CS174A Lecture 10

Announcements & Reminders

- *11/08/22: Team project proposals due, initial version*

- *11/09/22: A3 due*

- *11/10/22: Midway demo, online zoom*

- *11/20/22: Assignment #4 due*

- *11/22/22: Team project proposals due, final version*

- *11/29/22: Prof Demetri's talk*

- *12/02/22 (Discussion Sessions): Team project presentations*

- *12/06/22: Final Exam, 6:30-9:30 PM PST, in class, in person*

TA Session This Friday

- *Team project proposals*
- *Assignment #3*
- *Gouraud and Phong shadings*

Last Lecture Recap

- *Geometric Calculations*
- *Midterm*

Next Up

- *Hidden Surface Removal*
  - Backface culling
  - Painter's algorithm
  - Z-buffer algorithm
  - Scanline z-buffer algorithm
- *Lighting/Illumination Models*
- *Flat and Smooth Shading*
- *Shadow Algorithms*
  - 2-pass z-buffer algorithm
- *Hidden Surface Removal*
  - Ray casting, ray tracing

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

Backface Culling

N = outward normal vector of face
P = a point on face
E = eye vector (from a point on face to eye = Eye - P)

It is a front face if:
- In World Space: $N \cdot E > 0$
- In Eye/Camera Space: $N \cdot (-P) > 0$ or $N \cdot P < 0$
- In Projection Space (after perspective division): $N_z < 0$

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/World/Eye 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

- Object Space Algorithms
  - Painter's

- Image Precision Algorithms
  - Z-Buffer
  - Scanline Z-Buffer
  - Ray Casting
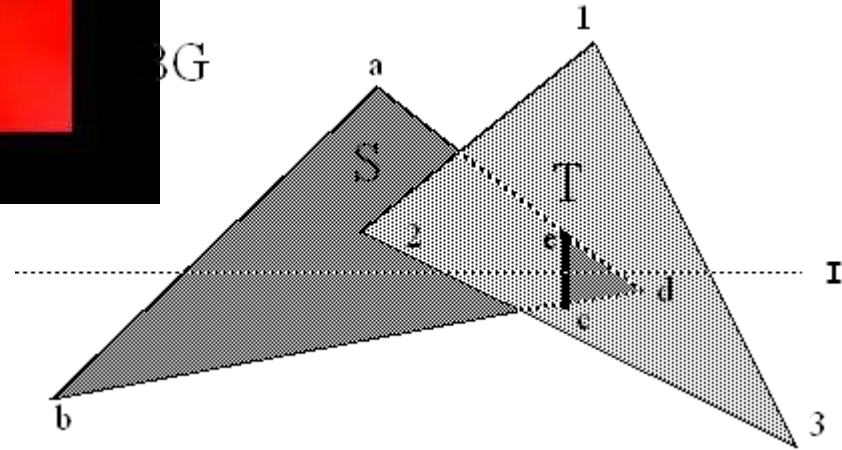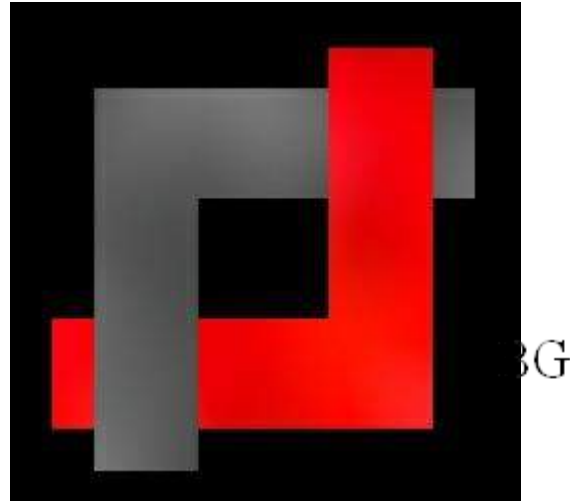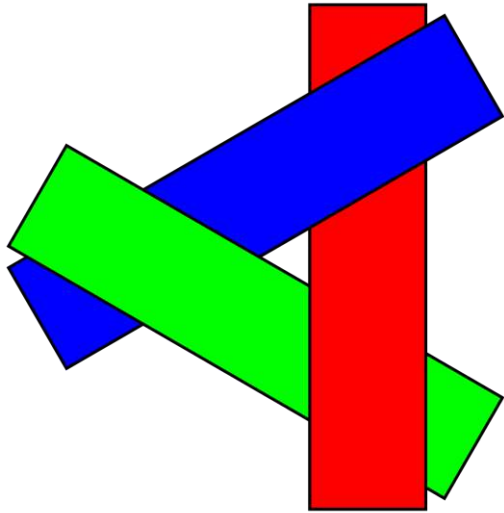
Painter's Algorithm

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

Observer position

d1

d2

d3

d3 > d2 > d1
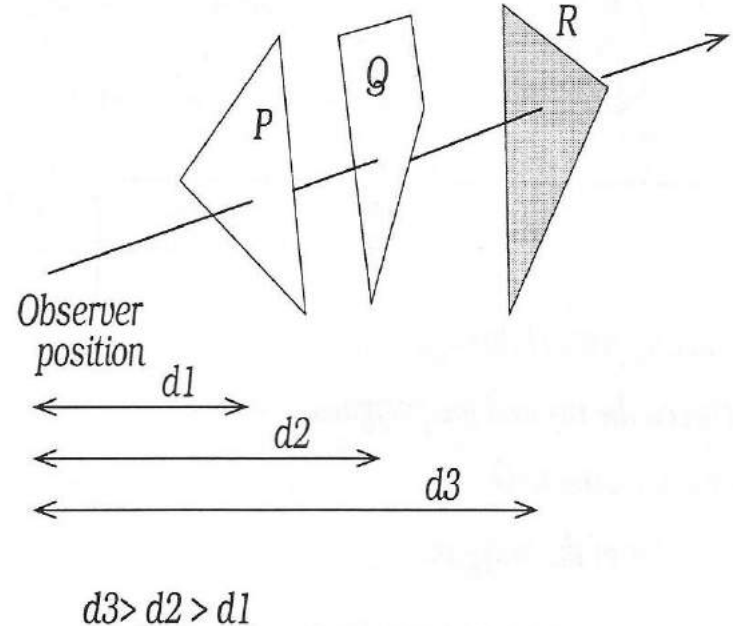
Painter's Algorithm

Z-Buffer Algorithm

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

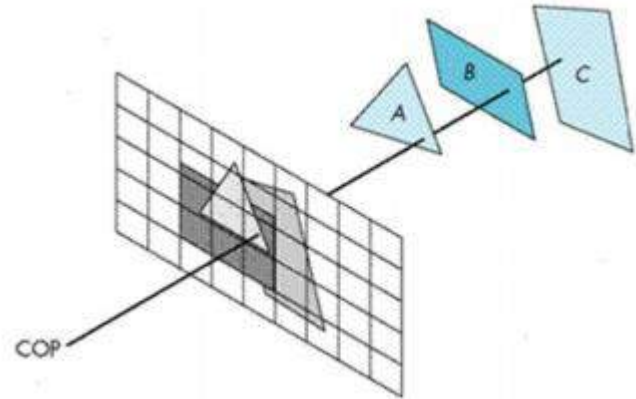$cb[X_{res}][Y_{res}]$ = background

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

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

1) for each scanline (y = scanline, reset zb and cb)

2)   for each polygon which intersects scanline

3)   scan convert for specific scanline; determine span segment

4)   for each pixel in span segment (x = pixel location)

5)   Calculate z for polygon at (x,y)

6)   if (z < zb[x])

7)   zb[x] = z

8)   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 bilinear interpolation (Step 5)
    Ax + By + Cz + D = 0
    $$z = - \frac{Ax+By+D}{C}$$
    $$z_{x+1} - z_x = - \frac{A}{C} \quad \Rightarrow \quad z_{x+1} = z_x + \Delta z \text{ (where } \Delta z = -A/C)$$
  - Space subdivision
  - Hierarchical subdivision