Visibility

Tom Funkhouser
Princeton University
COS 426, Spring 2006

Outline

• Visibility culling
  ◦ Determine which surfaces are completely hidden with respect to the camera

• Hidden surface removal
  ◦ Determine which parts of which surfaces are hidden with respect to the camera

• Shadows
  ◦ Determine which parts of which surfaces are hidden with respect to the light sources

Visibility Culling & Hidden Surface Removal

• Determine which parts of which surfaces should be scan converted

Visibility Culling & Hidden Surface Removal

• Determine which parts of which surfaces should be scan converted

Visibility Culling & Hidden Surface Removal

Where should this be done in the rendering pipeline?

Visibility Culling & Hidden Surface Removal

Outline

• Visibility culling
  ◦ Determine which surfaces are completely hidden with respect to the camera

• Hidden surface removal
  ◦ Determine which parts of which surfaces are hidden with respect to the camera

• Shadows
  ◦ Determine which parts of which surfaces are hidden with respect to the light sources
Visibility Culling

- Back-face culling
- View frustum culling
- Occlusion culling

Back-Face Culling

- Do not draw polygons facing backwards with respect to camera

View Frustum Culling

- Do not draw polygons outside the view frustum
  - Accelerate checks with bounding volume hierarchies
**View Frustum Culling**

- Do not draw polygons outside the view frustum
  - Accelerate checks with bounding volume hierarchies or spatial data structures (e.g., octree)

**Occlusion Culling**

- Do not draw polygons completely occluded by other polygons

\[ \text{Occlusion} = \text{Depth} + \text{Overlap} \]

**Occlusion Culling**

- Cells & Portals

  - Camera view
  - Bird's eye view

**Outline**

- Visibility culling
  - Determine which surfaces are completely hidden with respect to the camera

- Hidden surface removal
  - Determine which parts of which surfaces are hidden with respect to the camera

- Shadows
  - Determine which parts of which surfaces are hidden with respect to the light sources

**Hidden Surface Removal**

- Object space
  - Weiler-Atherton
  - Depth sort

- Screen space
  - Scan-line
  - Area subdivision
  - Z-buffer
  - Ray casting

**Weiler-Atherton Polygon Clipping**

- Clip to polygons overlapping and in front of them
  - Computes exact visible areas of every polygon
  - Way slow!
### Depth sort
- "Painter’s algorithm"
  - Sort surfaces in order of decreasing maximum depth
  - Scan convert surfaces in back-to-front order

### 3D Rendering Pipeline
- 3D Primitives
- 3D Modeling Coordinates
- 3D World Coordinates
- Viewing Transformation
- Camera Coordinates
- Projection Transformation
- Screen Coordinates
- Clipping
- Visible Surface
- Rendering
- 2D Image Coordinates
- Depth sort
- Lighting

#### Depth sort comments
- \(O(n \log n)\)
- Better with frame coherence?
- Implemented in software
- Render every polygon
- Often use BSP-tree or static list ordering

### Hidden Surface Removal Algorithms
- Object space
  - Weiler-Atherton
  - Depth sort
- Screen space
  - Ray casting
  - Scan-line
  - Area subdivision
  - Z-buffer

### Ray Casting
- Fire a ray for every pixel
  - If ray intersects multiple objects, take the closest

### Ray Casting Pipeline
- Ray casting comments
  - Embarassingly parallel
  - Only shade each pixel once
  - No pixel coherence
  - Requires access to all polygons
  - Traditionally considered slow
Scan-Line Algorithm

- For each scan line, construct and sort spans

![Scan-Line Algorithm Diagram]

Scan-Line Algorithm

- For each scan line, construct and sort spans
  - Sort by depths within each scan line

![Scan-Line Algorithm Diagram]

Scan-Line Algorithm

- For each scan line, construct and sort spans

![Scan-Line Algorithm Diagram]

Area Subdivision

Warnock's algorithm

- Fill area if:
  - All surfaces are outside area, or
  - Only one surface intersects area, or
  - One surface occludes other surfaces in area
  - Otherwise, subdivide

![Area Subdivision Diagram]

Area Subdivision

- Color & depth of closest object for every pixel
  - Update only pixels whose depth is closer than in buffer
  - Depths are interpolated from vertices, just like colors

![Area Subdivision Diagram]
**Z-Buffer**

- **3D Primitives**
  - 3D Modeling Coordinates
  - 3D World Coordinates
  - Lighting
  - 3D Camera Coordinates
  - Projection Transformation
  - Clipping
  - 2D Screen Coordinates
  - 2D Image Coordinates
  - Z-Buffer

**Z-buffer comments**
- Polygons rasterized in any order
- Process one polygon at a time
- Suitable for hardware pipeline
- Requires extra memory for z-buffer
- Subject to aliasing (A-buffer)
- Commonly in hardware

**Outline**

- **Visibility culling**
  - Determine which surfaces are completely hidden with respect to the camera
- **Hidden surface removal**
  - Determine which parts of which surfaces are hidden with respect to the camera
- **Shadows**
  - Determine which parts of which surfaces are hidden with respect to the light sources

**Hidden Surface Removal Algorithm**

**Shadows**

- **Occlusions from light sources**

**Shadow Rays**

- **Cast ray towards light point for each lighting calculation**

\[
I = I_e + K_n I_e + \sum i (K_d (N \cdot L) + K_r (V \cdot R) \cdot S_i)
\]

**Shadow Rays**

- **General, simple**
- **Slow**

\[
I = I_e + K_n I_e + (K_d (N \cdot L) + K_r (V \cdot R) \cdot S_i)
\]
Shadow Maps

- Precompute image of depths from light
  - Store image of distances from light
  - Lookup depth of surface point in image when shade

[Image: Foley et al.]

- Suitable for hardware pipeline
  - Projection into light coordinate system is 4x4 matrix
  - Shadow map stored in texture

- Problems
  - Field of view
  - Aliasing & bias

[Image: Teller & Durand, MIT]

Shadow Volumes

- Compute planar boundaries of shadow volumes

[Image: Teller & Durand, MIT]

- Compute planar boundaries of shadow volumes
  - Draw shadow boundary planes with surfaces
  - Draw all in front-to-back order
  - Keep counter of shadow boundary crossings and light only if zero (not in shadow)

[Image: Teller & Durand, MIT]

Shadow Volumes

- Problems:
  - Requires sorting polygons front-to-back
  - Requires rendering of extra polygons

[Image: Teller & Durand, MIT]

Summary

- Visibility culling
  - Determine which surfaces are completely hidden with respect to the camera

- Hidden surface removal
  - Determine which parts of which surfaces are hidden with respect to the camera

- Shadows
  - Determine which parts of which surfaces are hidden with respect to the light sources