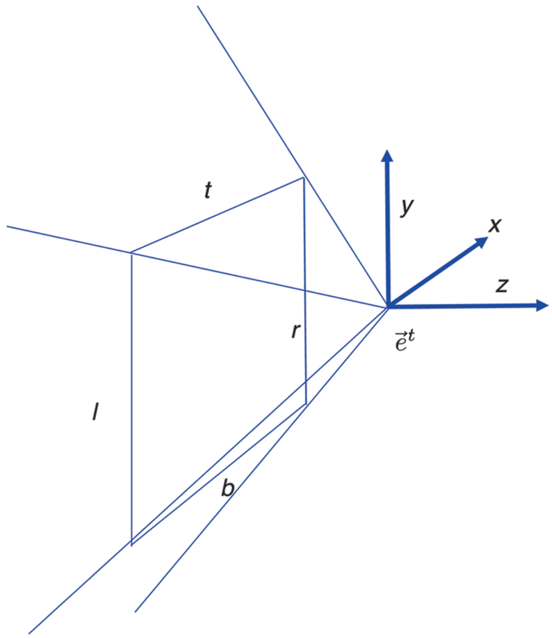


Figure 10.9
The 3D frustum is defined by specifying an image rectangle on the near plane.

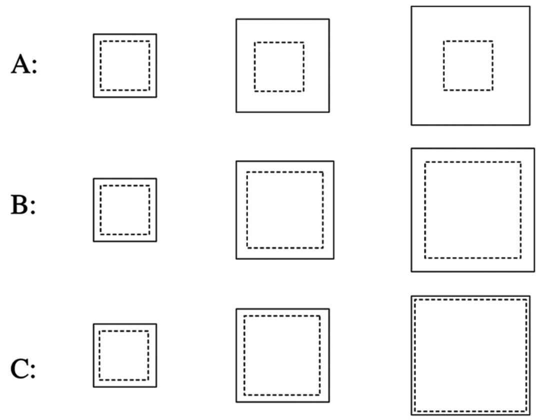
axis, but this can be done as an appropriate camera rotation in the original definition of the $\vec{\mathbf{e}}^t$ frame!

10.4 Context

The projection operation we have described is a mapping that can be applied to any point given in eye coordinates to obtain its normalized device coordinates. In OpenGL, though, this mapping is only applied to the vertices of a triangle. Once the normalized device coordinates of the three vertices of a triangle are obtained, the interior points are simply filled in by computing all of the pixels that fall in the interior of the triangle on the image plane.

Exercises

10.1 Suppose we take a picture of an ice cube. Let us draw the projection of the front face with a solid outline and the rear face with a dotted outline. Suppose three images are taken using fields of view of 40, 30, and 20 degrees, respectively. All other camera parameters remain fixed. Which of the following three sequences is plausible:



10.2 (Tricky) Look at the following two photographs (from [26], copyright Phillip Greenspun):

Can you figure out how these were taken? What is the difference in the eye coordinates systems and projection matrices used? (Hint: Remember that we can build a camera with a shifted film plane.)

10.3 (Tricky) Let $\vec{\mathbf{e}}^t = \vec{\mathbf{w}}^t E$ and let P be a camera matrix that is just some arbitrary 4 by 4 matrix with "dashes" in its third row. Given the world coordinates of six points in space as well as their normalized device coordinates, how would one compute the 12 unknown entries in the matrix PE^{-1}? (Note: This can only be solved up to an unknown scale factor.) (Hint: Set up an appropriate homogeneous linear system with a right-hand side of zero.)

```
Matrix4 makeProjection(double top, double bottom, double
left, double right, double zNear, double ZFar)
```

that return the appropriate projection matrices. We also need the procedure

```
sendProjectionMatrix(projmat)
```

to send the projection matrix to the appropriately named matrix variable in the vertex shader. The code for these procedures is found on the book's website. The actual multiplication with the projection matrix is done in the vertex shader as we did in section 6.3.

Exercises

11.1 Suppose we have two triangles such that the closest-in-z vertex of one triangle is farther than the farthest-in-z vertex of the other triangle? If we linearly interpolated the z_e value over the interior of a triangle, would z-buffering produce a picture with the correct occlusion?

11.2 Starting from the projection matrix P of equation (11.2), suppose we replace P with PQ where

$$Q = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

What will be the effect in our resulting image?

11.3 Suppose we replace P with PS where

$$S = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}.$$

What will be the effect in our resulting image?

11.4 Suppose we replace P with QP. What will be the effect in our resulting image?

−1
0
1

drawn twice, this can cause problems if we are modeling transparency using alpha blending (see section 16.4). Boundary rules must be implemented carefully to ensure that such a pixel is only drawn once by this pair of triangles.

A related topic to triangle rasterization is line/curve rasterization. In this case, we are not looking for pixels "inside" of a 2D shape such as a triangle, but instead pixels "near" a 1D shape. Thus, even defining the problem is a bit trickier, and it is one that we will not delve into here. The interested reader can consult [5].

Exercises

12.1 Assume that in the vertex shader of section 6.3 we added the following line at the end: `gl_Position = gl_Position/gl_Position.w;`. What impact would this have on clipping?

12.2 What is the formula for finding the $[x_t, y_t]^t$ abstract texture coordinates (in the unit square) of a texture pixel with integer texture-pixel index of (i, j).

12.3 Starting from the viewport matrix V from equation (12.5), suppose we replace V with QV where

$$Q = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

What will be the effect in our resulting image?

12.4 How can one use equation (B.2) in appendix B to obtain the coefficients of an edge function for use in rasterization?