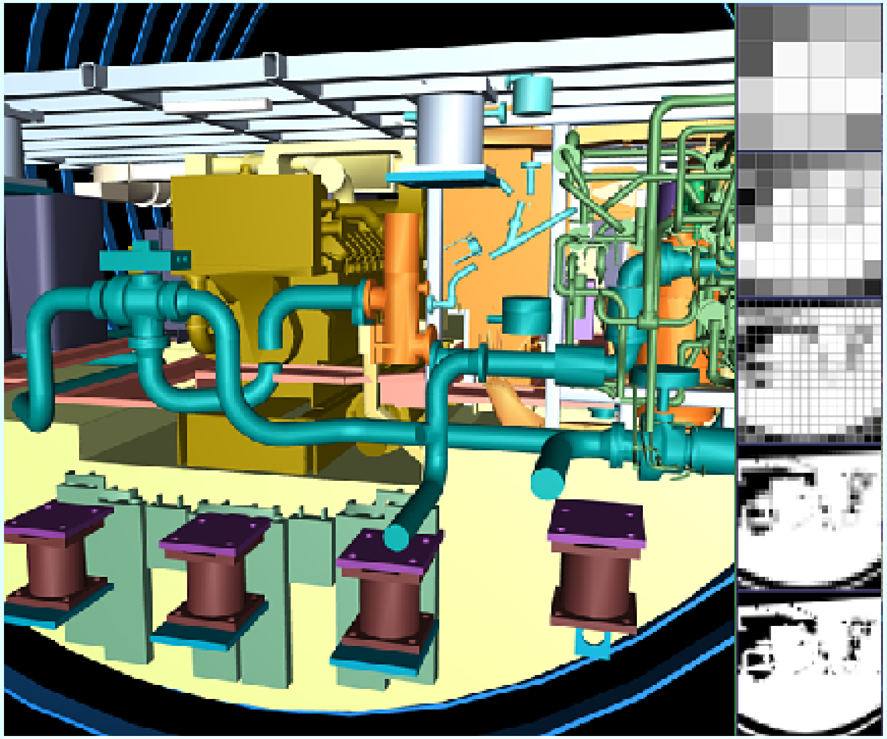
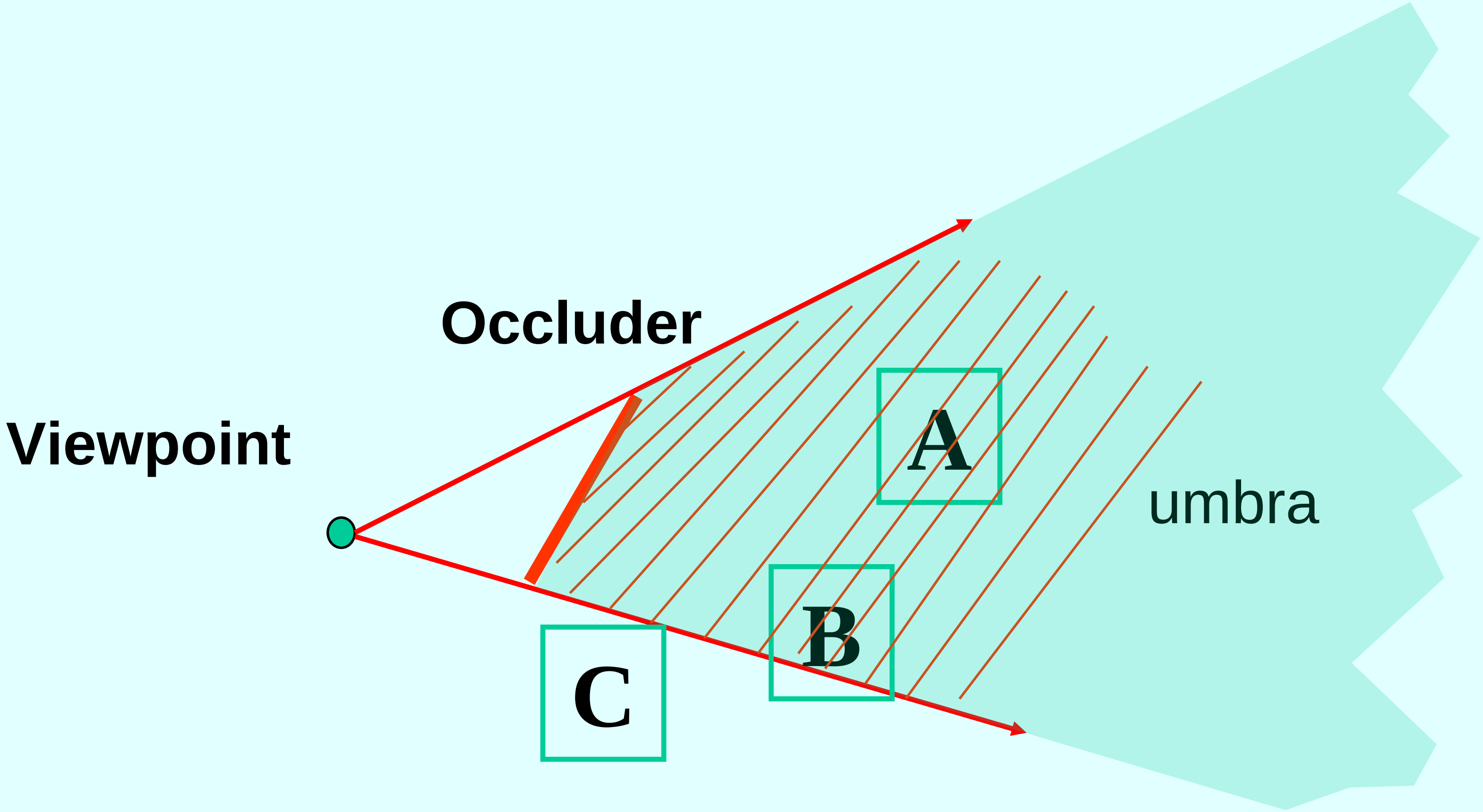
Image Space Occlusion Culling



Hudson et al, SoCG 97

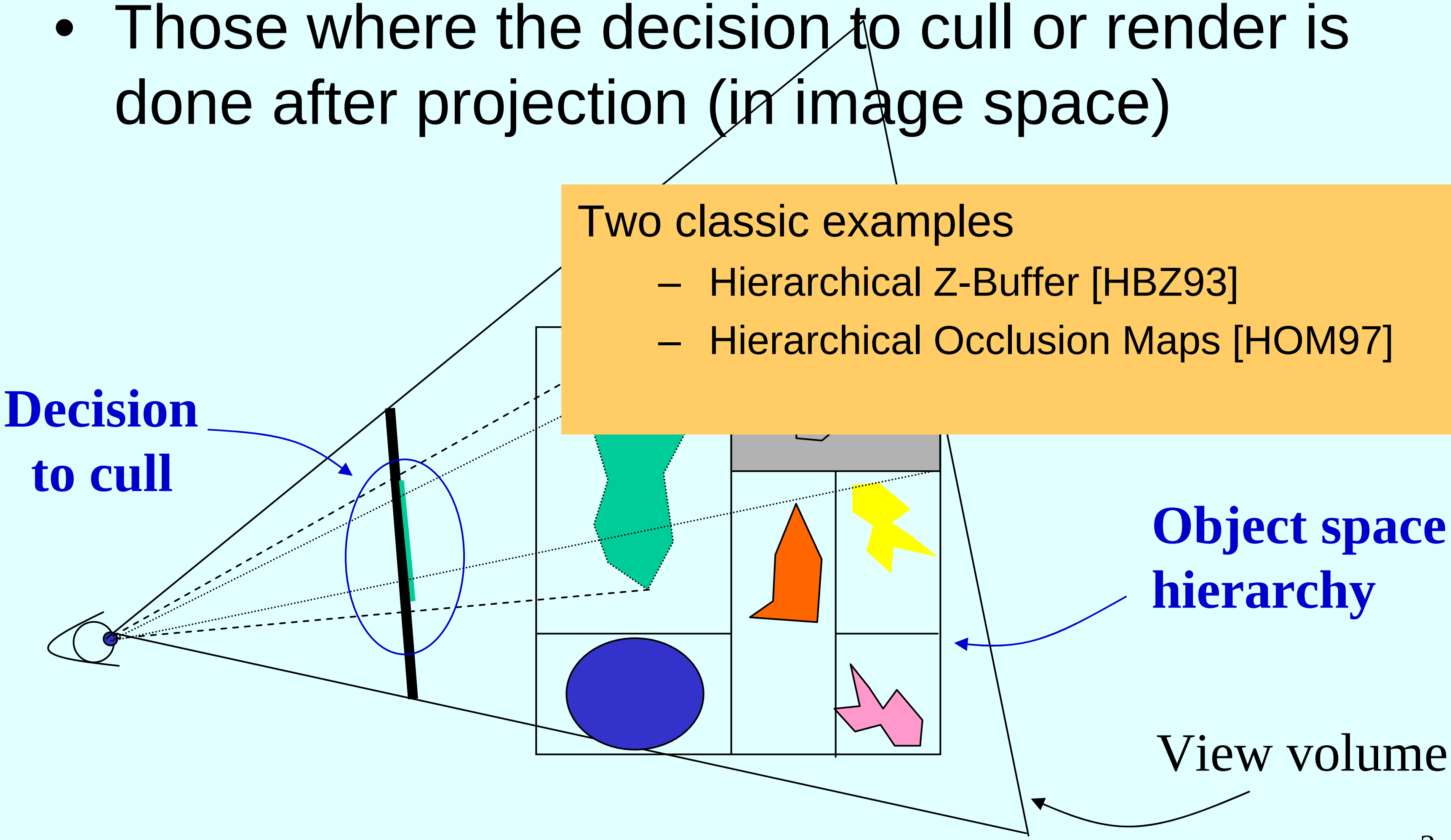


What Methods are Called Image-Space?

- Those where the decision to cull or render is done after projection (in image space)

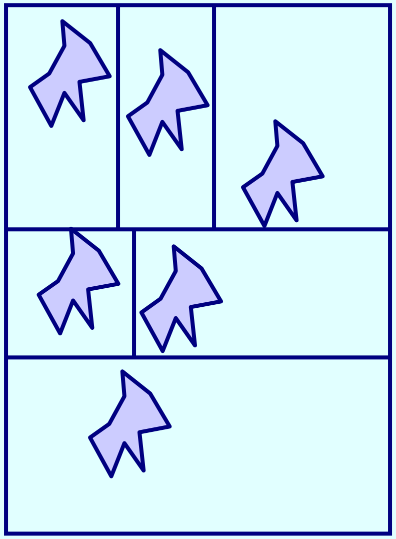
Two classic examples

- Hierarchical Z-Buffer [HBZ93]
- Hierarchical Occlusion Maps [HOM97]

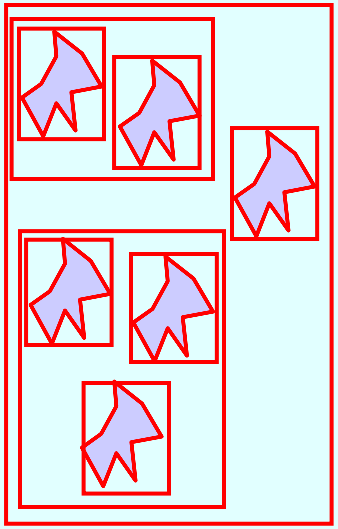


Ingredients of an Image Space Method

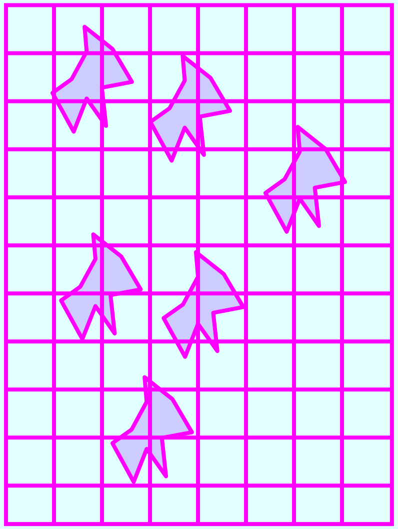
- An object space data structure that allows fast queries to the complex geometry



Space partitioning



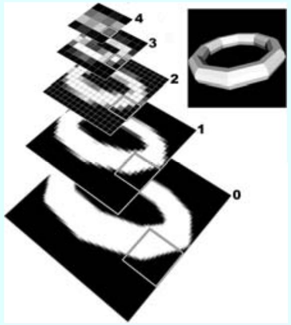
Hierarchical bounding boxes

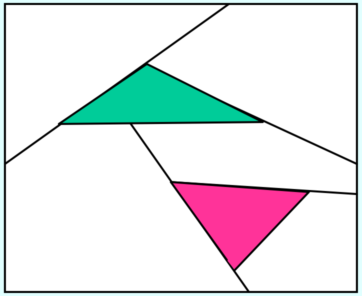


Regular grid

An image space representation of the occlusion information

- Discrete
  - – Z-hierarchy
  - – Occlusion map hierarchy
- Continuous
  - – BSP tree
  - – Image space extends



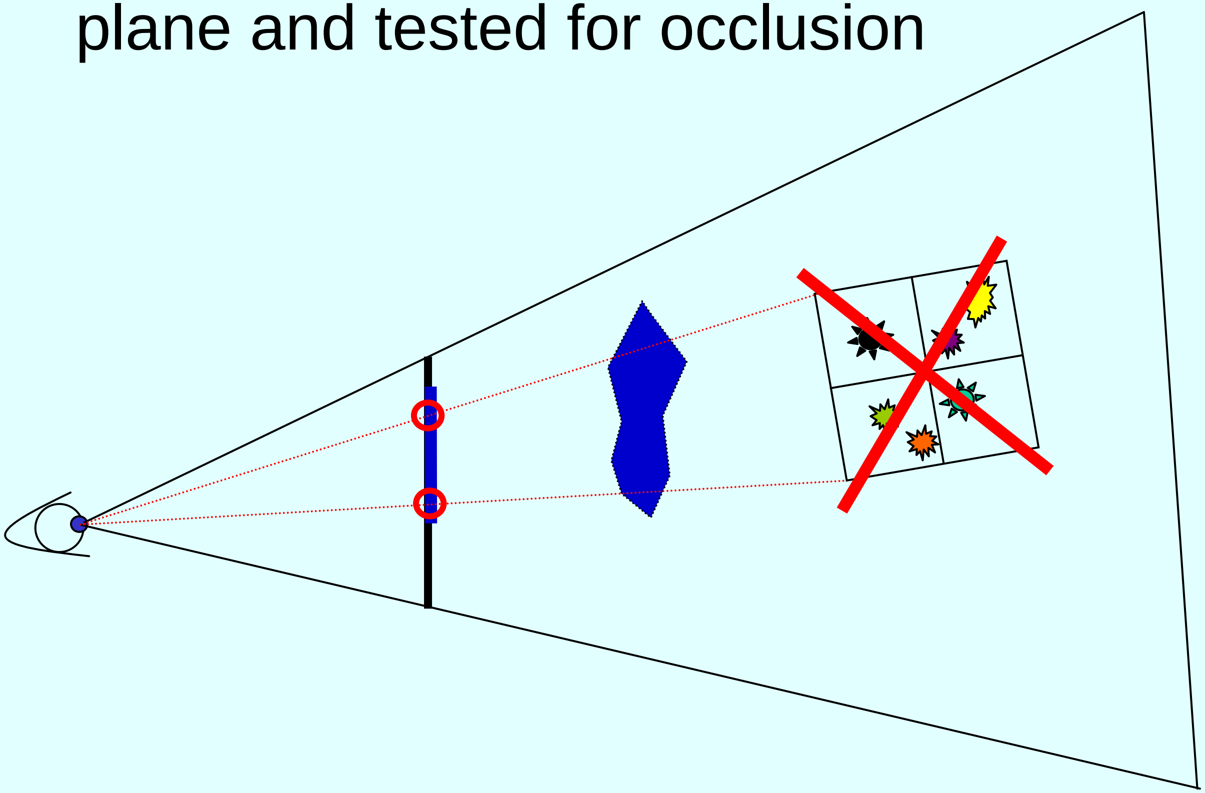


General Outline of Image Space Methods

- During the (front-to-back) traversal of the scene hierarchy do:
  - compare each node against the view volume
  - if not culled, test node for occlusion
  - if still not culled, render objects/occluders **augmenting the image space occlusion**

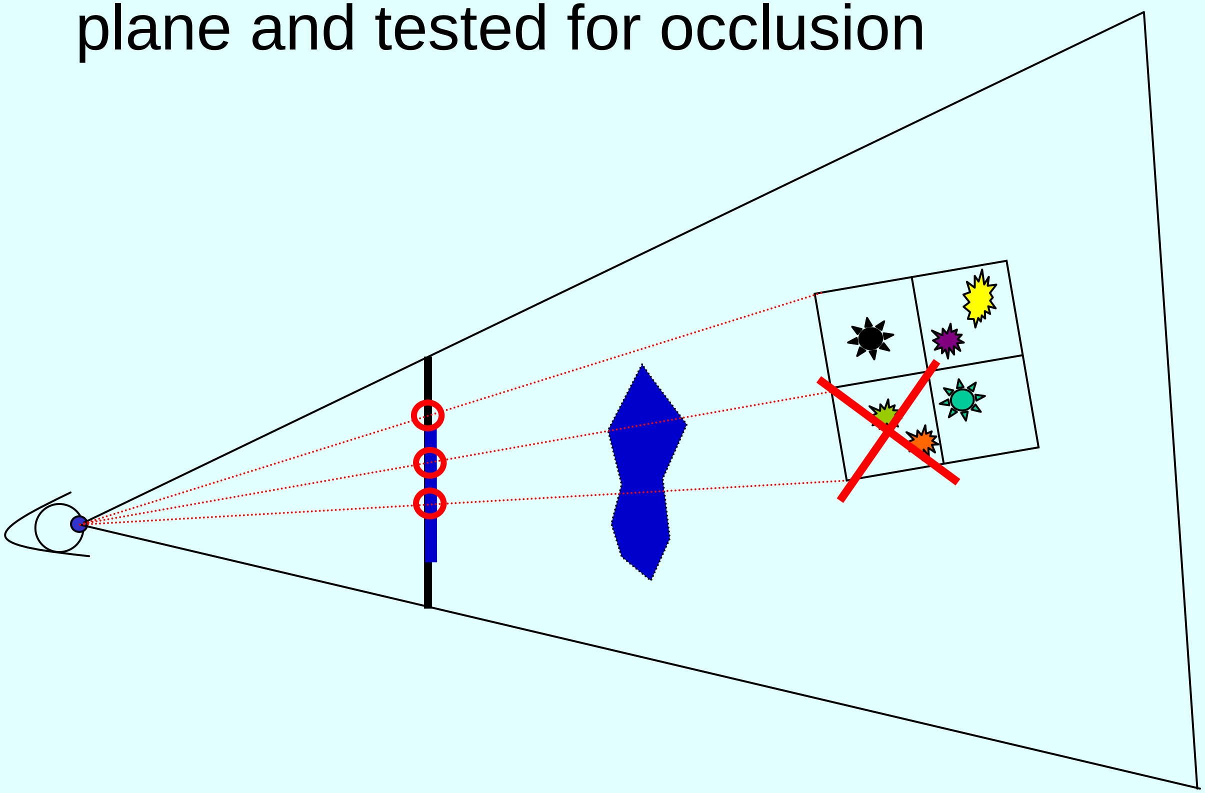
Testing a Node for Occlusion

- If the box representing a node is not visible then nothing in it is either
- The faces of the box are projected onto the image plane and tested for occlusion

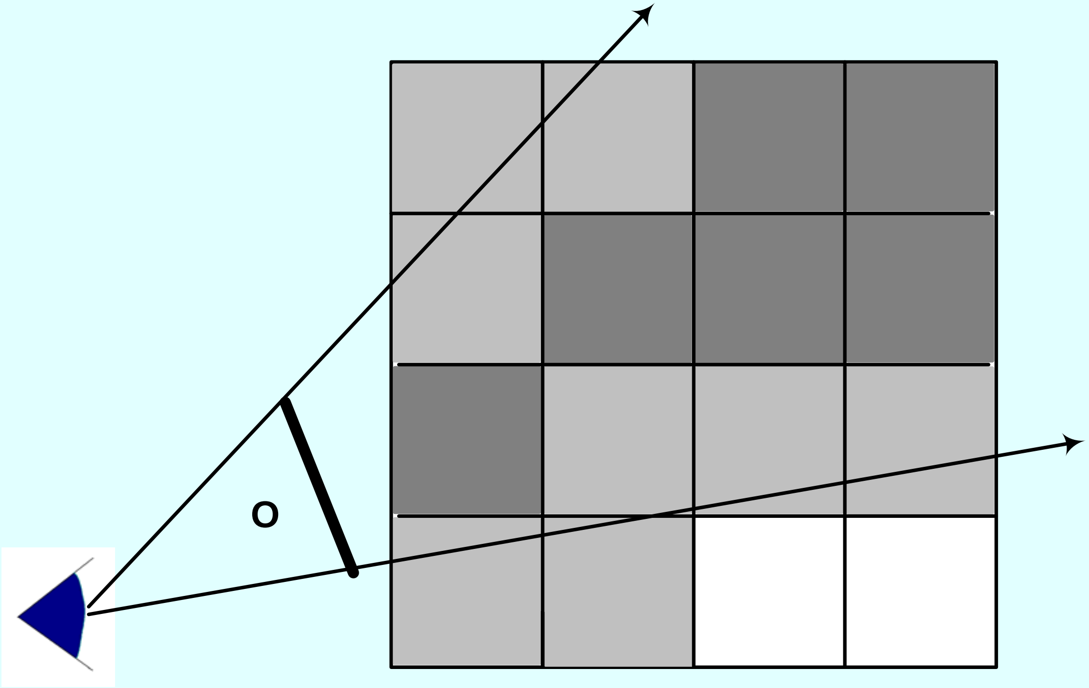


Testing a Node for Occlusion

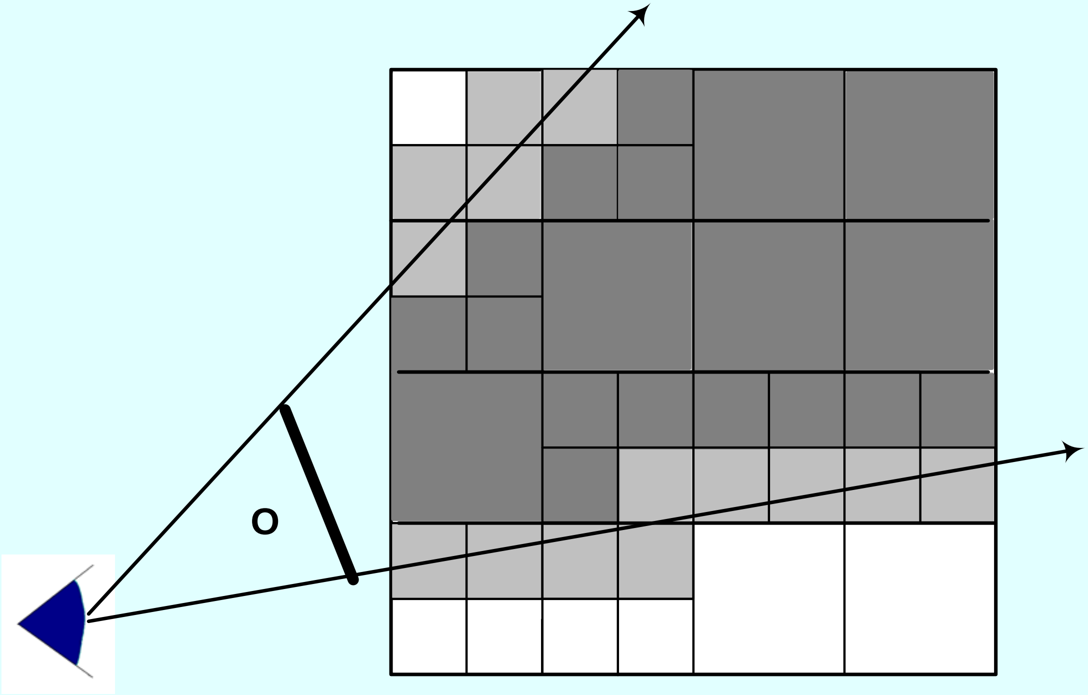
- If the box representing a node is not visible then nothing in it is either
- The faces of the box are projected onto the image plane and tested for occlusion



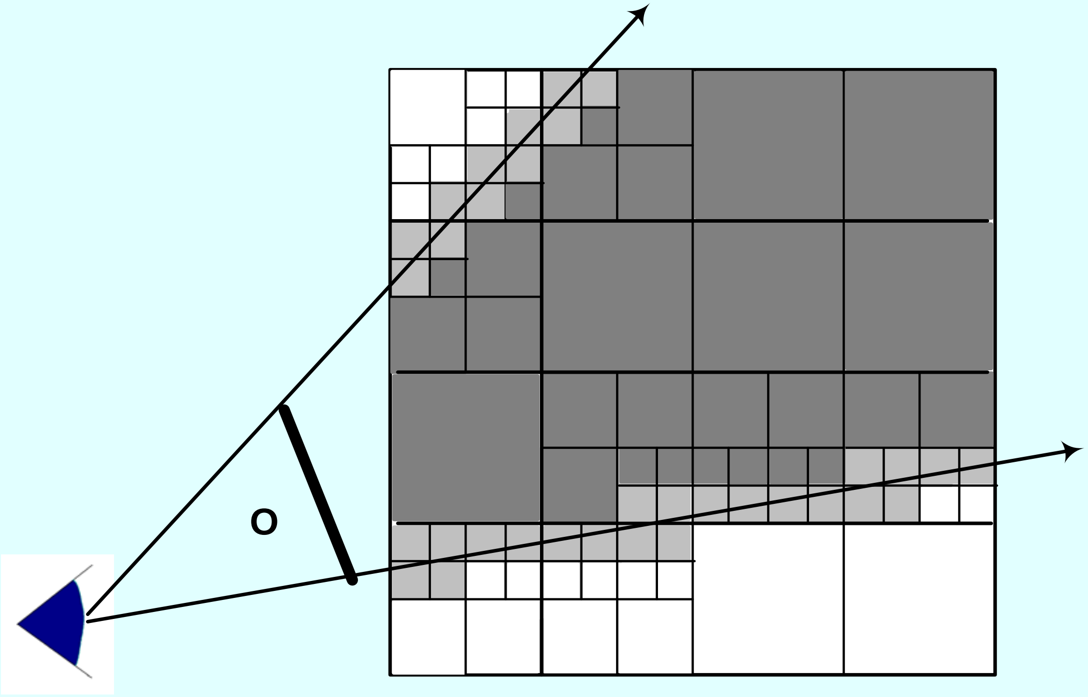
Hierarchical Tests



Hierarchical Tests



Hierarchical Tests



Differences of Algorithms

- The most important differences between the various approaches are:
  - the representation of the (augmented) occlusion in image space and,
  - the method of testing the hierarchy for occlusion

Hierarchical Z-Buffer (HZB)

(Ned Greene, Michael Kass 93)

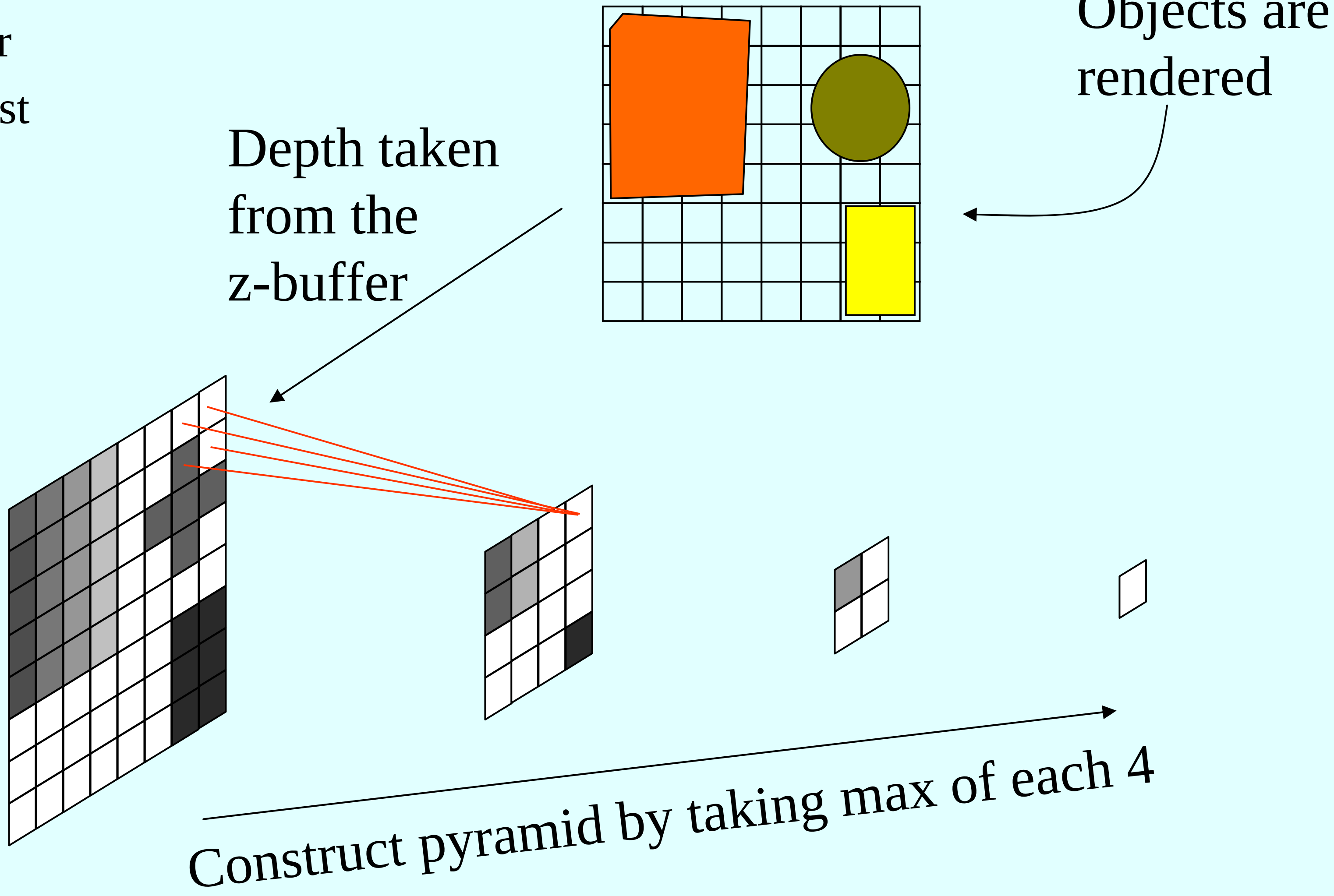
- An extension of the Z-buffer VSD algorithm
- It follows the outline described above.
- Scene is arranged into an octree which is traversed top-to-bottom and front-to-back.
- During rendering an occlusion map is incrementally built.
- Octree nodes are compared against occlusion map.
- The occlusion map is a z-pyramid…

The Z-Pyramid

□ = furthest

▒ = closer

■ = closest



Depth taken from the z-buffer

Objects are rendered

Construct pyramid by taking max of each 4

Maintaining the Z-Pyramid

- Ideally every time an object is rendered causing a change in the Z-buffer, this change is propagated through the pyramid
- However this is not a practical approach

More Realistic Implementation

- Make use of frame-to-frame coherence:
  - at start of each frame render the nodes that were visible in previous frame
  - read the z-buffer and construct the z-pyramid
  - now traverse the octree using the z-pyramid for occlusion but without updating it

Cool idea!

HZB: discussion

- It provides good acceleration in very dense scenes
- Getting the necessary information from the Z-buffer is costly
- A hardware modification was proposed for making it real-time

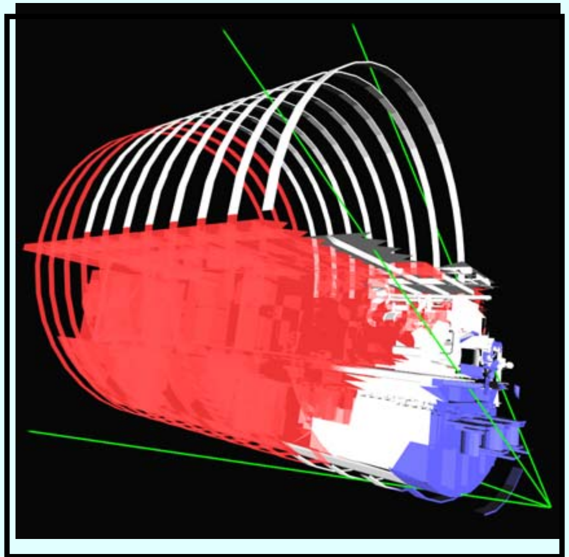
Hierarchical Occlusion Maps
(Hansong Zhang et.al 97)

Similar idea to HZB but:

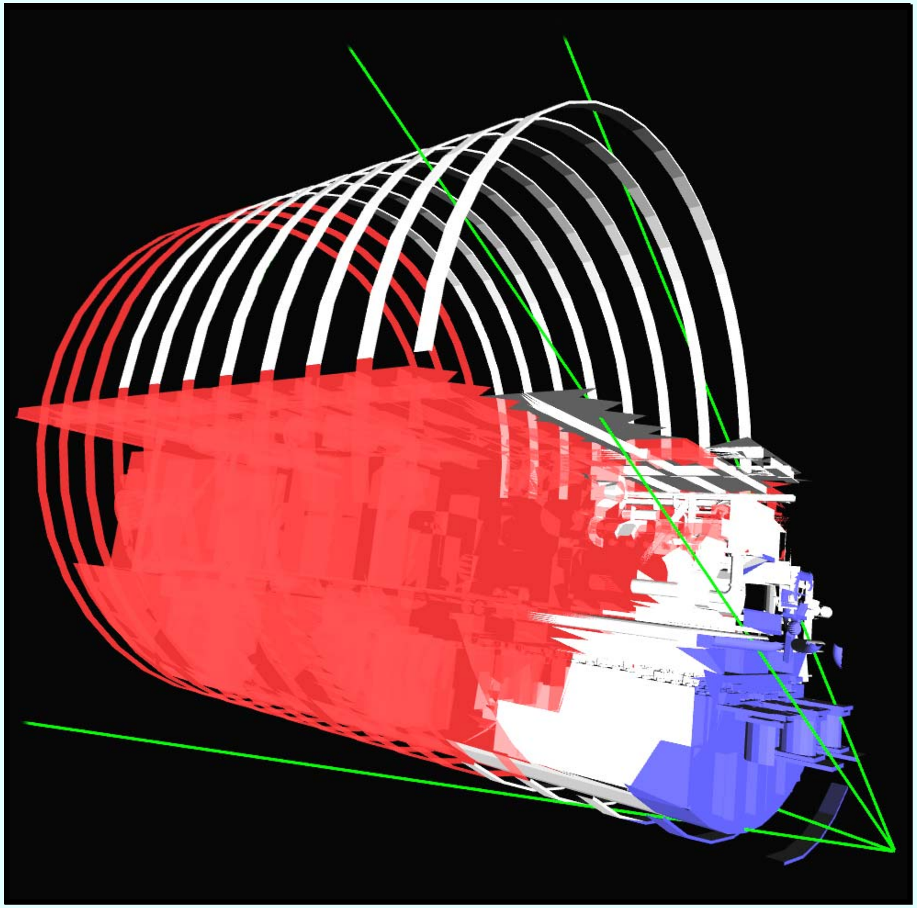
- they separate the coverage information from the depth information, two data structures
  - hierarchical occlusion maps
  - depth (several proposals for this)

HOM:Algorithm Outline

- – Select occluders until the set is large enough
- – Build occlusion representation
- – Occlusion culling & final rendering

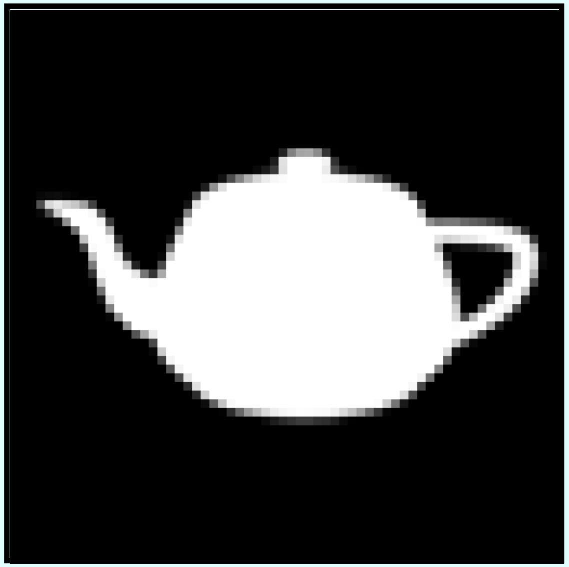


Demonstration

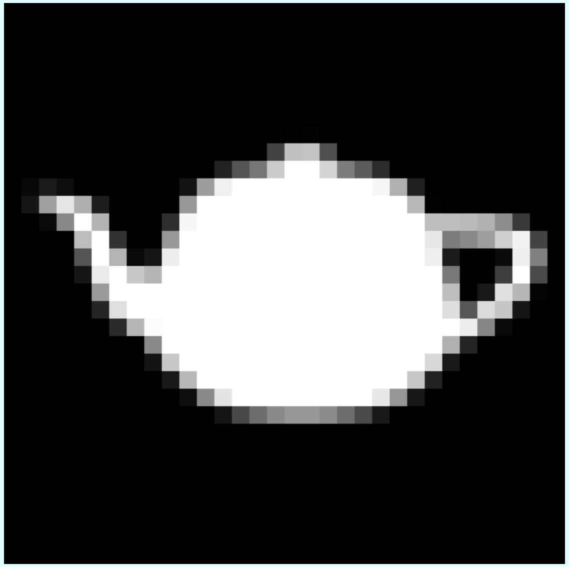


Blue parts: occluders
Red parts: occludees

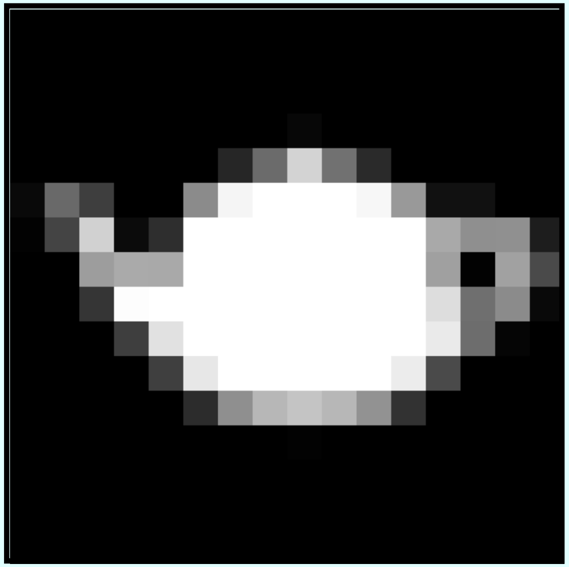
Occlusion Map Pyramid



64 x 64

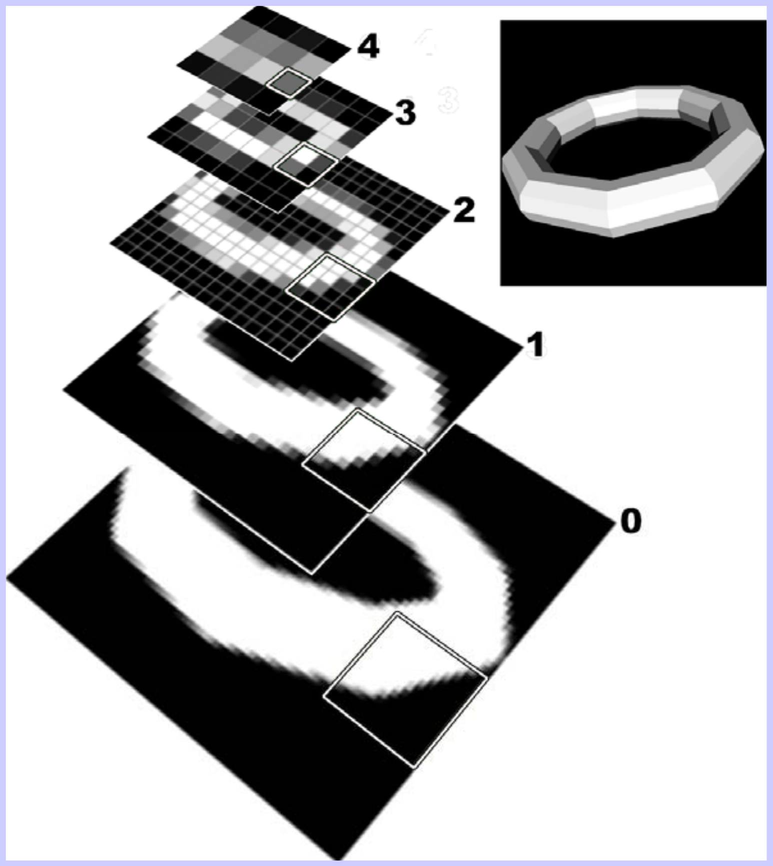


32 x 32



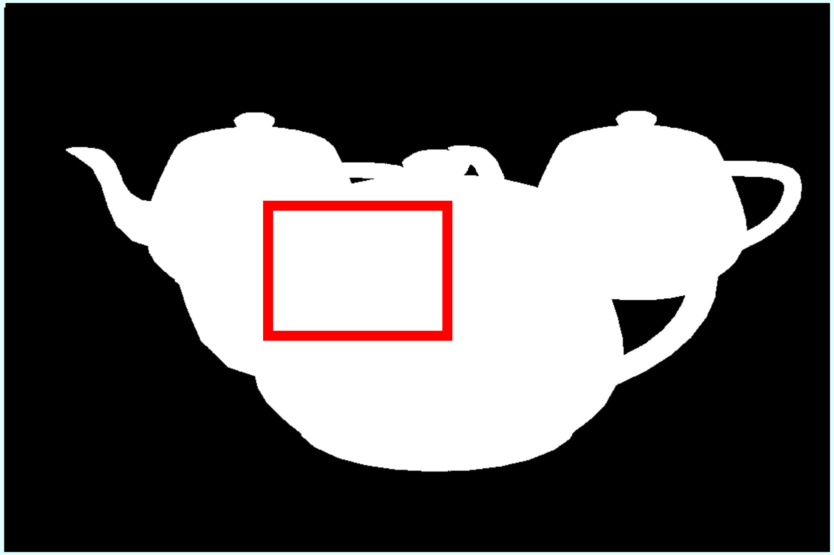
16 x 16

Occlusion Map Pyramid



Representing Occluders

Set of Occluders



Occlusion Map

Aggressive Approximate culling