calculate the desired field of view.)

The preceding paragraph mentions inches and millimeters - do these really have anything to do with OpenGL? The answer is, in a word, no. The projection and other transformations are inherently unitless. If you want to think of the near and far clipping planes as located at 1.0 and 20.0 meters, inches, kilometers, or leagues, it's up to you. The only rule is that you have to use a consistent unit of measurement. Then the resulting image is drawn to scale.

Orthographic Projection

With an orthographic projection, the viewing volume is a rectangular parallelepiped, or more informally, a box (see Figure 3-15). Unlike perspective projection, the size of the viewing volume doesn't change from one end to the other, so distance from the camera doesn't affect how large an object appears. This type of projection is used for applications such as creating architectural blueprints and computer-aided design, where it's crucial to maintain the actual sizes of objects and angles between them as they're projected.

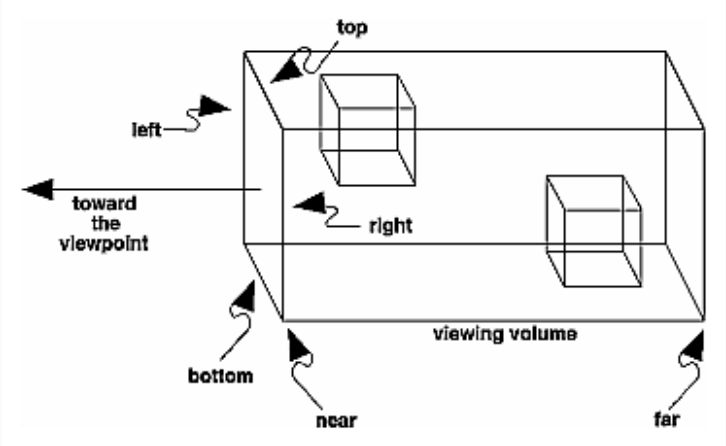

Figure 3-15 : Orthographic Viewing Volume

The command **glOrtho()** creates an orthographic parallel viewing volume. As with **glFrustum()**, you specify the corners of the near clipping plane and the distance to the far clipping plane.

*void **glOrtho**(GLdouble left, GLdouble right, GLdouble bottom,*
GLdouble top, GLdouble near, GLdouble far);

> *Creates a matrix for an orthographic parallel viewing volume and multiplies the current matrix by it. (left, bottom, -near) and (right, top, -near) are points on the near clipping plane that are mapped to the lower-left and upper-right corners of the viewport window, respectively. (left, bottom, -far) and (right, top, -far) are points on the far clipping plane that are mapped to the same respective corners of the viewport. Both near and far can be positive or negative.*

With no other transformations, the direction of projection is parallel to the z-axis, and the viewpoint faces

toward the negative z-axis. Note that this means that the values passed in for far and near are used as negative z values if these planes are in front of the viewpoint, and positive if they're behind the viewpoint.

For the special case of projecting a two-dimensional image onto a two-dimensional screen, use the Utility Library routine **gluOrtho2D**(). This routine is identical to the three-dimensional version, **glOrtho**(), except that all the z coordinates for objects in the scene are assumed to lie between -1.0 and 1.0. If you're drawing two-dimensional objects using the two-dimensional vertex commands, all the z coordinates are zero; thus, none of the objects are clipped because of their z values.

*void **gluOrtho2D**(GLdouble left, GLdouble right,*
GLdouble bottom, GLdouble top);

> *Creates a matrix for projecting two-dimensional coordinates onto the screen and multiplies the current projection matrix by it. The clipping region is a rectangle with the lower-left corner at (left, bottom) and the upper-right corner at (right, top).*

Viewing Volume Clipping

After the vertices of the objects in the scene have been transformed by the modelview and projection matrices, any primitives that lie outside the viewing volume are clipped. The six clipping planes used are those that define the sides and ends of the viewing volume. You can specify additional clipping planes and locate them wherever you choose. (See "Additional Clipping Planes" for information about this relatively advanced topic.) Keep in mind that OpenGL reconstructs the edges of polygons that get clipped.

Viewport Transformation

Recalling the camera analogy, you know that the viewport transformation corresponds to the stage where the size of the developed photograph is chosen. Do you want a wallet-size or a poster-size photograph? Since this is computer graphics, the viewport is the rectangular region of the window where the image is drawn. Figure 3-16 shows a viewport that occupies most of the screen. The viewport is measured in window coordinates, which reflect the position of pixels on the screen relative to the lower-left corner of the window. Keep in mind that all vertices have been transformed by the modelview and projection matrices by this point, and vertices outside the viewing volume have been clipped.