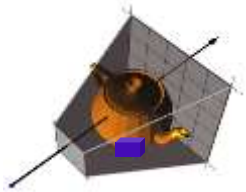
Topic 8:

Visibility

- Elementary visibility computations:
  - Clipping
  - Backface culling
- Algorithms for visibility determination
  - Z-Buffering
  - Painter's algorithm
  - BSP Trees

Visibility Problem

What is NOT visible?



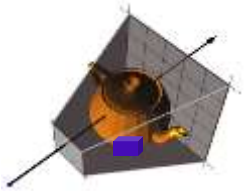
Visibility Problem

What is NOT visible?

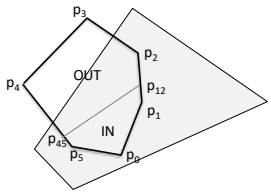
primitives outside of the field of view

back-facing primitives

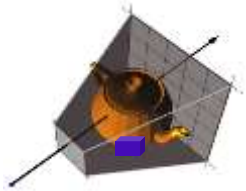
primitives occluded by other objects closer to the camera



Polygon Clipping (wrt to a single plane)



Input	Output
edge (p_k, p_{k+1})	
in, in	p_{k+1}
in, out	$p_{intersect}$
out, out	
out, in	$p_{intersect}$, p_{k+1}



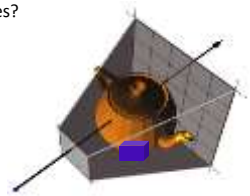
Polygon Clipping (wrt to a volume)

Clip with respect to each plane of the volume in sequence!

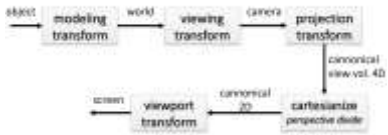
Does the order of the planes matter?

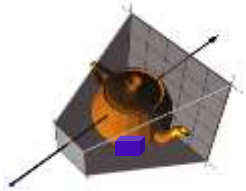
Does it work for concave polygons?

Does it work for concave volumes?

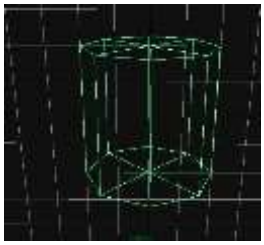


Polygon Clipping (when to clip?)

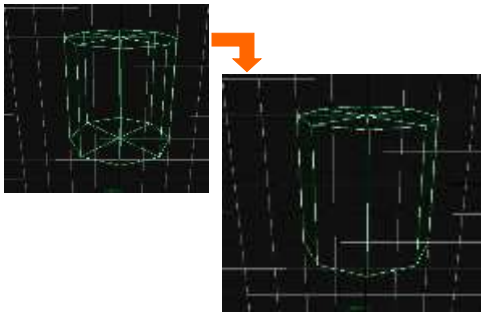




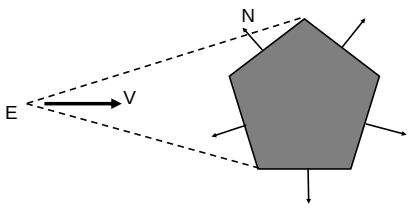
Backface culling



Backface culling

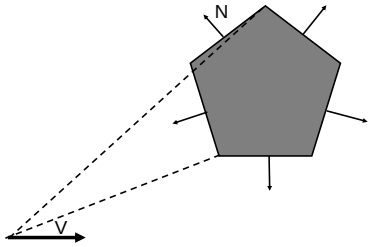


Backface culling



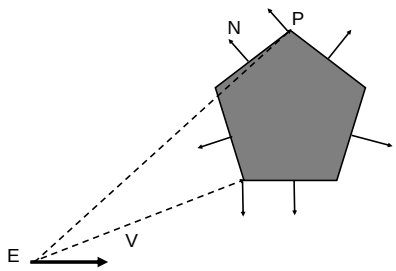
Backface culling

N.V > 0 is a back face?



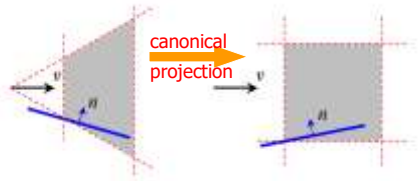
Backface culling

N.(P-E) >0



Backface culling (when to cull?)

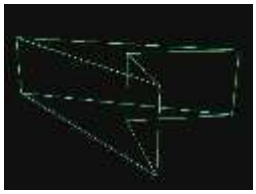
Where in the graphics pipeline can we do backface culling?



@alec: Would be nice to redo this image

Occluded faces

Does backface culling always determine visibility completely for a single object?

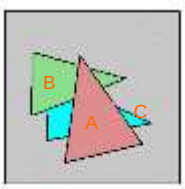


Occluded faces

In typical scenes some polygons will overlap, we must determine which portion of each polygon is visible to eye!

Painters Algorithm

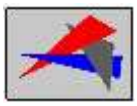
Sort primitives in Z.
Draw primitives back to front (CBA).

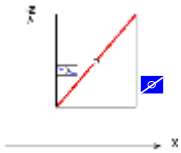


Painters Algorithm

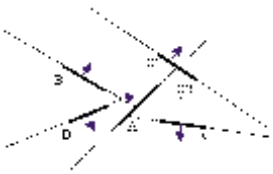
Problems

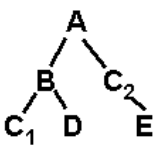
- **Large faces**
- **Intersecting faces**
- **Cycles**





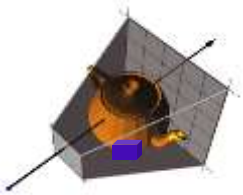
BSP tree





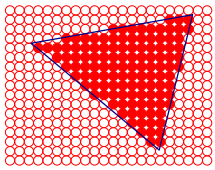
Visibility Problem

- Z-Buffer
- Scanline



Rasterization or Scan Conversion

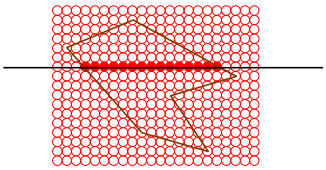
Rasterization takes shapes like triangles and determines which pixels to fill.



Filling Polygons

First approach:

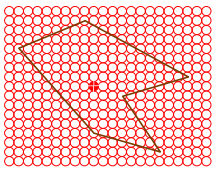
1. Polygon Scan-Conversion
   - Rasterize a polygon scan line by scan line, determining which pixels to fill on each line.



Filling Polygons

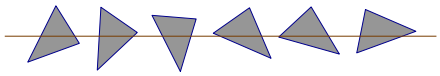
Second Approach:

2. Polygon Fill
   - Select a pixel inside the polygon. Grow outward until the whole polygon is filled.

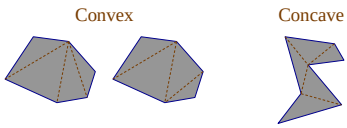


Triangles

Always convex: No matter how you rotate a triangle, it only has one span per scan line.

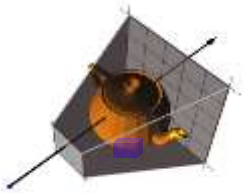


Any polygon can be decomposed into triangles.



Visibility Problem

- Z-Buffer
- Scanline
- A-Buffer



Visibility Problem

Image space algorithms

- Operate in display terms pixels, scanlines
- Visibility resolved to display resolution
- Examples: Z-buffer, ray-tracing
- O(n*resolution)

Object Space algorithms

- Analytically compute visible fragmer
- Examples: painters algorithm, BSP
- $O(n^2)$