CS174A Lecture 10

Announcements & Reminders

- *Team project proposals due (first draft): Oct 31*
- *Team project proposals due (final version): Nov 5*
- *Project #3 due Nov 3 midnight*

TA Session This Friday

- *Team project proposals*
- *Project assignment #3*
- *Midterm*

Last Lecture Recap

- *Backface Culling*
- *Geometric Calculations*
- *MIDTERM REVIEW*

Next Up

- ***Hidden Surface Removal***
  - Painter's algorithm
  - Z-buffer algorithm
  - Scanline z-buffer algorithm
- ***Flat and Smooth Shading***
- ***Lighting/Illumination Models***
- ***Hidden Surface Removal***
  - 2-pass z-buffer algorithm
  - Ray casting

Hidden Surface Removals

- **Object Types**
  - Polymesh
  - Free form surfaces
  - Volume
  - CSG
  - Implicit surfaces
- **Basic Operations**
  - Establish priorities among polygons, objects, etc.
  - Collect overlapping elements and use priorities to resolve visibility

Hidden Surface Removals

- **Algorithm Types**
  - Image Space: operations 1 and 2, both at pixel resolution
  - List-Priority: operation 1 at object resolution, operation 2 at pixel resolution
  - Object Space: operation 1 and 2, both at object resolution
- **Evaluation Criteria**
  - Flexibility: what types of objects can it handle?
  - Special Effects: transparency, antialiasing
  - Memory Requirements
  - Speed

Hidden Surface Removals

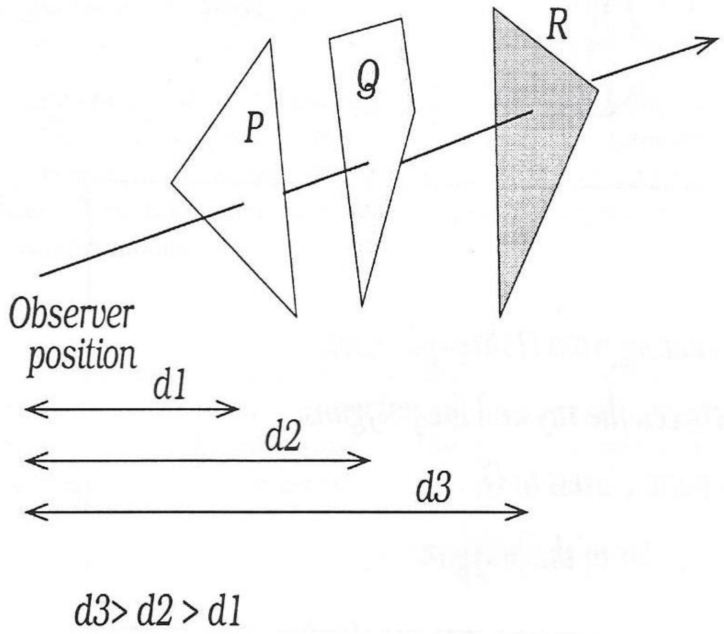
- Image Precision Algorithms
  - Painter’s
  - Z-Buffer
  - Scanline Z-Buffer
  - 2-Pass Z-Buffer
  - Ray Casting

Painter's Algorithm

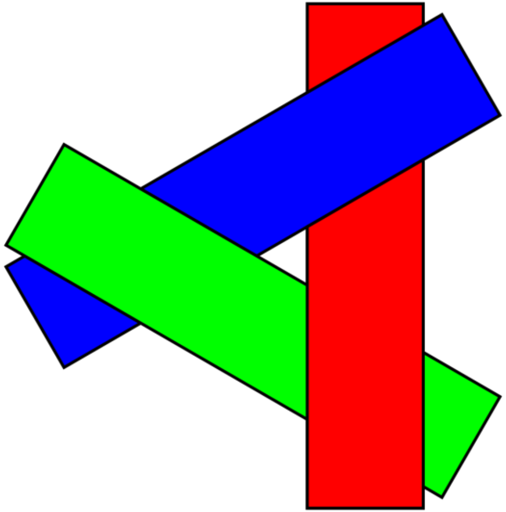
1. Sort all polygons by z-depth
2. Scan-convert polygons in back-to-front order
   So paint poly R first, then Q, then P

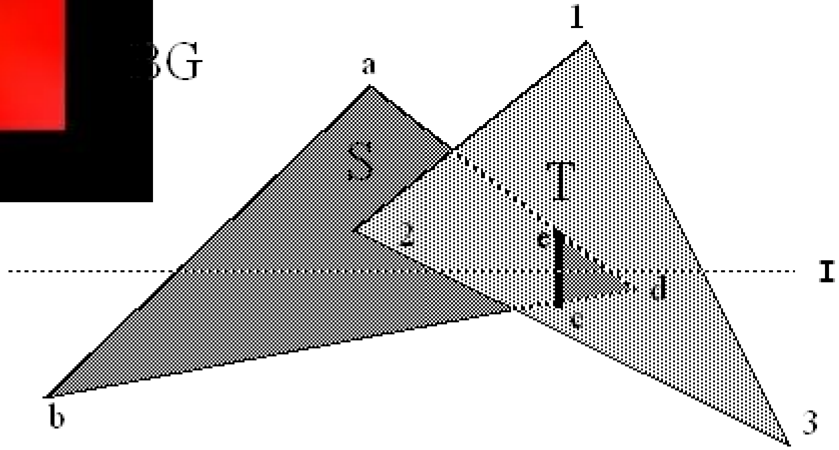
Cannot handle certain cases:

1. Cyclic polygons
2. Intersecting polygons



Painter’s Algorithm





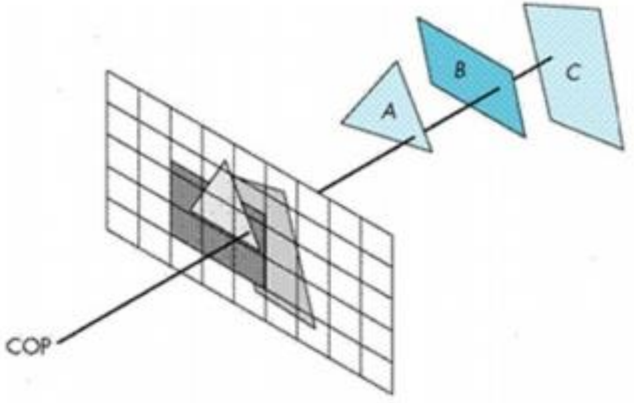
Z-Buffer Algorithm

$zb[X_{res}][Y_{res}] = \infty$

$cb[X_{res}][Y_{res}] = background$

for each polygon

1. for each pixel covered by polygon
   a) Calculate z for polygon at (x,y)
   b) If (z < zb[x][y])
      i. zb[x][y] = z
      ii. cb[x][y] = color of polygon



Z-Buffer Algorithm

- **Properties**
  - Image precision algorithm
  - Easy to implement in software and hardware
  - Polygons scan-converted into framebuffer in random order
  - No pre-sorting of objects/polygons necessary

Z-Buffer Algorithm

- **Disadvantages**
  - Memory requirements for storing color and depth for entire image
  - Aliasing issues
  - Complexity depends on polygon’s projection area on the screen
  - Hard to handle transparency

Z-Buffer Algorithm

- **Advantages**
  - Handles penetrating and cyclic objects
  - Extends to various kinds of faces other than polygons
  - Simplicity and ease of software implementation
  - Easily implementable in hardware
  - Be modified to reduce memory requirements
  - Can be extended to A-buffer to reduce aliasing
  - Theoretically it can handle any number of polygons

Scanline Z-Buffer Algorithm

zb[X_{res}]; cb[X_{res}];

```
for each scanline (y = scanline)
    for each polygon which intersects scanline
        scan convert for specific scanline; determine span segment
        for each pixel in span segment (x = pixel location)
            Calculate z for polygon at (x,y)
            if (z < zb[x])
                zb[x] = z
                cb[x] = color of poly
```

Scanline Z-Buffer Algorithm

- **Advantages**
  - Same as z-buffer
  - Less memory requirement than z-buffer
- **Disadvantages**
  - Multiple passes through polygon database

Efficiency Considerations in Z-Buffer Algorithms

- **Speed Considerations**
  - Bounding box testing
    - Y_{min}/Y_{max} test: associate Y_{min}/Y_{max} with each face (Step 2)
    - Calculate left and right ends: x-intercept with scanline (Step 3)
  - Incremental calculation of Z or tri-linear interpolation (Step 5)

    $Ax + By + Cz + D = 0$

    $z = -\frac{Ax+By+D}{C}$

    $z_{x+1} - z_x = -\frac{A}{C} \quad \Rightarrow \quad z_{x+1} = z_x + \Delta z$ (where $\Delta z = -A/C$)
  - Space subdivision
  - Hierarchical subdivision