Vertex to Pixel

A brief introduction
Textbook Chapter 12
Guest Lecture by Prof. van de Panne

(some slides courtesy of Min Kim)

Rasterization

- This is part of the fixed function pipeline
- There are very clever and sophisticated algorithms underneath the hood, but most users just set a few knobs using OpenGL function calls
- We will speed through these issues for now, with the goal of getting to the fun topic of lighting asap!
- We may return to some of these issues at the end of the course, if we have time
Path from vertex to pixel

Eye coordinates (projected) $\rightarrow$ clip coordinates $\rightarrow$ normalized device coordinates (NDCs)

Dividing clip coordinates $(x_c, y_c, z_c, w_c)$ by the $w_c$ component (the fourth component in the homogeneous coordinates) yields normalized device coordinates (NDCs).

\[
\begin{bmatrix}
  x_c w_c \\
  y_c w_c \\
  z_c w_c \\
  w_c \\
\end{bmatrix}
= \frac{1}{w_c}
\begin{bmatrix}
  x_c \\
  y_c \\
  z_c \\
  1 \\
\end{bmatrix}
= \begin{bmatrix}
  s_x & 0 & -c_x & 0 \\
  0 & s_y & -c_y & 0 \\
  0 & 0 & f + n & 2fn \\
  0 & 0 & f - n & f - n \\
\end{bmatrix}
\begin{bmatrix}
  x_e \\
  y_e \\
  z_e \\
  1 \\
\end{bmatrix}
\]
Viewport matrix

- We need a transform that maps the lower left corner to \([-0.5, -0.5]\) and upper right corner to \([W - 0.5, H - 0.5]\)
- The appropriate scale and shift can be done using the viewport matrix:

\[
\begin{bmatrix}
    x_w \\
    y_w \\
    z_w \\
    1
\end{bmatrix}
= \begin{bmatrix}
    W/2 & 0 & 0 & (W-1)/2 \\
    0 & H/2 & 0 & (H-1)/2 \\
    0 & 0 & 1/2 & 1/2 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x_n \\
    y_n \\
    z_n \\
    1
\end{bmatrix}
\]

Path from vertex to pixel
Interpolation of varying variables

- Topic of Chapter 13. Optional for this course, but please remember that there is a subtle issue.
- In between the vertex and fragment shader, we need to interpolate the values of the varying variables.
- This is surprisingly subtle (called “perspective correct interpolation”).

Wrong representation of texture

When texture coordinates are linearly interpolated in window coordinates, an incorrect image results.
Correct representation of texture
Vertex shader

Object OCS

Modeling matrix

World WCS

Viewing matrix

Camera or viewing

Projection matrix

Viewport matrix

Clipping

Normalized device

Device DCS

Fixed function pipeline

Fragment shader

1000 x 1000 pixel window

\[
\begin{bmatrix}
1 & 0 & -1 \\
0 & 1 & 2 \\
1 & -6 & 1
\end{bmatrix}
\]

\[
T_M
\]

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 4 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
T_V
\]

\[
\begin{bmatrix}
1 & 0 & 0 \\
1 & 1/3 & 2/3 \\
5 & 1
\end{bmatrix}
\]

\[
T_P
\]

\[
\begin{bmatrix}
500 & 499.5 \\
500 & 499.5 \\
0.5 & 0.5 \\
1 & 1
\end{bmatrix}
\]

\[
T_{wp}
\]

Side view:

X-axis comes out of paper

X = 0 for all points in this example.