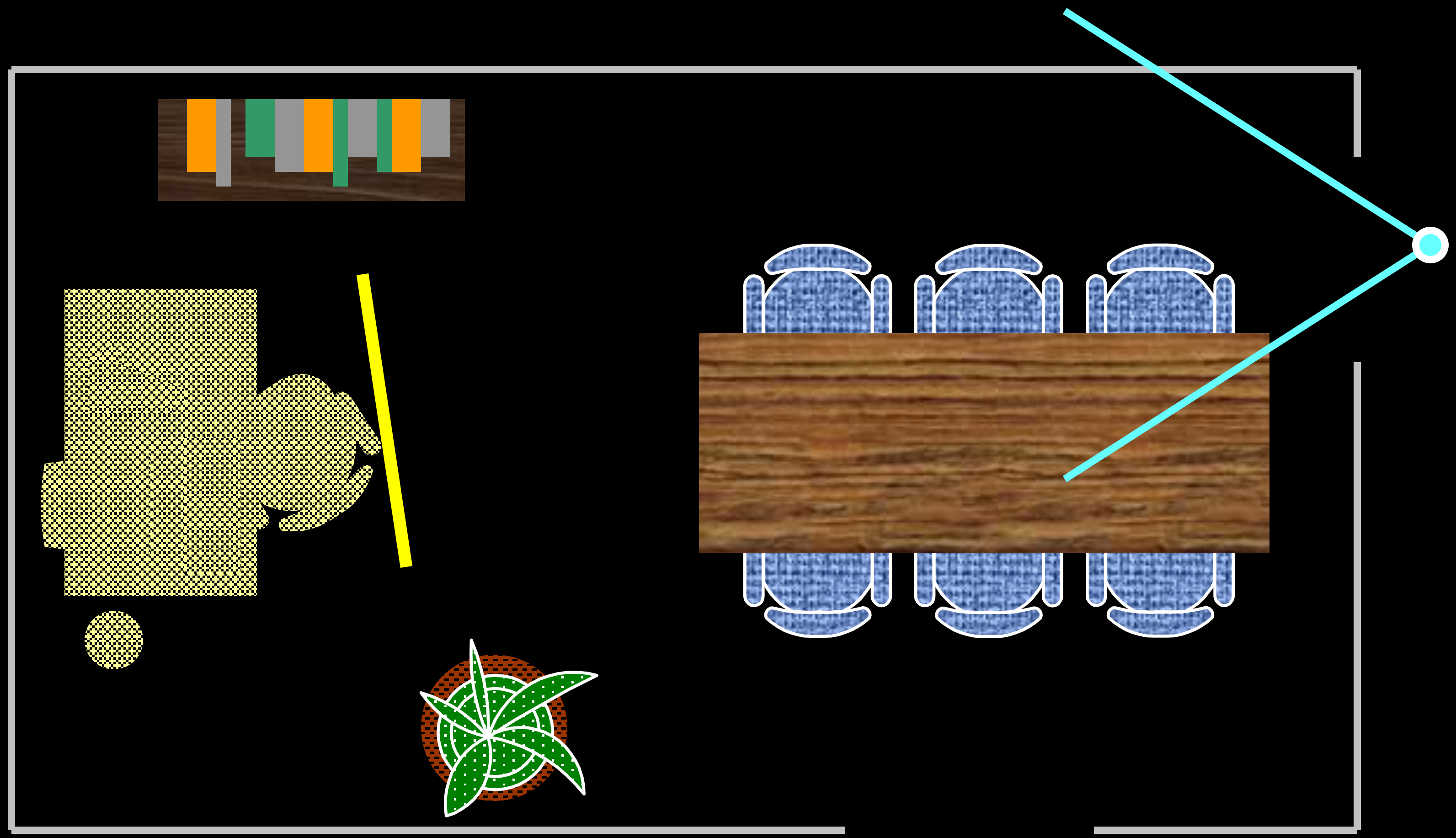
Simple Example



Simple Example



Simple Example



Issues

Imposter placement

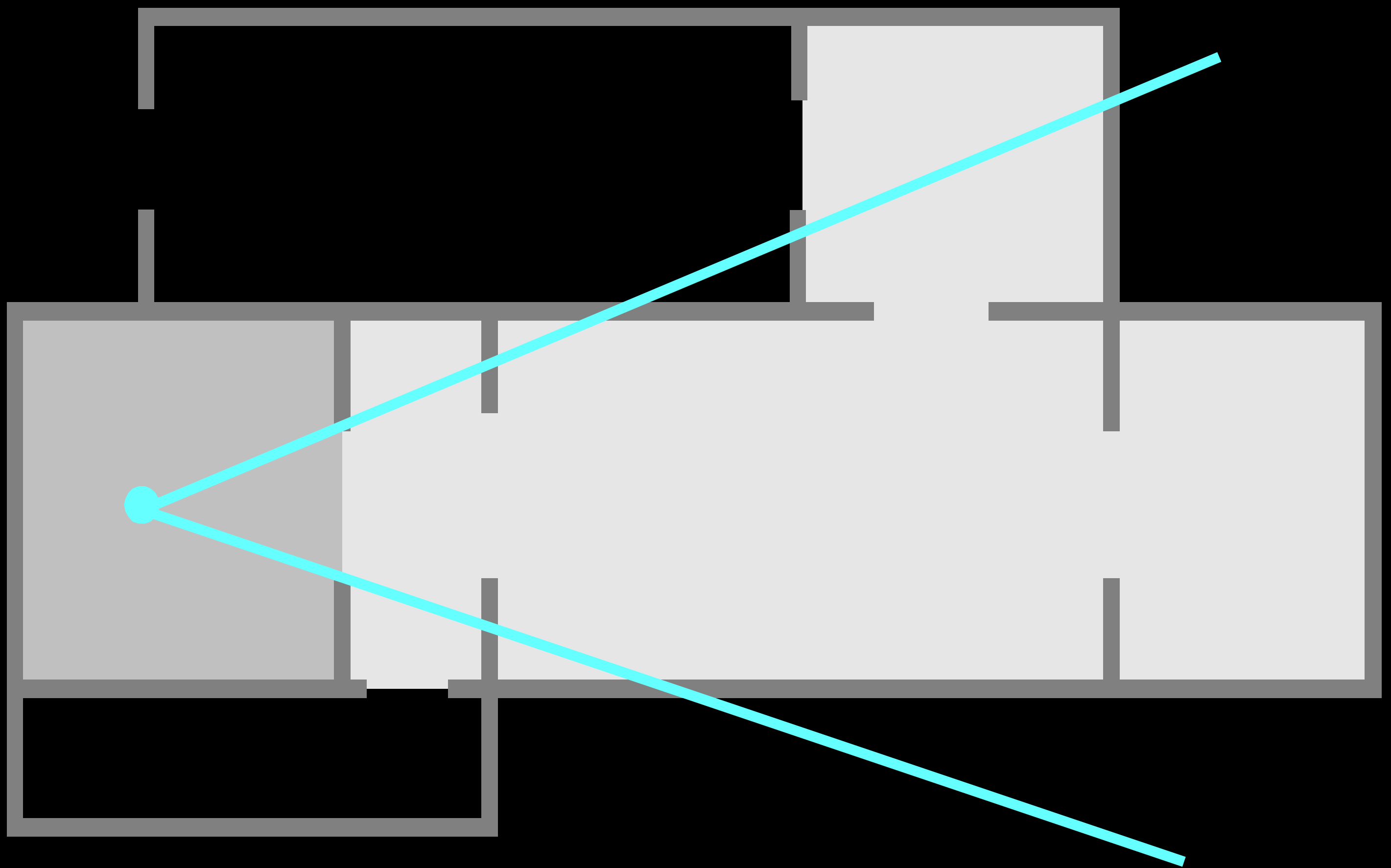
- What geometry should be replaced by images
- How should images be integrated into scene

Imposter representation

- What viewpoint(s) should be captured in image?
- How render from arbitrary viewpoints?

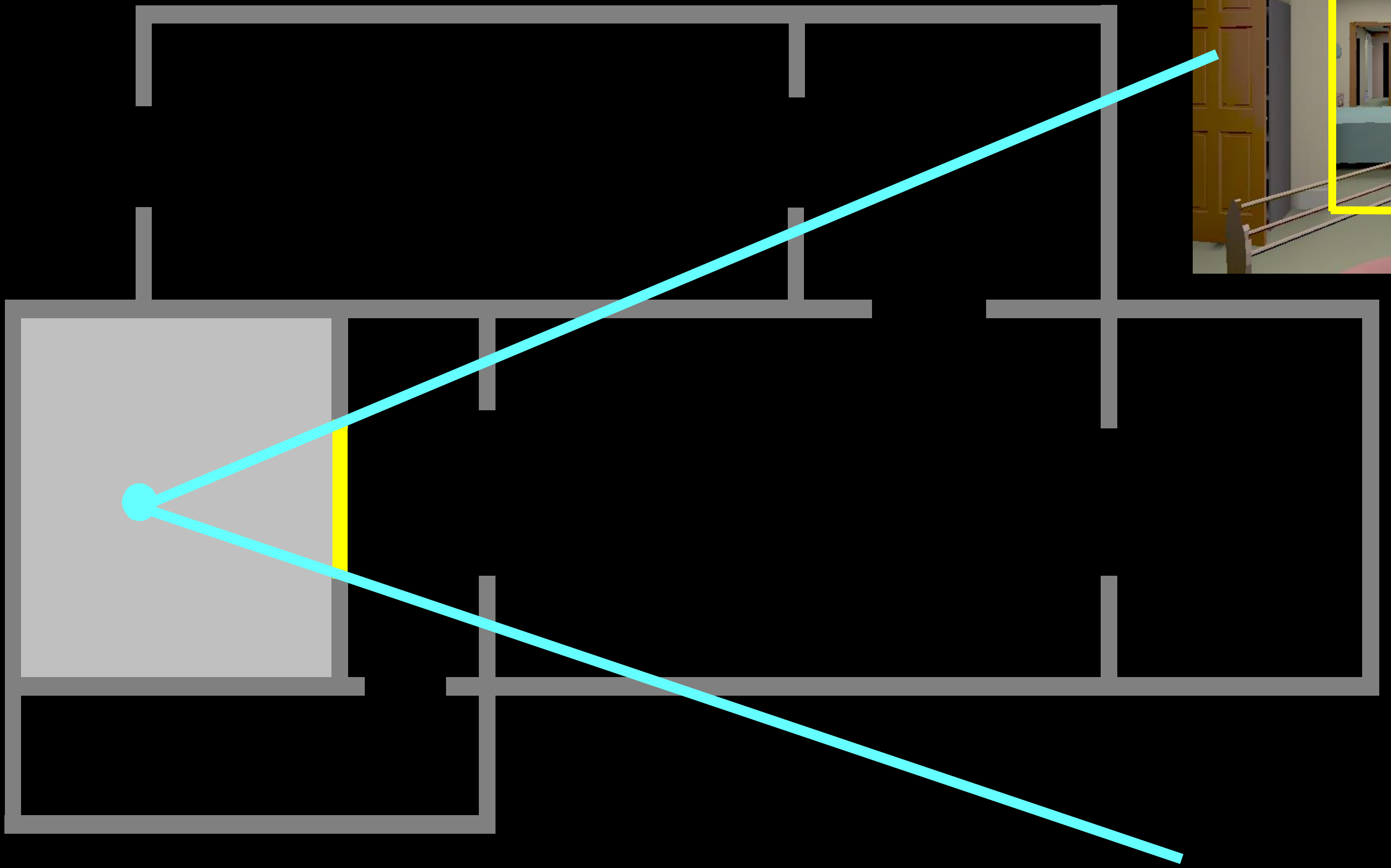
Imposter Placement

Cells and Portals



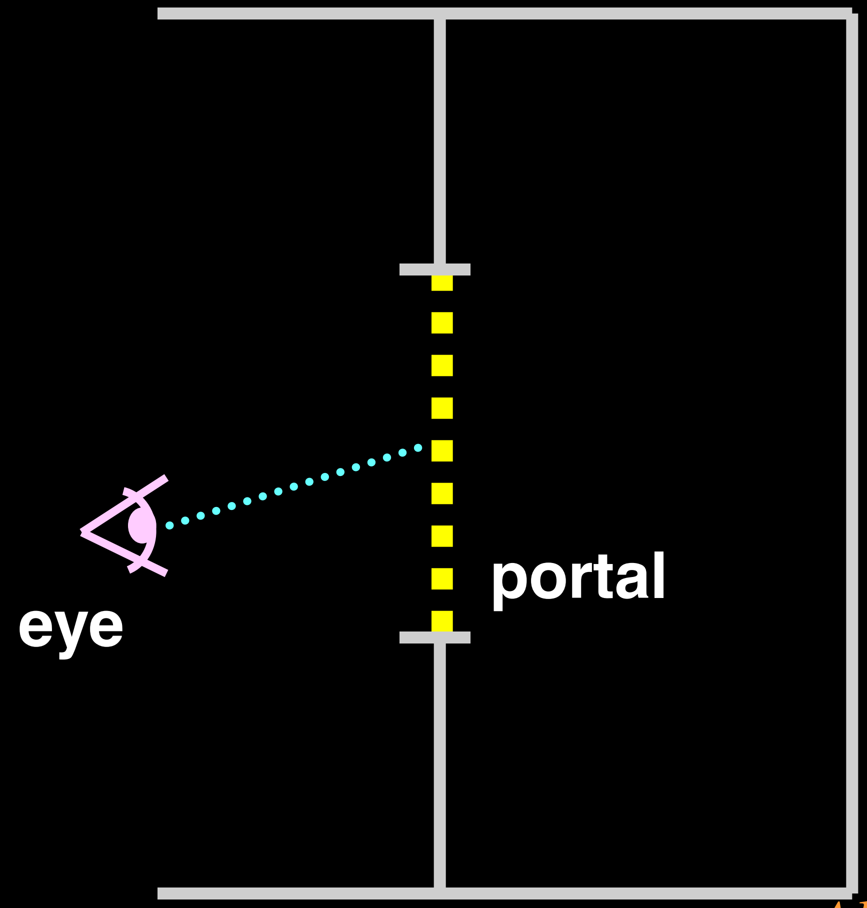
Aliaga

Portal Images



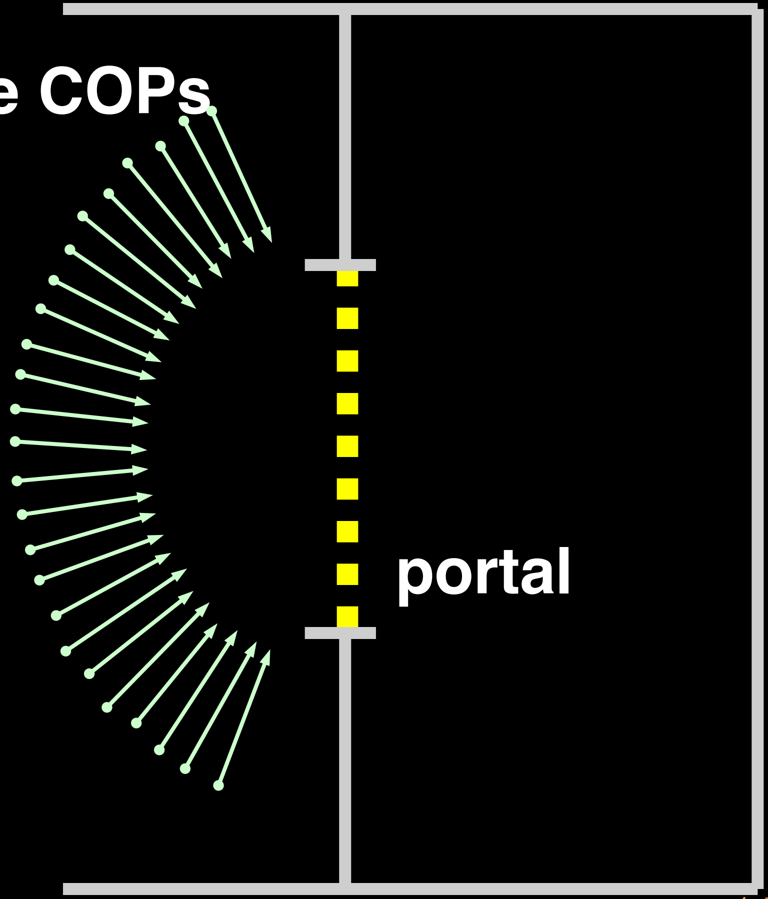
Creating Portal Images

Ideal portal image would be one sampled from the current eye position



Creating Portal Images

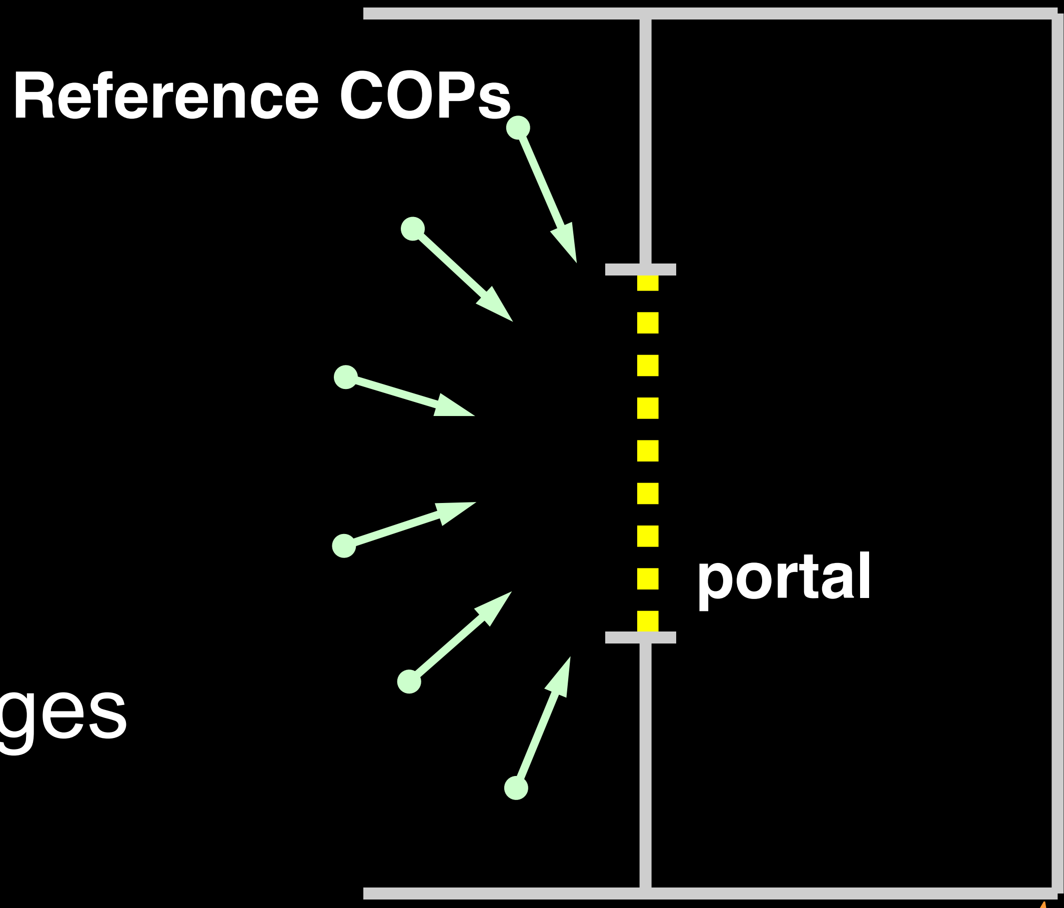
Display one of a large number of pre-computed images (~120)



Creating Portal Images

or…

Warp one of a much smaller number of reference images



Summary

Visibility Culling

- Backface culling, view-frustum culling, occlusion culling, …

Detail Elision

- Levels of detail, multiresolution, …

Images

- Textures, billboards, imposters, …

Recurring Themes:

Trivial reject checks

Hierarchical processing