Hidden Surface Removal

April 27, 2006

Assignment 4 Hints

- Ed Angel 5.9.2:

$$\begin{bmatrix} \frac{-2\,far}{right-left} & 0 & \boxed{\frac{right+left}{right-left}} & 0 \\ 0 & \frac{-2\,far}{top-bottom} & \boxed{\frac{top+bottom}{top-bottom}} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & \frac{2\,far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

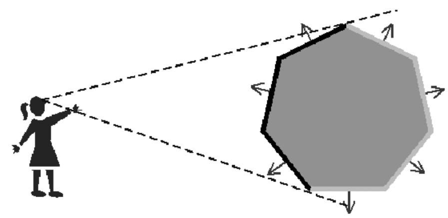
- Shear or Translation?
- Using only integers in Bresenham’s?

Hidden Surface Removal

- Object-space algorithms:
  - Back-face culling (removal)
  - Depth sorting and Painter's algorithm
- Image-space algorithm:
  - Z Buffer!
  - Fast, but requires more memory.

Back-Face Culling

- For convex objects, we can't see the *back faces.*
- But, how do we determine the back faces?



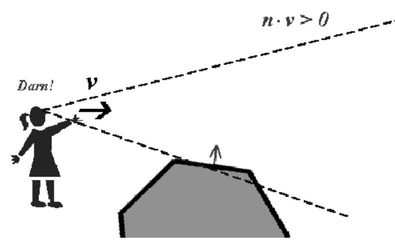
Removing Back-Faces

Idea: Compare the normal of each face with the viewing direction

Given n, the outward-pointing normal of F

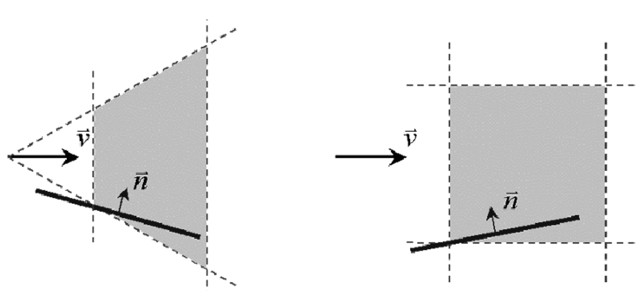
foreach face F of object
  if (n · v > 0)
    throw away the face

Does it work?



Fixing the Problem

We can't do view direction clipping just anywhere!



Downside: Projection comes fairly late in the pipeline. It would be nice to cull objects sooner.

Upside: Computing the dot product is simpler. You need only look at the sign of the z.

Culling Plane-Test

Here is a culling test that will work anywhere in the pipeline.

Remove faces that have the eye in their negative half-space. This requires computing a plane equation for each face considered.

$$\begin{bmatrix} n_x & n_y & n_z & -d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0$$

We'll will still need to compute the normal (How?).

But, we don't have to normalize it. (How do we go about computing a value for *d*?)

Painter's Algorithm

- Draw from back to front.
- No solution for:
  - Cyclic ordering
  - Intersecting surfaces

Z Buffer

- At each pixel, store the Z of the front-most surface.
- If the new Z is larger, it's occluded.
- If the new Z is smaller, then:
  - Draw the new surface
  - Update the Z

Other Algorithms

- Scan-line algorithm: See Section 7.11 of Ed Angel's book (4^{th} Ed).
- For more advanced research in this area, see:
  - Chen and Wang, SIGGRAPH 1996.
  - Snyder and Lengyel, SIGGRAPH 1998.

from the previous lecture…

Projection Matrix

$$\begin{bmatrix} wx' \\ wy' \\ wz' \\ w \end{bmatrix} = \begin{bmatrix} \frac{-2\,far}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{-2\,far}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & \boxed{-\frac{far+near}{far-near}} & \boxed{\frac{2\,far*near}{far-near}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Range of Z

- If Z = near, what is Z'?

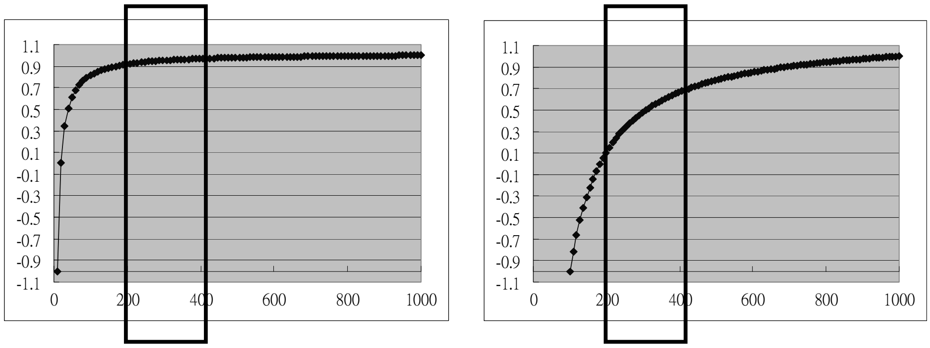
  -1
- If Z = far, what is Z'?

  1
- Does Z' change linearly with Z?
  - No!
  - Z' = wZ' / w = (a*Z+b) / Z = a + b/Z

Z Resolution

- Since screen Z' is expressed in the form of a+b/Z, most of the Z resolution is used up by the Z's closer to the near plane.
- So, what does this mean?
- You shouldn't set zNear to be very close to the eye position.

Near=10 → **Near=100**
Far=1000 → Far=1000



Notice the change in the range of Z after transformation (in NDC space) for the original Z (in eye space) between 200 and 400 (marked by the Red boxes).

Why Not Linear?

- To make it linear, we will have to make WZ' = $a*Z^2 + bZ$ (so that Z' = WZ'/W = $a*Z + b$)
- But that's impossible with the perspective matrix…

Linear Z Buffer or W Buffer

- Wait! Why is linear Z impossible under perspective projection? Can't we simply ignore the divide-by-w step for Z?
- Yes, but we no longer have the nice math of the homogeneous coordinates

$$\begin{bmatrix} wx' \\ wy' \\ wz' \\ w \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}, \qquad \begin{bmatrix} wx' \\ wy' \\ z' \\ w \end{bmatrix} = ?$$

Division by w