ECE 4680L/6680L

Lab #5 - triangle rendering

In this lab you are to write a C program that renders triangles. The program should read a PLY object filename and 3 rotation angles (degrees) as command line arguments. It should then set a camera position using the 3 rotation angles and render a 256 x 256 pixel image saving it in PPM format.

At the course website are several PLY object files that can be used. The program may be written under linux using gcc or under Windows using Visual C++.

The specific rendering steps include:

1. Parse the PLY file header to determine the number of vertices and faces. All other header info is irrelevant for this lab.
2. Read the PLY file vertices and faces.
3. Calculate the bounding box on the vertices. This will include the following:
   (a) Minimum and maximum X, Y and Z (two vectors denoted $\langle min \rangle$ and $\langle max \rangle$).
   (b) Center X, Y and Z (vector denoted $\langle center \rangle$).
   (c) Maximum extent of bounding box $E =$ scalar that is largest component of $\langle max - min \rangle$, i.e. largest extent of the three axes.
4. Calculate the camera position and orientation using two vectors $\langle camera \rangle$ and $\langle up \rangle$ as follows.
   (a) By default assume $\langle camera \rangle$ is $\langle 1, 0, 0 \rangle$ (positioned on the X axis) with $\langle up \rangle$ oriented as $\langle 0, 0, 1 \rangle$ (positive on the Z axis).
   (b) Rotate both the camera vector and up vector by X degrees about the X-axis, Y degrees about the Y-axis, and Z degrees about the Z-axis, where X, Y and Z were supplied as command line arguments. Rotations are calculated using equations 2-3.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix} \quad (1)$$

$$R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \quad (2)$$

$$R_z(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

   (c) Move and scale the camera vector according to equation 4.

$$\langle \text{camera} \rangle = 1.5E\langle \text{camera} \rangle + \langle \text{center} \rangle \quad (4)$$

5. Determine the 3D coordinates bounding the image using equations 5-12.

$$\begin{aligned}
\langle left\rangle &= \langle up\rangle \times \langle center - camera\rangle && (5)\\
a &= \|\langle left\rangle\| && (6)\\
\langle left\rangle &= \frac{E}{2a}\langle left\rangle + \langle center\rangle && (7)\\
\langle right\rangle &= (\langle center - camera\rangle \times \langle up\rangle && (8)\\
\langle right\rangle &= \frac{E}{2a}\langle right\rangle + \langle center\rangle && (9)\\
\langle top\rangle &= \frac{E}{2}\langle up\rangle + \langle center\rangle && (10)\\
\langle bottom\rangle &= \frac{-E}{2}\langle up\rangle + \langle center\rangle && (11)\\
\langle topleft\rangle &= \frac{E}{2}\langle up\rangle + \langle left\rangle && (12)
\end{aligned}$$

6. For each pixel r, c in the image:

   (a) Default image color is black (greyscale=0).

   (b) Default z-buffer depth is very far (for example, 999999). Note the z-buffer image must be floats or doubles. Recall it stores the distance to the closest triangle for each pixel so that only the color for that triangle is drawn.

   (c) Calculate vector coordinates $\langle image\rangle$ for the image pixel using equation 13, where $COLS$ and $ROWS$ are the width and height of the image in pixels and it is assumed c and r index from 0 to $COLS - 1$ and $ROWS - 1$ respectively.

$$\begin{aligned}
\langle image\rangle &= \langle topleft\rangle + \frac{c}{COLS-1}\langle right - left\rangle + \\
&\quad \frac{r}{ROWS-1}\langle bottom - top\rangle && (13)
\end{aligned}$$

   (d) For each triangle having coordinates v_0, v_1 and v_2:

      i. Find the plane equation $\langle A, B, C, D\rangle$ that contains the triangle using equations 14-15.

$$\begin{aligned}
\langle A, B, C\rangle &= \langle v_1 - v_0\rangle \times \langle v_2 - v_0\rangle && (14)\\
D &= -\langle A, B, C\rangle \cdot \langle v_0\rangle && (15)
\end{aligned}$$

      ii. Find the distance along the image pixel ray to the triangle, denoted $\frac{n}{d}$, using equations 16-17. Test if ray is parallel to triangle (if d is near zero), and if so skip this triangle for this pixel.

$$\begin{aligned}
n &= -\langle A, B, C\rangle \cdot \langle camera\rangle - D && (16)\\
d &= \langle A, B, C\rangle \cdot \langle image - camera\rangle && (17)
\end{aligned}$$

iii. Find the 3D coordinates $\langle intersect \rangle$ of ray and plane using equation 18.

$$\langle intersect \rangle = \langle camera \rangle + \frac{n}{d} \langle image - camera \rangle \tag{18}$$

iv. Determine if intersection point lies within triangle by calculating the three dot products in equations 19-21.

$$\begin{aligned} dot1 &= \langle v_2 - v_0 \rangle \times \langle v_1 - v_0 \rangle \cdot \langle intersect - v_0 \rangle \times \langle v_1 - v_0 \rangle & (19) \\ dot2 &= \langle v_0 - v_1 \rangle \times \langle v_2 - v_1 \rangle \cdot \langle intersect - v_1 \rangle \times \langle v_2 - v_1 \rangle & (20) \\ dot3 &= \langle v_1 - v_2 \rangle \times \langle v_0 - v_2 \rangle \cdot \langle intersect - v_2 \rangle \times \langle v_0 - v_2 \rangle & (21) \end{aligned}$$

v. If any of the dot products is less than zero (if $dot1 < 0$ or $dot2 < 0$ or $dot3 < 0$), then the intersection point lies outside the triangle and it can be skipped.

vi. If the distance to the triangle n/d is greater than the current z-buffer value for this pixel, then the triangle lies behind a closer triangle and it can be skipped.

vii. Set pixel color to $155 + (i\%100)$ where i is the index of the triangle. This provides some variation in colors making the object easier to see.

7. Write PPM image.

Test the program using several different rotations (e.g. $90, 45, -135$) on several of the given PLY files. You may want to compile with optimization to speed up program execution.

All students are to complete the lab individually.

This lab is due by the due date given at the course website. Grading will be determined via demonstration. The lab TA will be available for demonstrations in the lab (Riggs 309) at the times posted at the course website. If you need to arrange an alternate demonstration time, work it out with the TA.

You must also submit your C-code canvas. The due date is posted on the course website.

Work for this lab must be completed independently by each student. If it is determined that a piece of work has been copied, all parties involved will receive zero credit. If it happens twice, the offending parties will fail the course. Please protect your work!