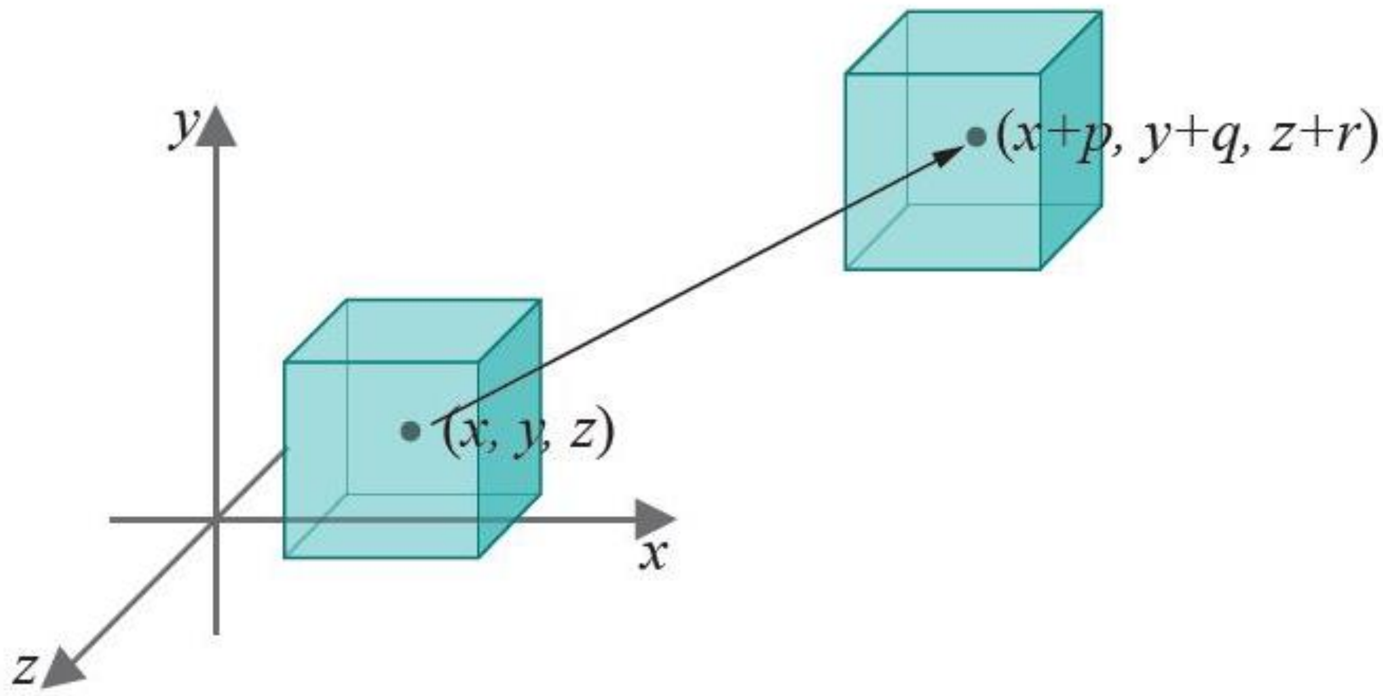
COMPUTER GRAPHICS

Modeling Transformation of OpenGL

Dr. Zhonggui Chen
School of Informatics, Xiamen University
http://graphics.xmu.edu.cn

Translation

- glTranslatef(p, q, r)



Translation

- glutWireCube(5.0)
- glFrustum(-5.0,5.0, -5.0, 5.0, 5.0, 100.0)
- glTranslatef(0, 0, -15)

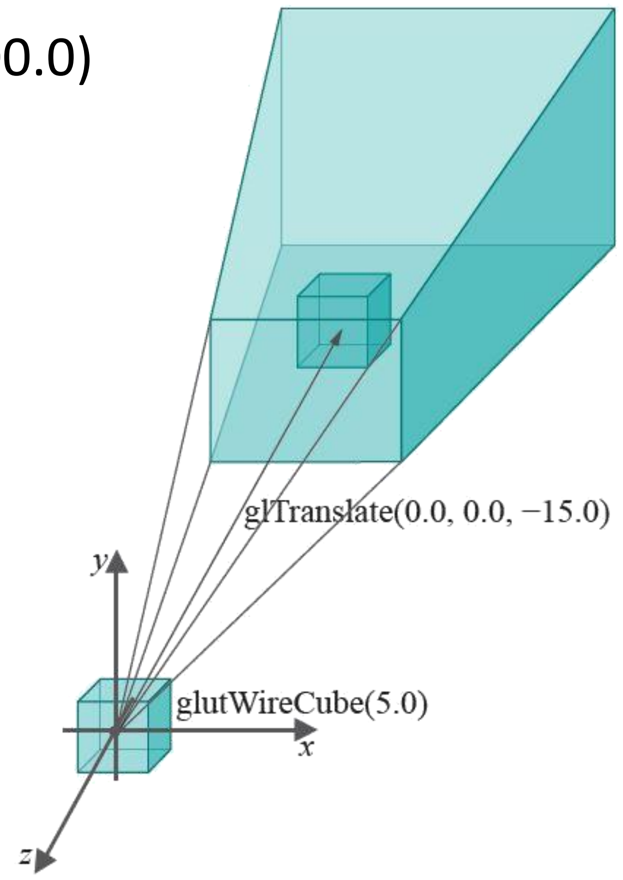
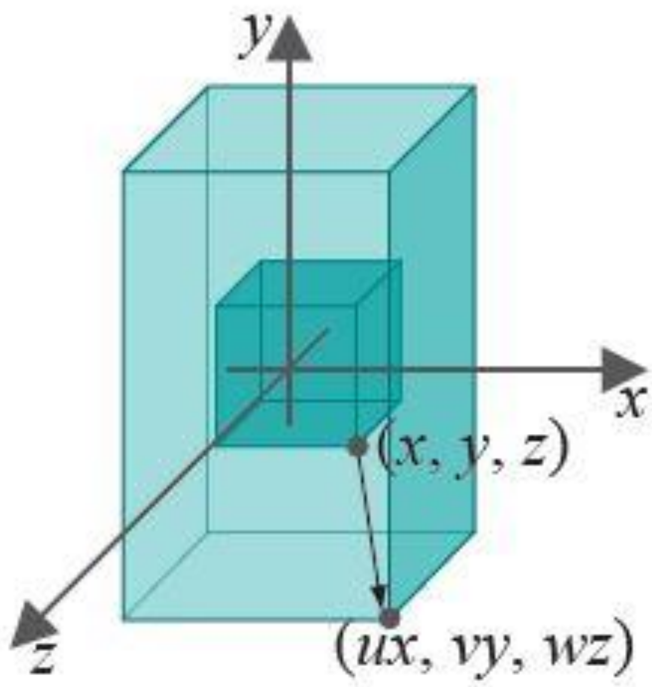


Figure 4.3: Translating into the viewing frustum.

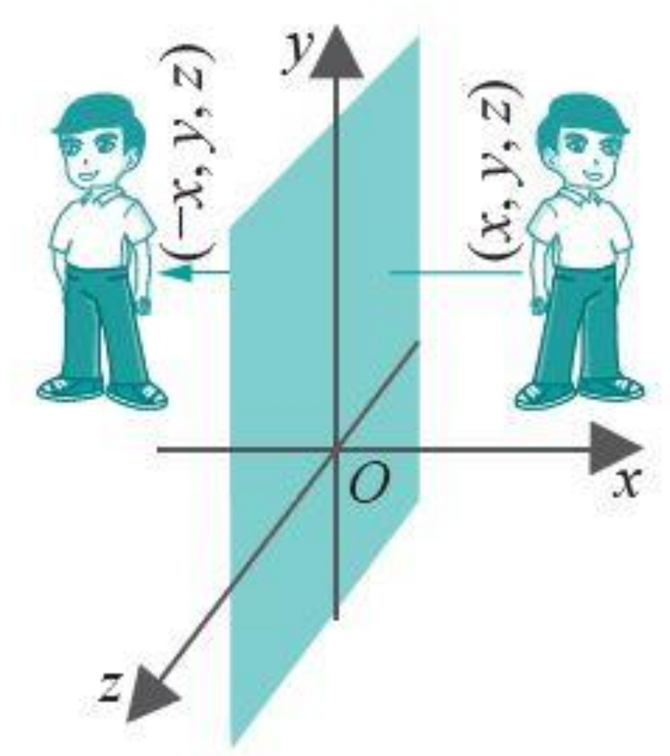
Scaling

- glScalef(u , v , w)



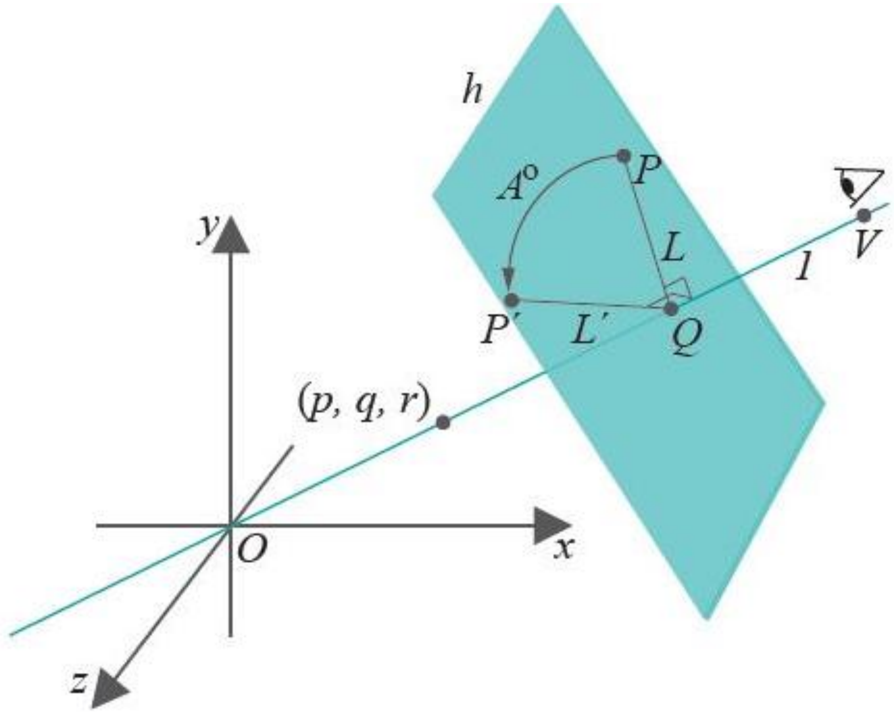
Scaling

- glScalef(-1 , 1 , 1) – Reflection in the yz-plane



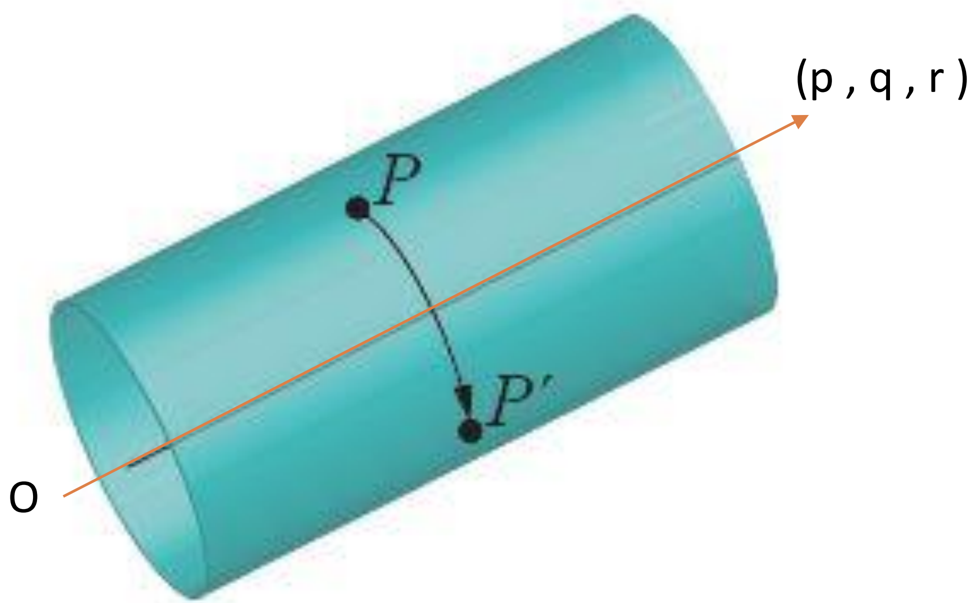
Rotation

- glRotatef(A , p , q , r)
  - Rotate an object about an axis from the origin O to the point (p,q,r)
  - The amount of rotation is A°
  - Measured counter-clockwise looking from (p,q,r) to the origin



Rotation

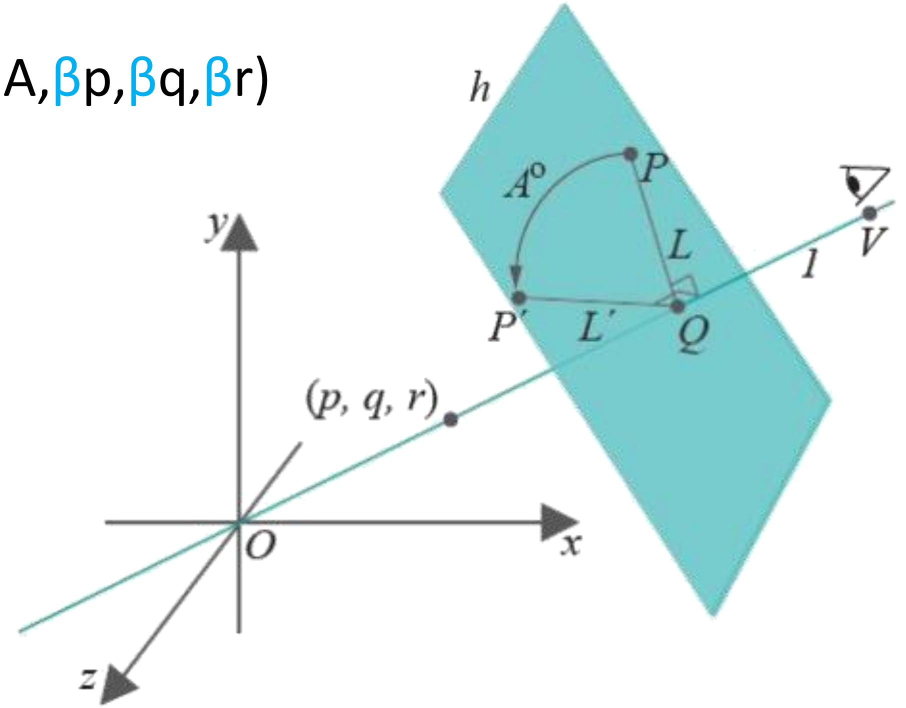
- glRotatef(A , p , q , r)
- Intuitive explanation: point turning along an imaginary cylinder on that axis from the origin O to the point (p,q,r)



Rotation

- glRotatef(A,p,q,r) = glRotatef(A,αp,αq,αr)
  where α is any positive scalar

- glRotatef(−A,p,q,r) = glRotatef(A,βp,βq,βr)
  where β is any negative scalar



Composing Modeling Transformations

- Example box.cpp
- Apply two modeling transformations to the box:

```
// Modeling transformations.
glTranslatef(0.0, 0.0, -15.0);
glRotatef(30.0, 1.0, 0.0, 0.0);

glutWireCube(5.0); // Box.
```

Composing Modeling Transformations

- A vertex V is represented in OpenGL as a 4 $\times$ 1 column matrix

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- OpenGL maintains a 4 $\times$ 4 **modelview** matrix, call it M , which initially is the identity

$$M = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{24} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Composing Modeling Transformations

- The matrix of each successive modeling transformation is multiplied from the left by the current modelview matrix

```
                               // M = I, initially
modelingTransformation 1;      // M = I M_1 = M_1
modelingTransformation 2;      // M = M_1 M_2
modelingTransformation 3;      // M = M_1 M_2 M_3
...
modelingTransformation n-1;    // M = M_1 M_2 ... M_{n-1}
modelingTransformation n;      // M = M_1 M_2 ... M_{n-1} M_n
object;
```

Composing Modeling Transformations

- multiply the object's vertices *V* from the left by the current modelview matrix *M*:

$$V \mapsto MV = (M_1 M_2 \ldots M_{n-1} M_n)V$$

$$= \quad M_1(M_2(\ldots M_{n-1}(M_n V)\ldots))$$

Placing Multiple Objects

- Example:
  - Replace the entire display routine of the original box.cpp with:

```
void drawScene(void)
{
        glClear(GL COLOR BUFFER BIT);
        glColor3f(0.0, 0.0, 0.0);
        glLoadIdentity();
        // Modeling transformations.
        glTranslatef(0.0, 0.0, -15.0);
        glTranslatef(5.0, 0.0, 0.0);
        glutWireCube(5.0); // Box.

        //More modeling transformations.
        glTranslatef(0.0, 10.0, 0.0);
        glutWireSphere(2.0, 10, 8); // Sphere.
        glFlush();
}
```