Perspective Interpolation and Normal Transformations

Zhengyun Zhang

CS248 Section, Friday, January 21, 2011
Errata
(mistake on my part)

- linear interpolation of depth in Assignment 1 is actually correct for the depth buffer!
- your starter code in Assignment 2 will thus have correct depth interpolation
- will be explained later in this session
Outline

- Additional State in Assignment 2
- Perspective Interpolation
  - foreshortening
  - rational linear interpolation
- Transformations on Normals
Assignment 2

- In addition to depth, you will need to compute at each pixel, interpolated:
  - Texture coordinate \((u,v)\)
  - Eye space normal
  - Eye space position
  - Positions of lights in eye space
  (all lighting computations done in eye space)
Foreshortening

viewer

wall

wall

wall
Foreshortening
Foreshortening
Foreshortening
Foreshortening

0 0.5? 1 1.5 2
Foreshortening

Diagram showing various points labeled 0, 0.5?, 1, 1.5, 2.
Foreshortening
Foreshortening

like orthographic projection!

eye coordinates

proj.
trans.

clip coordinates
Foreshortening

- eye coordinates
  \((x,y,z)\) coordinates relative to eye

- clip coordinates (after division)
  \((x_c,y_c,z_c)\) coordinates are foreshortened!
  i.e. divided by \(w\) (including \(z_c\)!

- clip \(z_c\) is rational linear function of eye \(z\)

- \(z\) coordinate ordering is preserved!
Foreshortening

proj.
trans.

foreshortened $z_c$
(valid for depth test)
Foreshortening

- can use clip $z_c$ value for depth test
- foreshortened coordinates interpolate linearly (affine function of projected $x_c, y_c$)
- what about texture coordinates and normals? (we want original values, not divided by $w$)
Rational Linear Interpolation

- $z_c = z/w$ (division for homogeneous coordinate)
- $z/w$ interpolates linearly in projective space
- ($u,v$) texture coordinates and ($x_n,y_n,z_n$) normal coordinates don’t, as discussed earlier
Rational Linear Interpolation

- affine functions of \((a,b,c)\) interpolate linearly in the space of \((a,b,c)\)

- any affine function of \((x,y,z)\) divided by \(w\) is an affine function of \((x/w,y/w,z/w)\) (proof in next slide)

- if texture coordinate \(u\) varies affinely with \((x,y,z)\), then \(u/w\) would vary affinely with \((x/w,y/w,z/w)\)
Rational Linear Interpolation

\[ f(x, y, z) = A \left( \begin{array}{c} x \\ y \\ z \end{array} \right) + f_0 \]

\[ f(x, y, z)/w(z) = A \left( \begin{array}{c} x \\ y \\ z \end{array} \right) /w(z) + f_0/w(z) \]

\[ = A \left( \begin{array}{c} x/w(z) \\ y/w(z) \\ z/w(z) \end{array} \right) + f_0/w(z) \]
Rational Linear Interpolation

- solution: interpolate both \((u/w, v/w)\) and \(1/w\) or \((x_n/w, y_n/w, z_n/w)\) and \(1/w\) linearly
- divide \(1/w\) into \(u/w, \) etc. to obtain correctly interpolated \(u, \) etc.
- called rational linear interpolation
Rational Linear Interpolation

\( u = \frac{u/w}{1/w} \)

divide to get result

interpolate linearly

(called a varying coordinate in GLSL)
Rational Linear Interpolation

- implementation: interpolate u/w like you linearly interpolated z in Assignment 1 (e.g. keeping track of “slopes”)
- also interpolate 1/w at the same time
- divide at each pixel to obtain correct u coordinate
Summary of Perspective Interpolation

- need to interpolate carefully after projection
- z ordering is preserved by foreshortening, so linearly interpolate to get foreshortened z coordinates for depth test
- need actual texture or normal coordinates, not “foreshortened” ones, so need rational linear interpolation to obtain these per pixel
Normals

- normals are specified for each vertex
- interpolate the normals rational linearly, just like texture coordinates (an approximation, but works well and is used in practice)
- be careful about transformations!
Normal Transformations

- naive approach: apply model-view transformation on normal vector
- can result in cases like: scale
Normal Transformations

- let $M$ be the model-view matrix
- since normals are directions (or $w=0$)
  translations do nothing, so ignore 4th column
- since model-view matrix is non-projective,
  ignore 4th row as well
Normal Transformations

$L := \text{top left 3x3 matrix in } M$

$N := \text{desired normal transformation matrix}$

$n := \text{untransformed unit normal vector}$

$v_1, v_2 := \text{any two points on the plane}$
Normal Transformations

\[ n^T (v_1 - v_2) = 0 \]
\[ (Nn)^T (Lv_1 - Lv_2) = 0 \]
\[ n^T N^T L (v_1 - v_2) = 0 \]
\[ (L^T N n)^T (v_1 - v_2) = 0 \]
\[ L^T N n = \alpha n \]
\[ N = \alpha L^{-T} \]

alpha has to be positive or normals will get flipped
alpha can be different per normal, and is used to normalize transformed normal
Summary of Normal Transformations

- transform normals by inverse of the transpose of the top left 3x3 part of the model-view matrix
- normalize the result at each vertex
- normalize the result at each pixel (fragment) before computing lighting
Assignment 2:  
Start Early!