



# Visibility

Tom Funkhouser  
Princeton University  
COS 426, Spring 2006



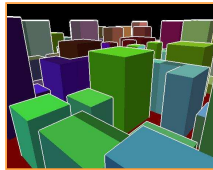
## Outline

- 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



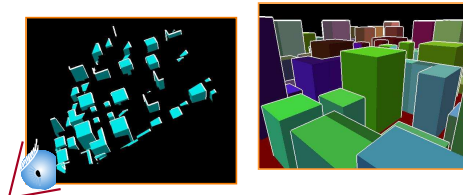
## Visibility Culling & Hidden Surface Removal

- Determine which parts of which surfaces should be scan converted

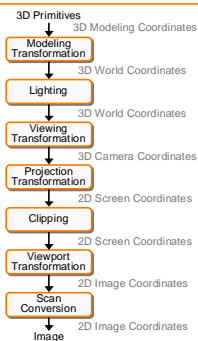


## Visibility Culling & Hidden Surface Removal

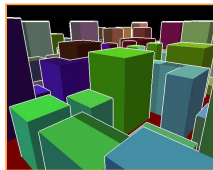
- Determine which parts of which surfaces should be scan converted



## Visibility Culling & Hidden Surface Removal



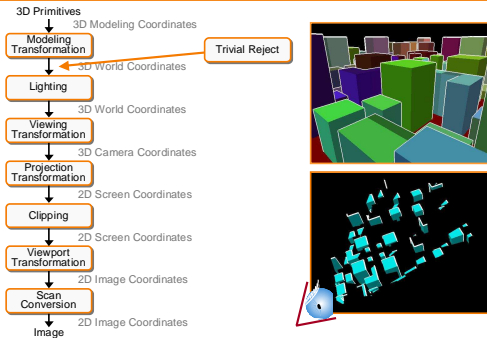
Where should this be done in the rendering pipeline?



## 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



## Visibility Culling

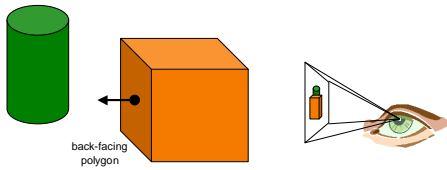


- Back-face culling
- View frustum culling
- Occlusion culling

## Back-Face Culling



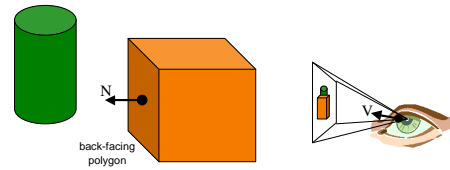
- Do not draw polygons facing backwards with respect to camera



## Back-Face Culling



- Do not draw polygons facing backwards with respect to camera

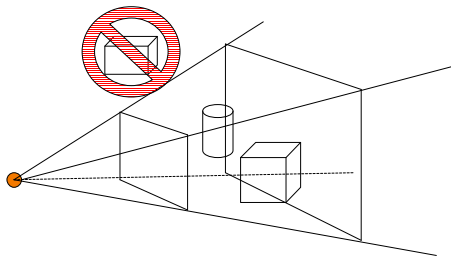


A polygon is backfacing if  $V \cdot N > 0$

## View Frustum Culling



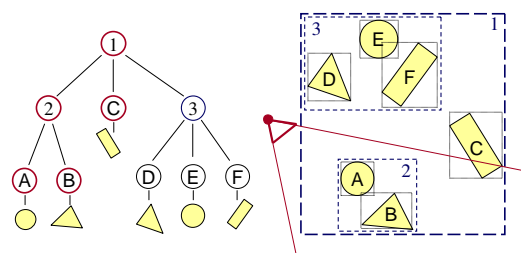
- Do not draw polygons outside the view frustum



## View Frustum Culling



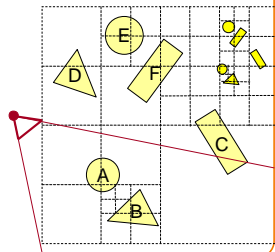
- Do not draw polygons outside the view frustum
  - Accelerate checks with bounding volume hierarchies



## View Frustum Culling



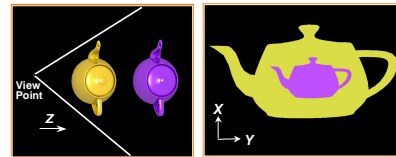
- Do not draw polygons outside the view frustum
  - Accelerate checks with bounding volume hierarchies or spatial data structures (e.g., octree)



## Occlusion Culling



- Do not draw polygons completely occluded by other polygons



Depth

Overlap

Occlusion = Depth + Overlap

Luebke

## Occlusion Culling



- Cells & Portals



Camera view



Bird's eye view

Luebke

## 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

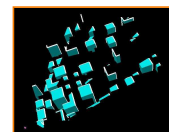
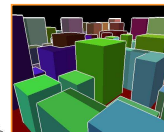
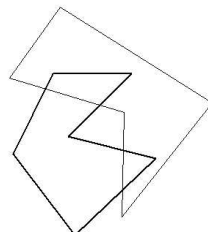


- Object space
  - Weiler-Atherton
  - Depth sort
- Screen space
  - Scan-line
  - Area subdivision
  - Z-buffer
  - Ray casting

## Weiler-Atherton Polygon Clipping



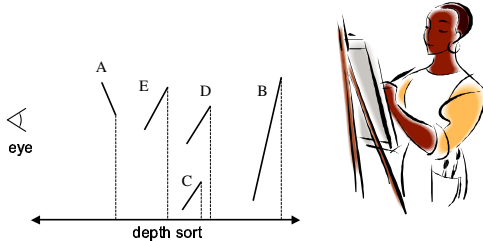
- Clip to polygons overlapping and in front of them
  - Computes exact visible areas of every polygon
  - Way slow!



## Depth sort



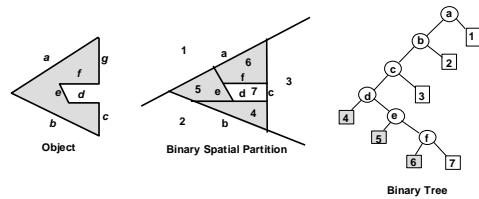
- "Painter's algorithm"
  - Sort surfaces in order of decreasing maximum depth
  - Scan convert surfaces in back-to-front order



## BSP Tree

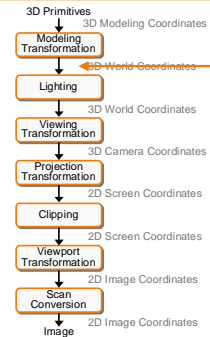


- Binary space partition with solid cells labeled
  - Constructed from polygonal representations
  - Provides linear-time depth sort for arbitrary view



Naylor

## 3D Rendering Pipeline



### 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

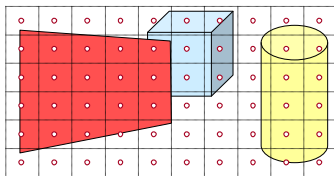


- Object space
  - Weiler-Atherton
  - Depth sort
- Screen space
  - Ray casting
  - Scan-line
  - Area subdivision
  - Z-buffer

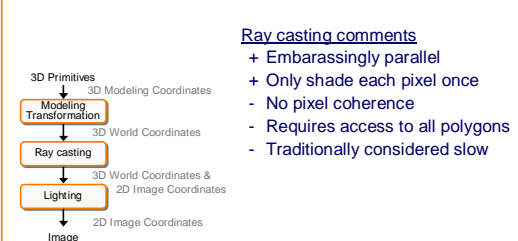
## Ray Casting



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



## Ray Casting Pipeline



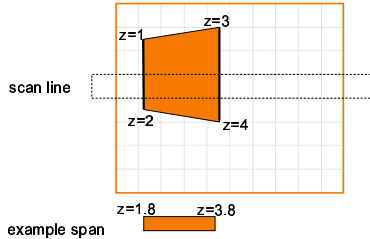
### Ray casting comments

- + Embarassingly parallel
- + Only shade each pixel once
- No pixel coherence
- Requires access to all polygons
- Traditionally considered slow

## Scan-Line Algorithm



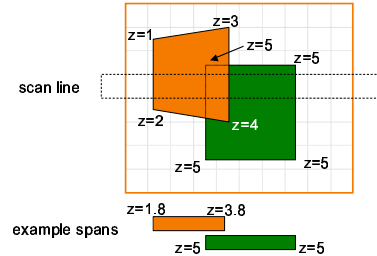
- For each scan line, construct and sort spans



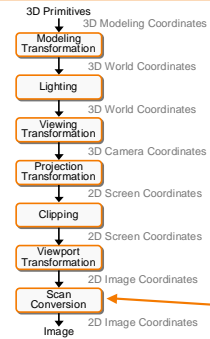
## Scan-Line Algorithm



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



## Scan-Line Algorithm



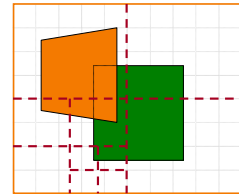
- Scan-line comments
- + Coherence among along scan lines
  - + Only shade each pixel once
  - Requires access to all polygons
  - Not suitable for hardware pipeline
  - o Commonly used in software

## Area Subdivision

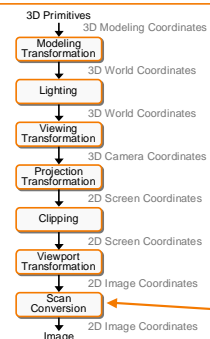


### Warnock's algorithm

- 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

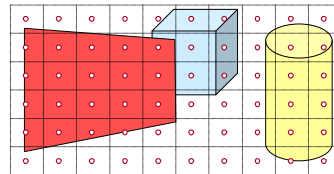


- Area subdivision comments
- o Augments scan conversion
  - o Further spatial coherence
  - o Not commonly used

## Z-Buffer



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

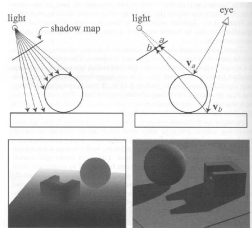




## Shadow Maps



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

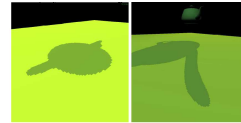


[Foley et al.]

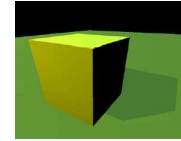
## Shadow Maps



- Suitable for hardware pipeline
  - Projection into light coordinate system is 4x4 matrix
  - Shadow map stored in texture
- Problems
  - Field of view
  - Aliasing & bias



Aliasing



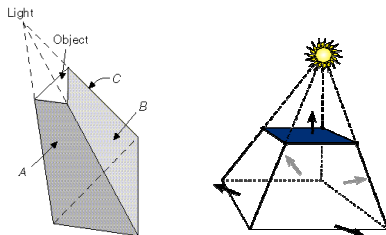
Bias

[Teller & Durand, MIT]

## Shadow Volumes



- Compute planar boundaries of shadow volumes

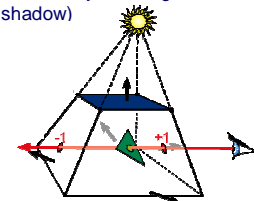


[Teller & Durand, MIT]

## Shadow Volumes



- 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)

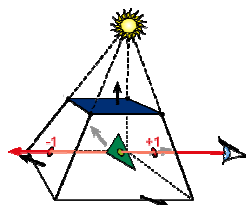


[Teller & Durand, MIT]

## Shadow Volumes



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



[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