Hierarchical Methods

Yiorgos Chrysanthou

University College London

Outline

- Introduction
- View volume culling
- Back face culling
- Occlusion culling
  - Hierarchical Occlusion Maps
- Conclusion

Introduction

- Hierarchical methods are usually employed in culling algorithms
- They are used to quickly identify and discard the portions of the scene not visible to the viewer
- Remaining geometry is typically passed to a separate hidden surface removal algorithm to render final image

Introduction

- A hierarchy can be placed:
  - in objects space
    - used for clustering objects together, one test on the extend of a cluster can classify everything within
  - in image space
    - Occluded pixels are grouped, anything projecting onto an already occluded region can be discarded
  - over time
    - not often explored

Introduction

- Hierarchical methods have been used in all 3 classes of culling:
  - view volume culling
  - back face culling
  - occlusion culling

View Volume Culling

- Scene geometry is placed into a hierarchy based on spatial proximity, eg:
  - bounding volumes [Clark 76]
  - octrees
  - BSP trees [Naylor92]
- Hierarchy is recursively compared against the view volume, usually in object space [Clark 76] but occasionally in view space [Bartz 99]

Back Face Culling

- Scene is placed into a hierarchy based on spatial proximity and direction of normals
- Hierarchy is recursively tested against the view parameters (view position and direction)

Occlusion Culling

- We usually have hierarchies in both object space and image space
- Many different methods exist, eg:
  - [Naylor 92] 2D BSP for image and 3D BSP for scene
  - [Greene 93] Z-pyramid for image and octree for scene
  - [Zhang 97] hierarchical occlusion map for image, bounding box hierarchy for scene

Occlusion Culling Example: Hierarchical Occlusion Maps

- A number of objects are rendered into the initial occlusion map
- The occlusion map is built into a hierarchy
- Objects are placed into a bounding box (BB) hierarchy
- The BB hierarchy is traversed and compared against the hierarchy of occlusion maps

Occlusion Maps

- An occlusion map
  - Corresponds to a screen subdivision
  - Records average opacity for each partition
- Can be generated by rendering occluders
  - Record pixel opacities (pixel coverage)
- Merge projections of occluders
- Represent occlusion in image-space

Occlusion Maps

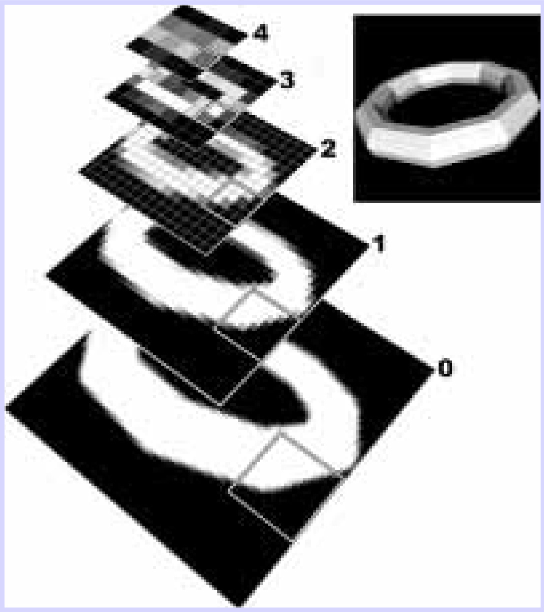
Rendered Image

Occlusion Map

Occlusion Map Pyramid

- Analyzing cumulative projection
  - A hierarchy of occlusion maps (HOM)
  - Made by recursive averaging (low-pass filtering)
  - Record average opacities for blocks of pixels
  - Represent occlusion at multiple resolutions
  - Construction accelerated by hardware

Occlusion Map Pyramid



Using the Occlusion Pyramid

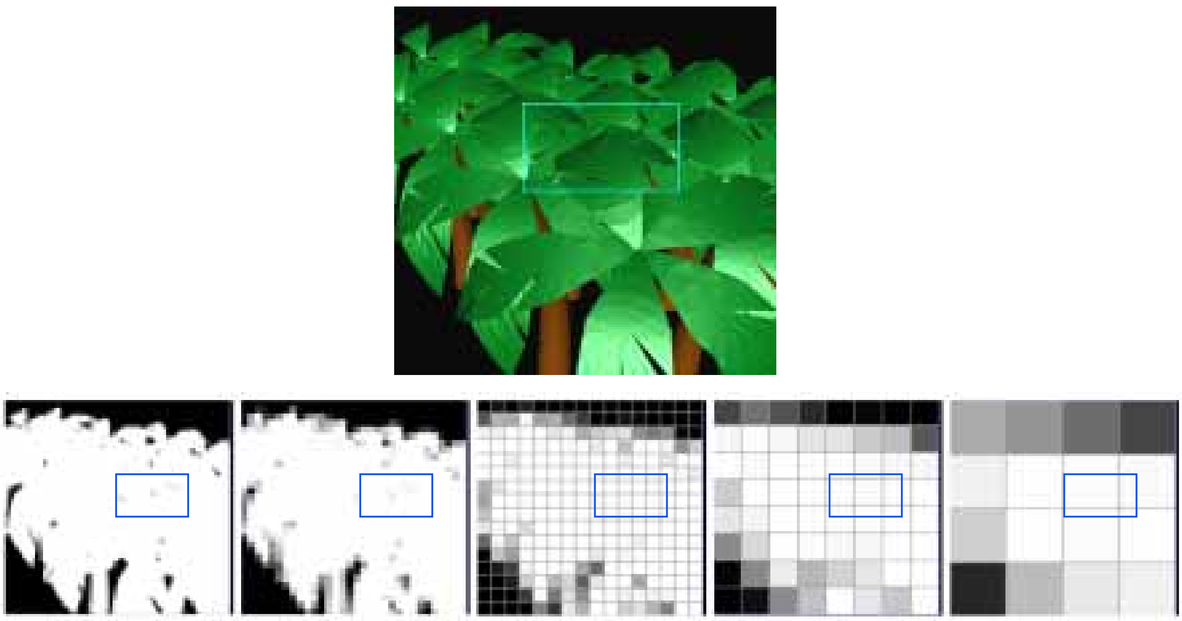
- As we traverse the BB object hierarchy, the faces of the bounding boxes are projected (bounding rectangles, BR) and tested to see if they satisfy two conditions:
  - overlap occluded regions
  - further away than the occluded regions
- If both tests give TRUE then the BB and its contents are occluded

Algorithm for Overlap Tests

- Given: HOM pyramid; the object to be tested
- Compute BR of and the initial level in the pyramid
- for each pixel touched by the BR

```
if pixel is fully opaque
        continue
else
        if level = 0
                return FALSE
        else
                descend...
```

Also Used for Approximate Culling



0 1 2 3 4

Resolving Depth

- There is a number of possibilities depending on information stored with HOM which can be:
  - A single plane corresponding to the farthest vertex of all occluders
  - A plane for each partition of the screen (depth estimation buffer)
  - The original Z-buffer of the occluders without the background depth values

The Method Tested

- It showed large speed-ups when used in scenes with high depth complexity
- It can be used with arbitrary models and any occluders
- Combines the occlusion of many polygons into one

Conclusion

- Hierarchical methods are a very useful tool
- The running time of algorithms based on them is typically logarithmic on the number of primitives
- They make for more scalable algorithms