Depth

Dinesh K. Pai

Textbook Chapter 11

Several slides courtesy of M. Kim

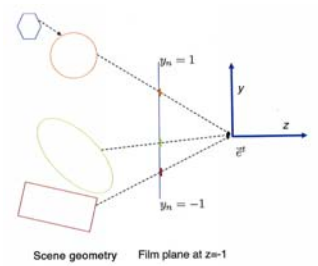
1

Announcements

- Midterm 2 results will be discussed next class
- Assignment 4 grading will be Friday-Wednesday
- Assignment 3 showcase

2

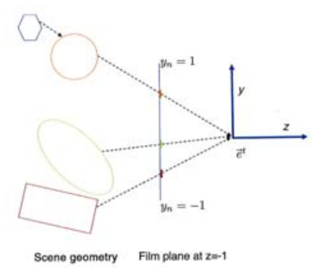
Visibility



- In the real world, opaque objects block light.
- We need to model this computationally.
- One idea is to render back to front and use overwriting
  - This will have problem with visibility cycles.

3

Visibility



- We could explicitly store everything hit along a ray and then compute the closest.
  - Make sense in a ray tracing setting, where we are working one pixel per ray at time, but not for OpenGL, where we are working one triangle at a time.

4

Z-buffer

- We will use z-buffer (or depth buffer)
- Triangles are drawn in any order
- Each pixel in frame buffer stores 'depth' value of closest geometry observed so far.
- When a new triangle tries to set the color of a pixel, we first compare its depth to the value stored in the z-buffer.
- Only if the observed point in this triangle is closer, we overwrite the color and depth values of this pixel.

5

Z-buffer

- This is done per-pixel, so there is no cycle problem.
- There are optimizations, where z-testing is done before the fragment shading is done.

6

Other uses of visibility calculations

- Visibility to a light source is useful for shadows.
  - We will talk about shadow mapping later.
- Visibility computation can also be used to speed up the rendering process.
  - If we know that some object is occluded from the camera, then we don't have to render the object in the first place.
  - We can use a conservative test.

7

Basic mathematical model

- For every point, we define its $[x_n, y_n, z_n]^t$ coordinates, using the following matrix expression:

$$\begin{bmatrix} x_n w_n \\ y_n w_n \\ z_n w_n \\ w_n \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} s_x & 0 & -c_x & 0 \\ 0 & s_y & -c_y & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix}$$

- We now also have the value $z_n = \frac{-1}{z_e}$
- Our plan is to use this z_n value to do depth comparison in our z-buffer.

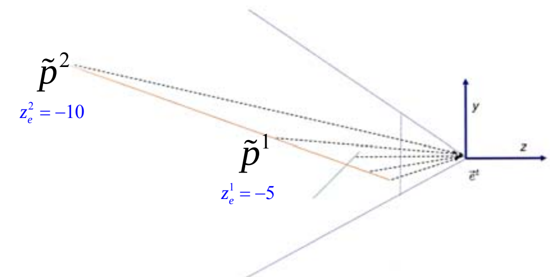
8

Correct ordering

- Given two points $\tilde{p}^1$ and $\tilde{p}^2$ with eye coordinates $[x_e^1, y_e^1, z_e^1, 1]^t$ and $[x_e^2, y_e^2, z_e^2, 1]^t$.
- Suppose that they both are in front of the eye, i.e., $z_e^1 < 0$ and $z_e^2 < 0$.
- And suppose that $\tilde{p}^1$ is closer to the eye than $\tilde{p}^2$, that is $z_e^2 < z_e^1$
- Then $-\frac{1}{z_e^2} < -\frac{1}{z_e^1}$,

  meaning

  $z_n^2 < z_n^1$



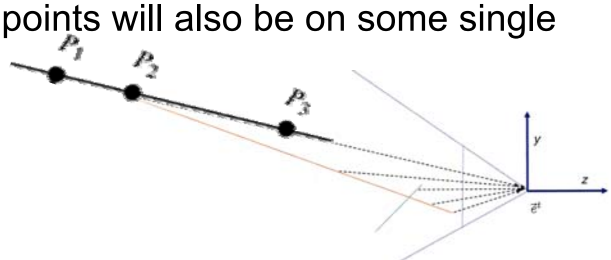
Projective transform

- We can now think of the process of taking points (given by eye coordinates) to points (given by normalized device coordinates) as an honest-to-goodness 3D geometric transformation.
- This kind of transformation is generally neither linear nor affine, but is something called a *3D projective transformation.*
- Projective transformation preserves co-linearity and co-planarity of points.

10

Co-linearity of points

- If three or more points are on a single line, the transformed points will also be on some single line.



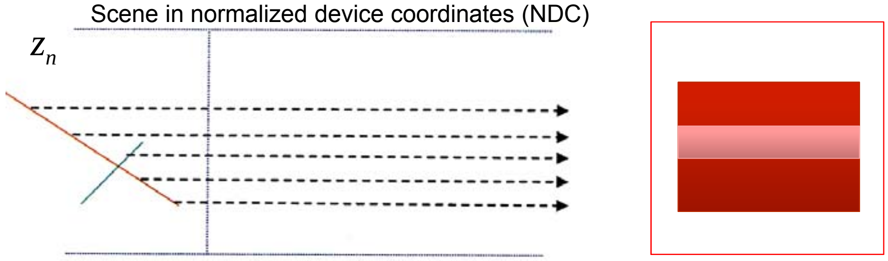
- Three points $\mathbf{x}_i = [x_i, y_i, z_i, 1]$ for $i = 1,2,3$

$$x_2 - x_1 : y_2 - y_1 : z_2 - z_1 = x_3 - x_1 : y_3 - y_1 : z_3 - z_1$$

$$\left|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_1 - \mathbf{p}_3)\right| = 0$$

11

Co-planarity of points



- Note that distances are not preserved by a projective transform.
- Evenly spaced pixel on the film do not correspond to evenly spaced points on the geometry in eye space.
- Meanwhile, such *evenly spaced pixels* correspond with evenly spaced points in *normalized device coordinates.*

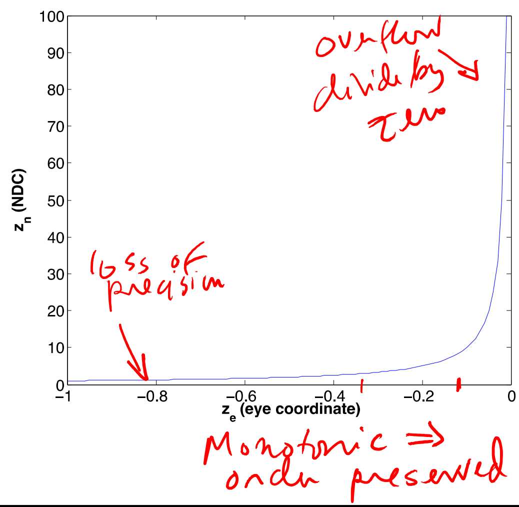
14

Numerics

- points very far from the eye have z_n values very close to zero

$$z_n = \frac{-1}{z_e}$$

```
ze=-1*[0.01:0.01:10];
zn=-1./ze;
plot(ze(1:100),zn(1:100))
```



18

How to use it

- In OpenGL, the z-buffer is turned on with a call to glEnable(GL_DEPTH_TEST).
- We may also need a call to glDepthFunc(GL_GREATER), since we are using a right handed coordinate system where ‘more-negative’ is ‘farther from the eye’.
- In practice, you may see other conventions (for how to interpret n and f, some of the signs of the matrix, and the handedness of the ultimate z-test.

19

Solution: Use near & far planes in projection

Note Title 2014-03-24

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & -1 & 0 \end{bmatrix} \xrightarrow{\text{generally}} P = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \alpha & \beta \\ \cdot & \cdot & -1 & \cdot \end{bmatrix}$$

Apply to $q = \begin{pmatrix} 0 \\ 0 \\ z_e \\ 1 \end{pmatrix}$ say

So $Pq = \begin{pmatrix} 0 \\ 0 \\ \alpha z_e + \beta \\ -z_e \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix}$

We can pick α & β to make $n \to +1$, $f \to -1$

i.e.

$$\boxed{\begin{aligned} \alpha n + \beta &= -n \\ \alpha f + \beta &= f \end{aligned}}$$

Reason: $\frac{z_c}{w_c} = +1$, ie $z_e = -n$

$\frac{z_c}{w_c} = -1$, ie $z_e = -w_c = f$

Solve:

$$\alpha(f-n) = f+n$$

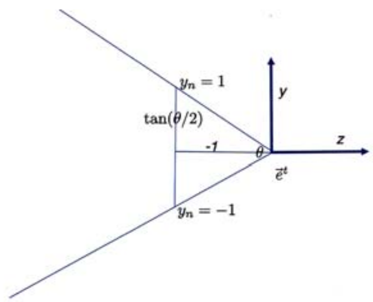
$$\alpha = \frac{f+n}{f-n}$$

Plug into 2nd eqn

$$\beta = f - \alpha f = f - \left(\frac{f+n}{f-n}\right) f$$

$$= \frac{\cancel{f^2} - fn - \cancel{f^2} - fn}{f-n} = \frac{-2fn}{f-n}$$

Glm: Perspective Eye coords → Clip coords



$$\begin{bmatrix} \frac{1}{\alpha \tan\left(\frac{\theta}{2}\right)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan\left(\frac{\theta}{2}\right)} & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

20

21