计算机图形学（实验二）

实验课内容

- OpenGL坐标系
- 投影变换

OpenGL的坐标系

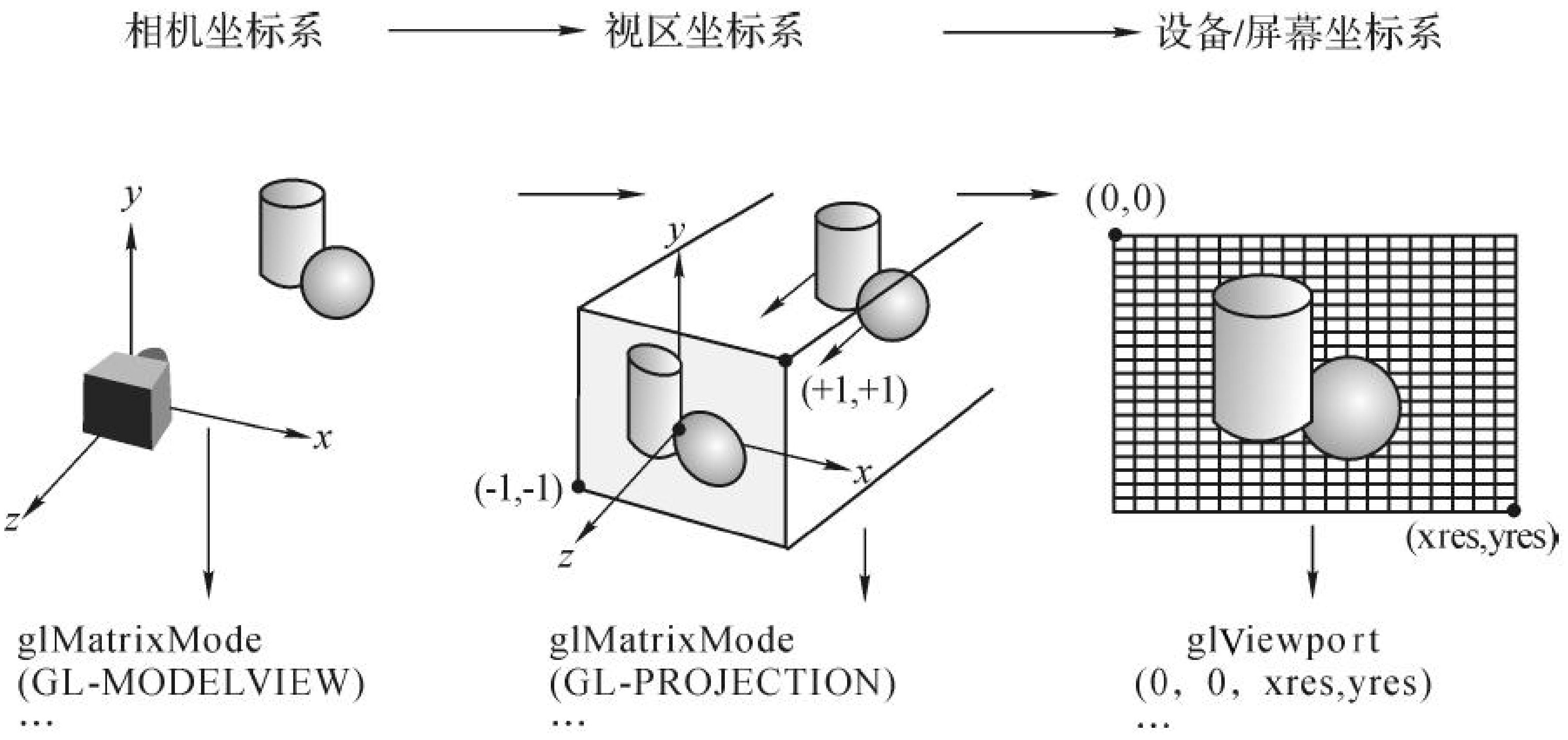
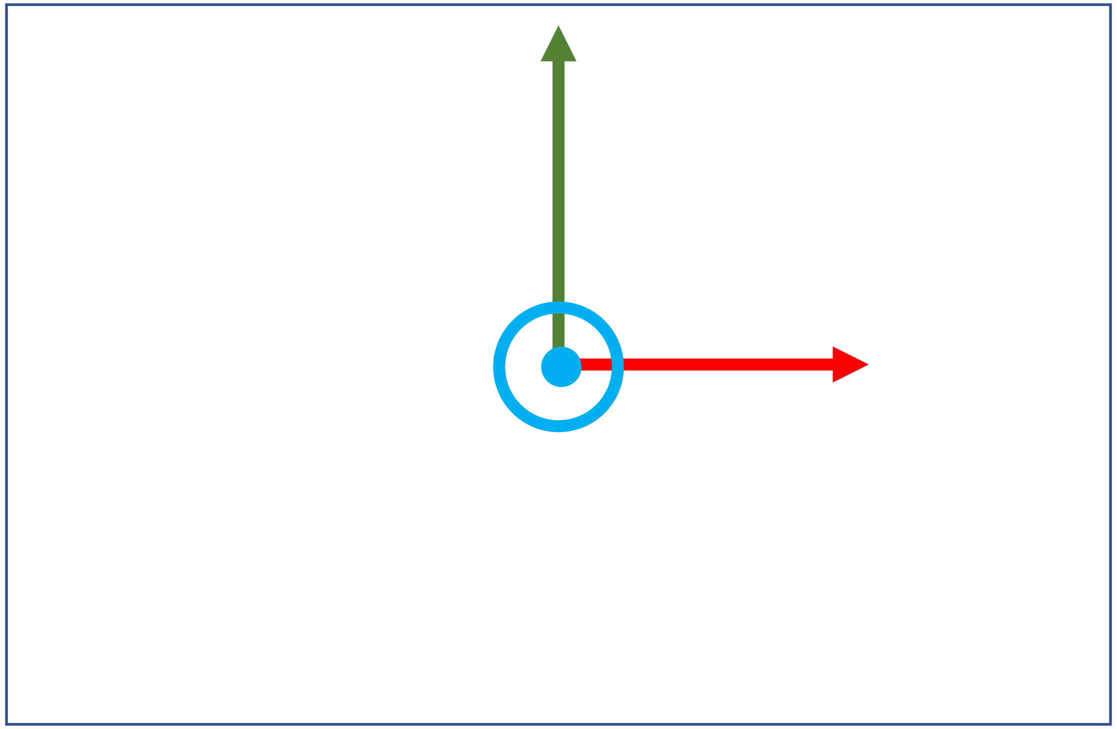


图 3.2 OpenGL 三维绘制流水线

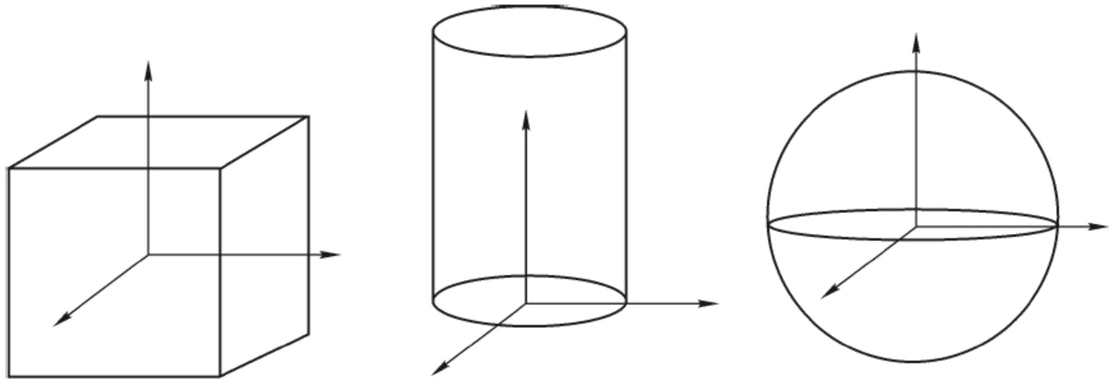
OpenGL的坐标系

- 初始时：
  - 相机坐标系与世界坐标系重合
  - 摄像机向右为X正方向
  - 摄像机向上为Y正方向
  - 摄像机向前为Z负方向



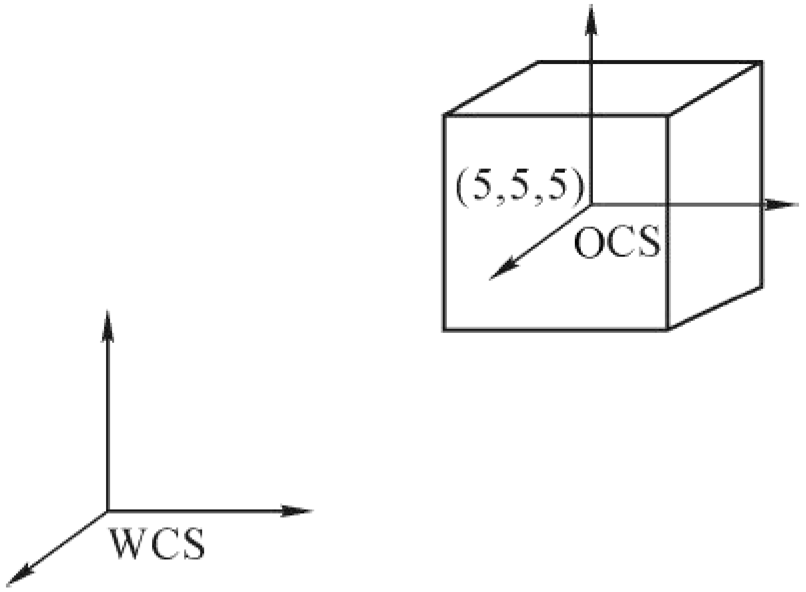
OpenGL的坐标系

- 使用glutWireCube()绘制立方体
  - 对象坐标系（Object Coordinate System, OCS）定义模型
  - 需要将立方体平移到世界坐标系（World Coordinate System, WCS）
- 相应函数
  - 旋转glRotatef(angle,vx,vy,vz)
  - 平移glTranslate(dx,dy,dz)
  - 缩放glScalef(sx,sy,sz)



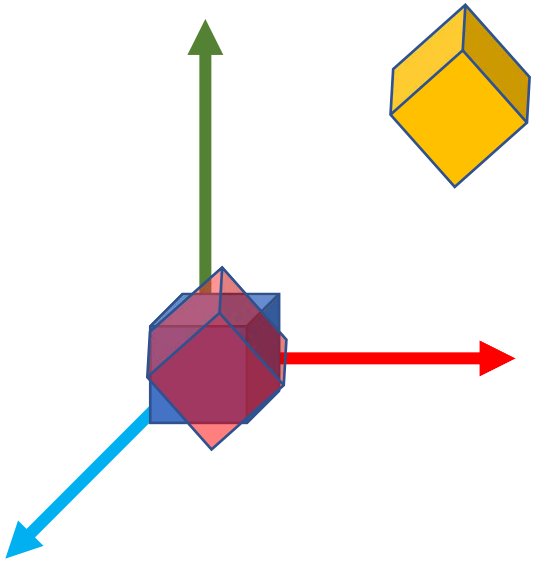
立方体的对象坐标系　　圆柱体的对象坐标系　　球体的对象坐标系

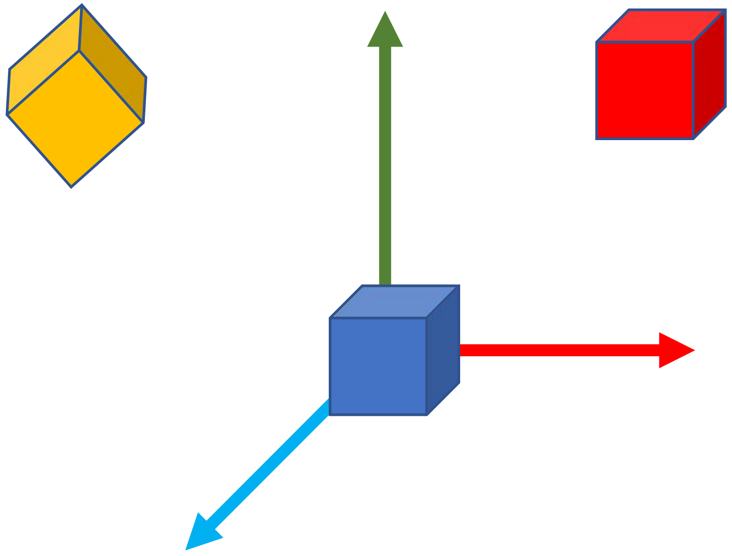
图 3.3　对象坐标系



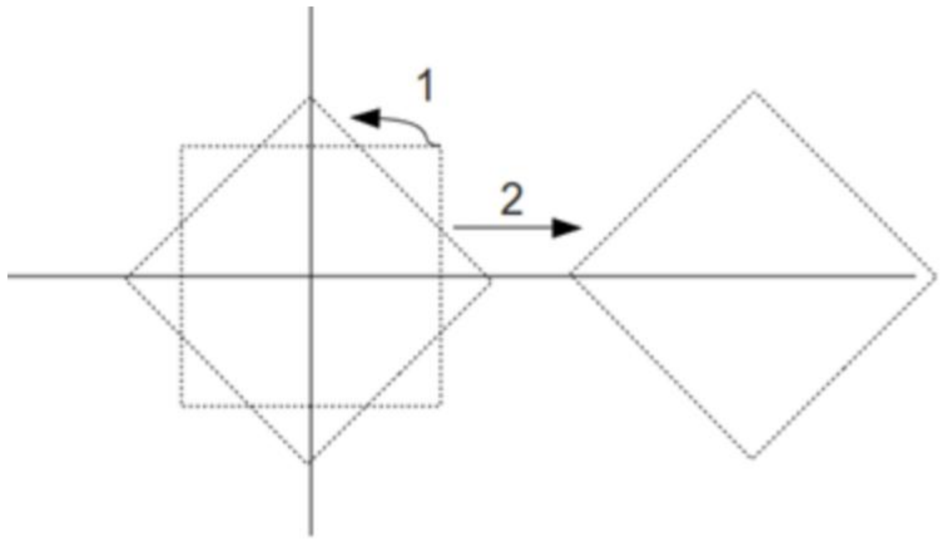
OpenGL的坐标系

- 旋转变换和缩放都是以坐标系原点为中心进行，最后指定的变换最先执行，即后进先出
- 通常我们在变换时先缩放，再旋转，最后平移（**代码顺序：先平移，再旋转，最后写缩放**）：

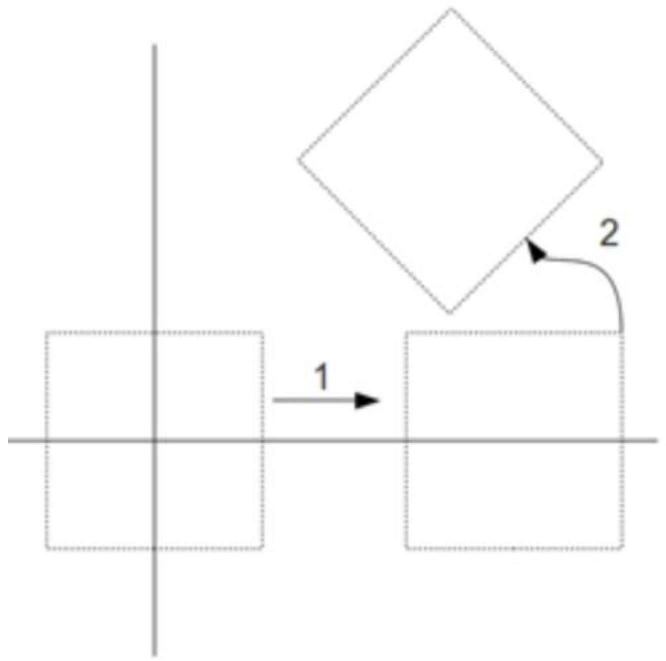




OpenGL的坐标系



• 先旋转，后平移



• 先平移，后旋转

OpenGL的坐标系

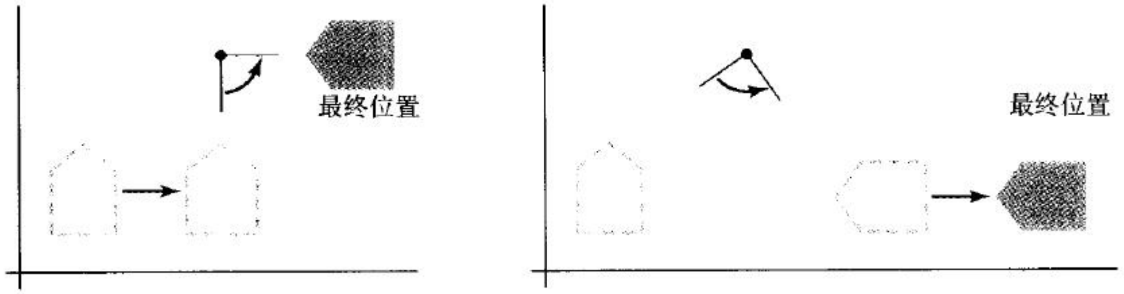
- 数学证明辅助：
  - 计算机图形学（第三版）p188-242
  - 计算机图形学课程设计 第四章p57-63

矩阵相乘符合结合律。对于任何三个矩阵 $\mathbf{M}_1$、$\mathbf{M}_2$ 和 $\mathbf{M}_3$，矩阵积 $\mathbf{M}_3 \cdot \mathbf{M}_2 \cdot \mathbf{M}_1$ 可先将 $\mathbf{M}_3$ 和 $\mathbf{M}_2$ 相乘或先将 $\mathbf{M}_2$ 和 $\mathbf{M}_1$ 相乘：

$$\mathbf{M}_3 \cdot \mathbf{M}_2 \cdot \mathbf{M}_1 = (\mathbf{M}_3 \cdot \mathbf{M}_2) \cdot \mathbf{M}_1 = \mathbf{M}_3 \cdot (\mathbf{M}_2 \cdot \mathbf{M}_1) \tag{5.40}$$

因此，依靠变换的描述次序，我们既可以使用从左到右（前乘）、也可以使用从右到左（后乘）的结合分组来求矩阵乘积。有些图形软件包要求变换按应用的次序描述。在这种情况下，我们先引入变换 $\mathbf{M}_1$，然后 $\mathbf{M}_2$，最后 $\mathbf{M}_3$。在每一个连续的变换子程序被调用时，其矩阵从左边与前面的矩阵乘积合并。而另一些图形系统是后乘矩阵，因此该变换序列按相反次序引入：最后引入的变换（本例中是 $\mathbf{M}_1$）是最先应用的，而第一个被调用的变换（此时是 $\mathbf{M}_3$）是最后应用的。

另一方面，变换积一般不可交换，矩阵积 $\mathbf{M}_2 \cdot \mathbf{M}_1$ 不等于 $\mathbf{M}_1 \cdot \mathbf{M}_2$。这说明如果要平移和旋转对象，必须注意复合矩阵求值的顺序（参见图 5.13）。对于变换序列中每一个类型都相同的特殊情况，变换矩阵的多重相乘是可交换的。例如，两个连续的旋转可以按两种顺序完成，但其最后位置是相同的。这种交换特性对两个连续的平移或两个连续缩放也同样适用。另一对可交换操作是旋转和一致缩放（$s_x = s_y$）。



(a) (b)

图 5.13 改变变换序列的顺序将影响对象的变换位置。在(a)中对象先平移后旋转，在(b)中对象先旋转后平移

OpenGL的坐标系

- 数学证明辅助：
  - 计算机图形学（第三版）p188-242
  - 计算机图形学课程设计第四章p57-63

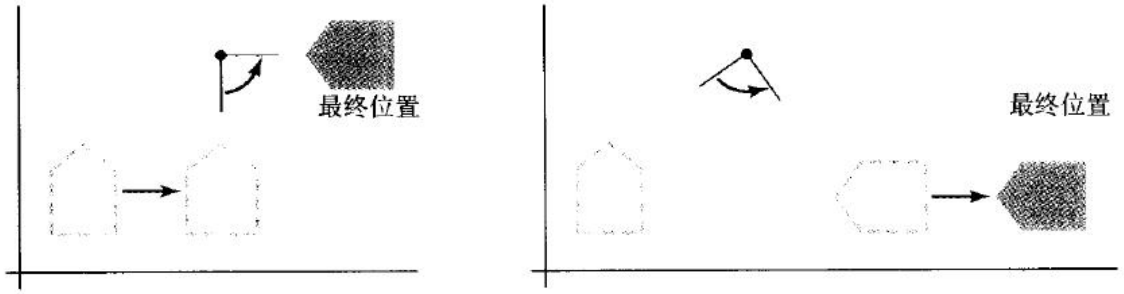
glRotatef(45.0f, 0.0f, 1.0f, 0.0f);
glTranslatef(2.0f, 0.0f, 0.0f);
Result ＝ Mr＊（Mt＊[x,y,z]T）

矩阵相乘符合结合律。对于任何三个矩阵 $\mathbf{M}_1$、$\mathbf{M}_2$ 和 $\mathbf{M}_3$，矩阵积 $\mathbf{M}_3 \cdot \mathbf{M}_2 \cdot \mathbf{M}_1$ 可先将 $\mathbf{M}_3$ 和 $\mathbf{M}_2$ 相乘或先将 $\mathbf{M}_2$ 和 $\mathbf{M}_1$ 相乘：

$$\mathbf{M}_3 \cdot \mathbf{M}_2 \cdot \mathbf{M}_1 = (\mathbf{M}_3 \cdot \mathbf{M}_2) \cdot \mathbf{M}_1 = \mathbf{M}_3 \cdot (\mathbf{M}_2 \cdot \mathbf{M}_1) \tag{5.40}$$

因此，依靠变换的描述次序，我们既可以使用从左到右（前乘）、也可以使用从右到左（后乘）的结合分组来求矩阵乘积。有些图形软件包要求变换按应用的次序描述。在这种情况下，我们先引入变换 $\mathbf{M}_1$，然后 $\mathbf{M}_2$，最后 $\mathbf{M}_3$。在每一个连续的变换子程序被调用时，其矩阵从左边与前面的矩阵乘积合并。而另一些图形系统是后乘矩阵，因此该变换序列按相反次序引入：最后引入的变换（本例中是 $\mathbf{M}_1$）是最先应用的，而第一个被调用的变换（此时是 $\mathbf{M}_3$）是最后应用的。

另一方面，变换积一般不可交换，矩阵积 $\mathbf{M}_2 \cdot \mathbf{M}_1$ 不等于 $\mathbf{M}_1 \cdot \mathbf{M}_2$。这说明如果要平移和旋转对象，必须注意复合矩阵求值的顺序（参见图 5.13）。对于变换序列中每一个类型都相同的特殊情况，变换矩阵的多重相乘是可交换的。例如，两个连续的旋转可以按两种顺序完成，但其最后位置是相同的。这种交换特性对两个连续的平移或两个连续缩放也同样适用。另一对可交换操作是旋转和一致缩放（$s_x = s_y$）。



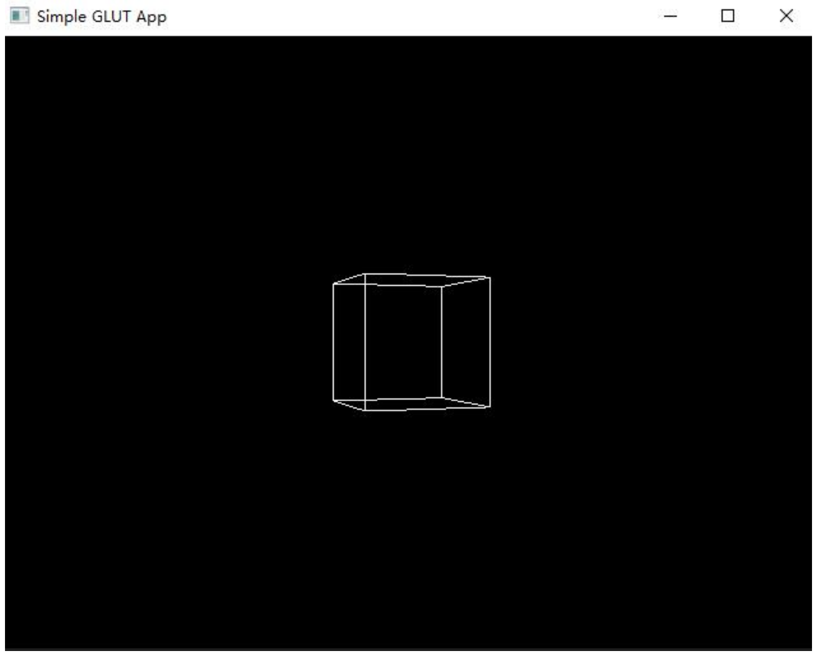
(a) (b)

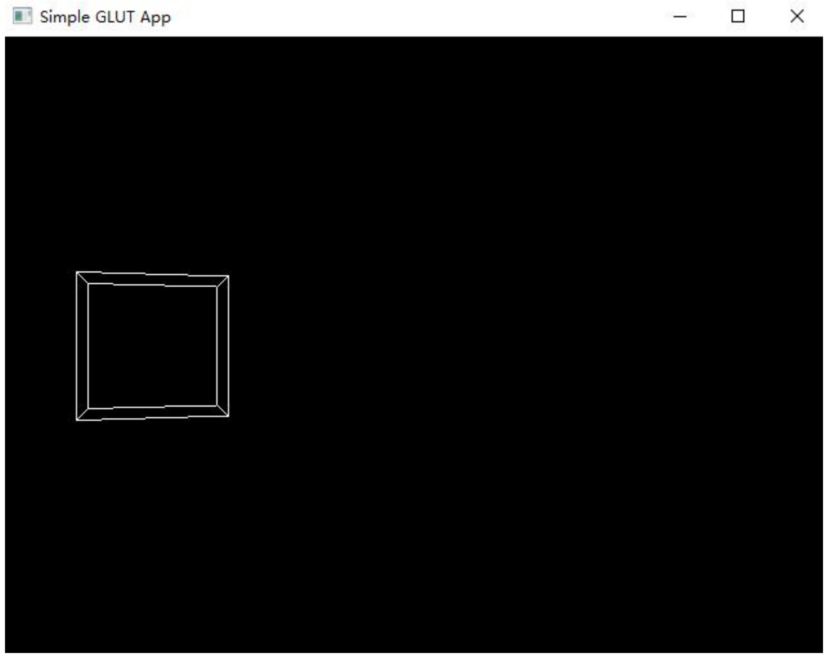
图 5.13　改变变换序列的顺序将影响对象的变换位置。在(a)中对象先平移后旋转，在(b)中对象先旋转后平移

OpenGL的坐标系

```
glPushMatrix();
glTranslatef(0.0f, 0.0f, -6.0f);
glRotatef(19.198f, 0.0f, 1.0f, 0.0f);
glutWireCube(1.0);
glPopMatrix();
```

```
glPushMatrix();
glRotatef(19.198f, 0.0f, 1.0f, 0.0f);
glTranslatef(0.0f, 0.0f, -6.0f);
glutWireCube(1.0);
glPopMatrix();
```



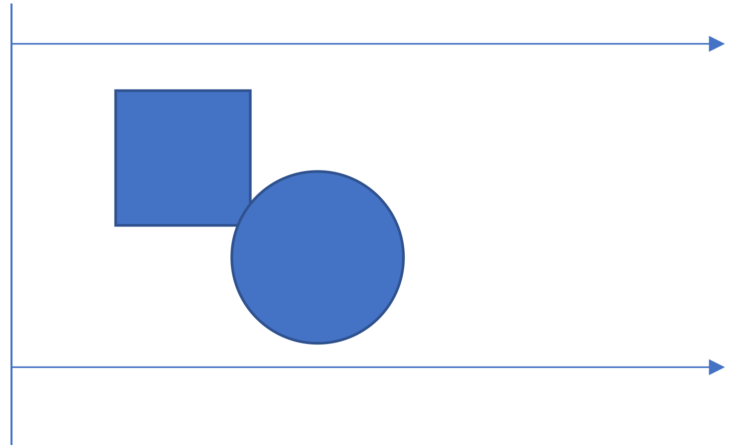


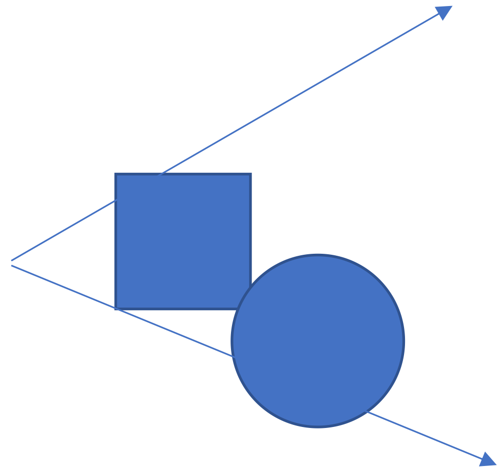
OpenGL坐标系

- 应用：
  - 简单的三维动画
  - 设置一个随时间变化的空间变换（如x=sin(t)）
  - 在绘制函数的最后，每次绘制完成后更改参数（如t=t+1）
  - 利用双缓冲实现动画效果

投影变换

- 投影有两种：
  - 正投影
    - 即平行投影
    - 平行线投影后也是平行线
  - 透视投影
    - 中心投影
    - 不与视线方向垂直的平行线有消失点





投影变换

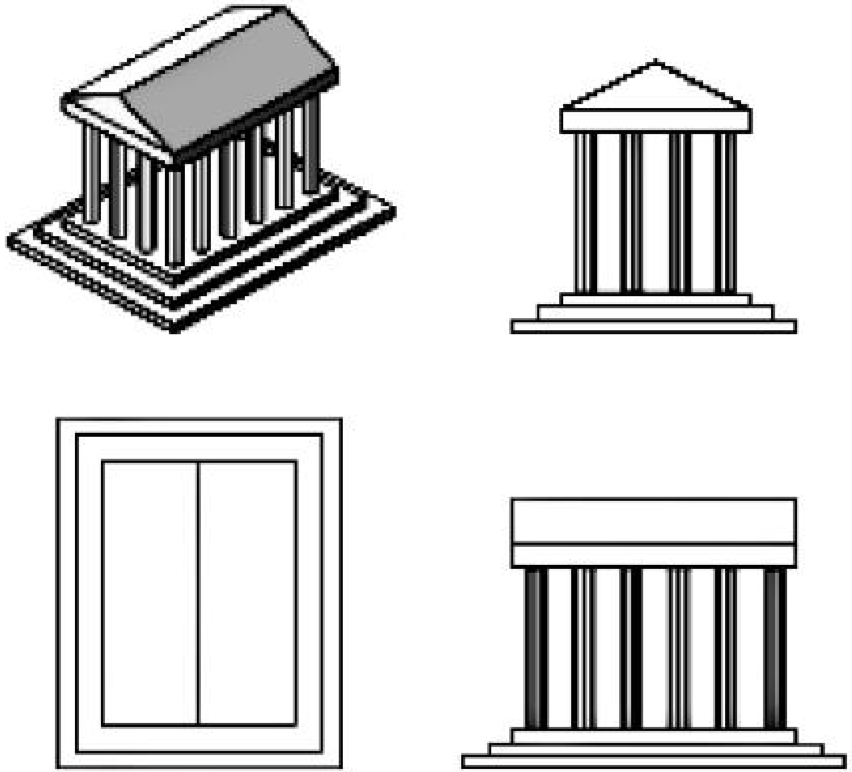


图 3.7　正投影示例

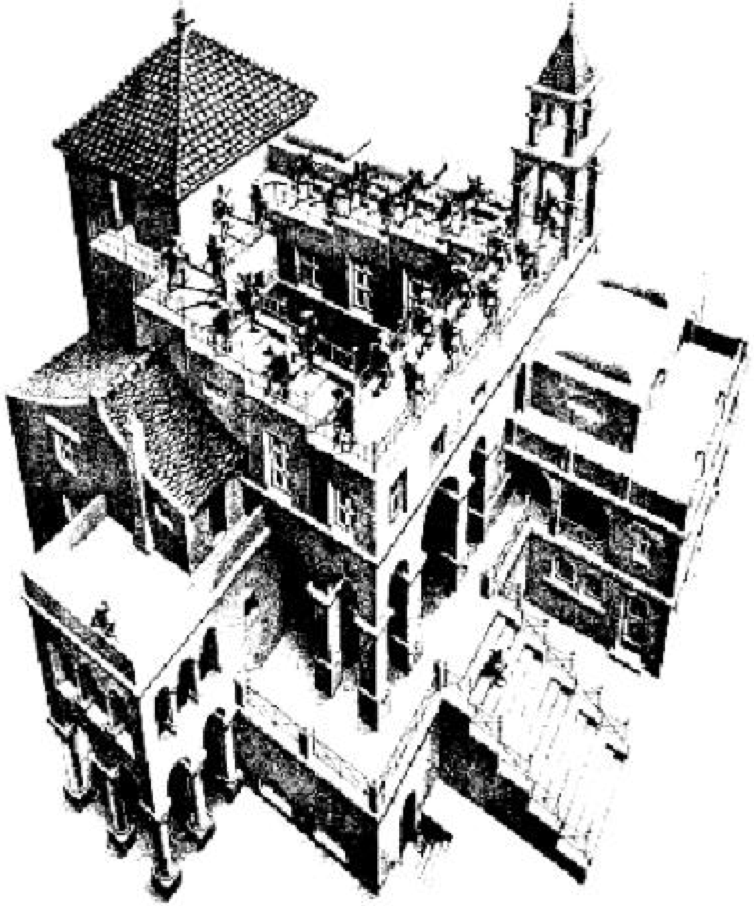


图 3.8　透视投影示例

投影变换

- 正投影
  - glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);
  - gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);

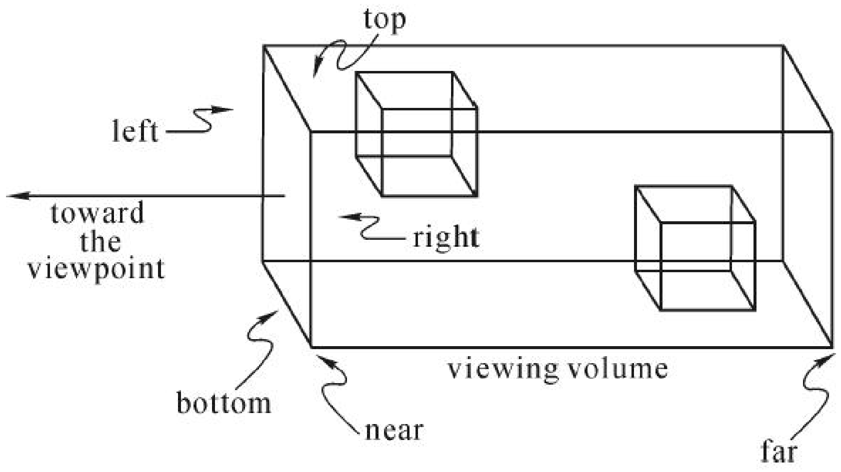


图 3.9　函数 glOrtho()的参数：定义了三维空间中的一个盒子，落在其内部的几何体将被正投影成像

投影变换

- 正投影矩阵

$$\mathbf{M}_{\text{orth}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

投影变换

- 透视投影
  - gluPerspective(GLdouble fovy, GLdouble aspect,
  - GLdouble near, GLdouble far);

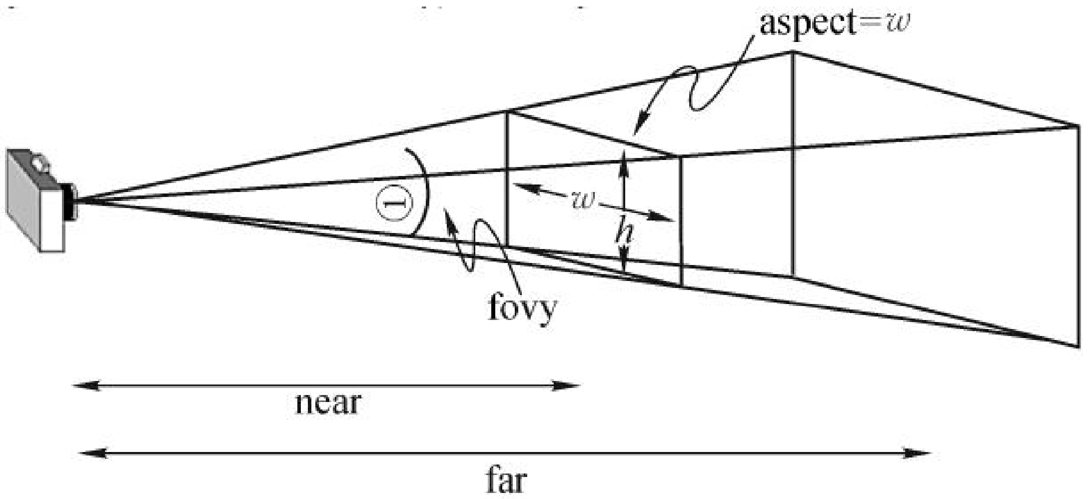
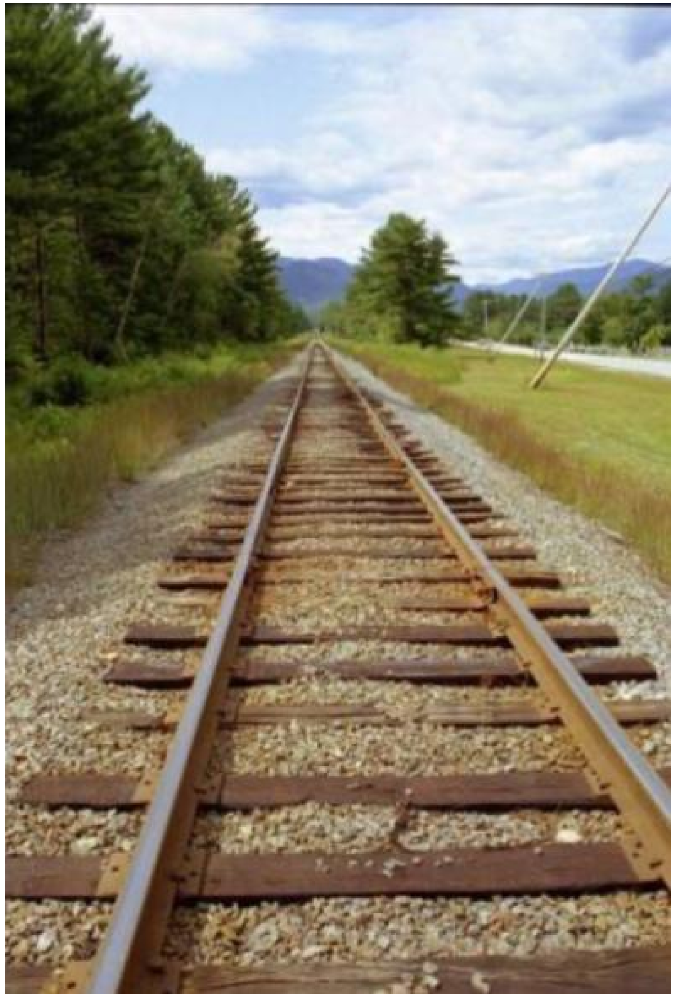


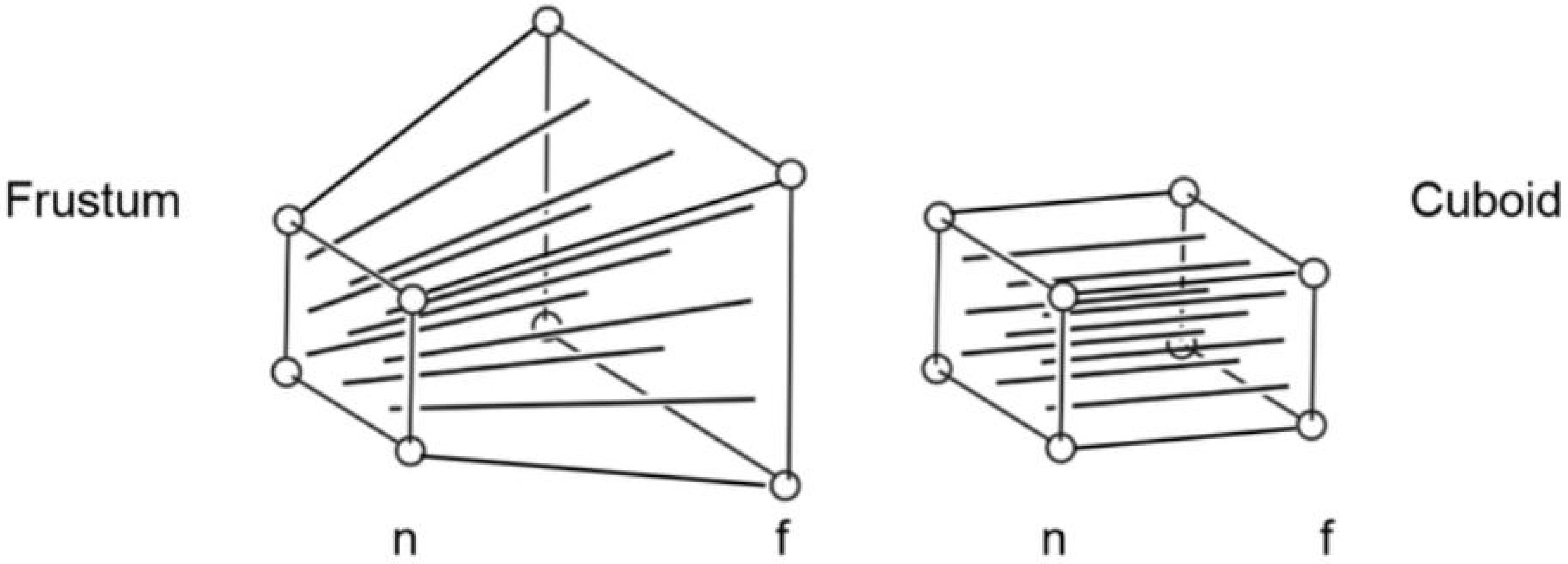
图 3.10　函数 gluPerspective()的参数:定义了三维空间中的四棱锥台,落在其内部的几何体将被投影成像



投影变换

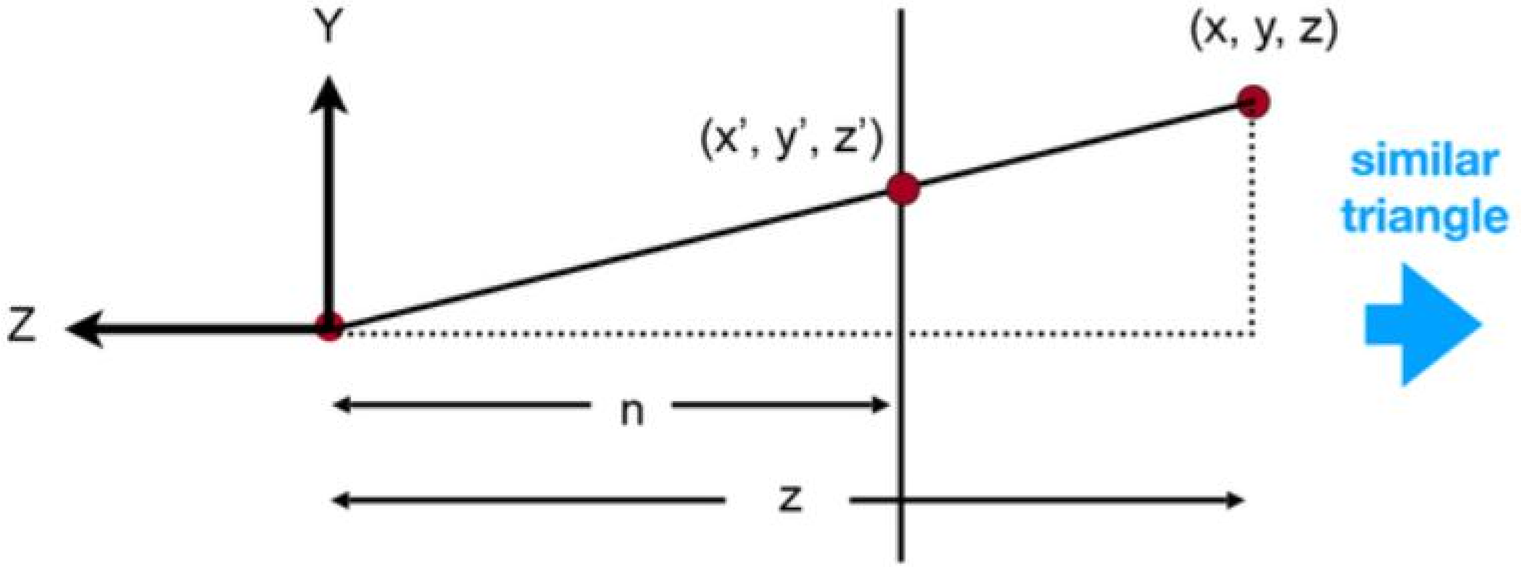
- 透视投影

- 近平面的所有点保持不变
- 远平面的所有点深度不变
- 远平面的中心点保持不变



投影变换

- 透视投影



$$y' = \frac{n}{z}y$$

投影变换

- 透视投影

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} nx/z \\ ny/z \\ \text{unknown} \\ 1 \end{pmatrix} \overset{\text{mult. by z}}{==} \begin{pmatrix} nx \\ ny \\ \text{still unknown} \\ z \end{pmatrix}$$

投影变换

- 透视投影

$$M_{persp\to ortho} = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ ? & ? & ? & ? \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

投影变换

- 透视投影
- 近平面所有点保持不变

$$\begin{pmatrix} x \\ y \\ n \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \\ n \\ 1 \end{pmatrix} == \begin{pmatrix} nx \\ ny \\ n^2 \\ n \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & A & B \end{pmatrix} \begin{pmatrix} x \\ y \\ n \\ 1 \end{pmatrix} = n^2$$

$$An + B = n^2$$

投影变换

- 透视投影
- 远平面中心点保持不变

$$\begin{pmatrix} 0 \\ 0 \\ f \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 \\ 0 \\ f \\ 1 \end{pmatrix} == \begin{pmatrix} 0 \\ 0 \\ f^2 \\ f \end{pmatrix} \qquad Af + B = f^2$$

投影变换

- 透视投影

$$An + B = n^2$$

$$Af + B = f^2$$

$$A = n + f, B = -nf$$

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

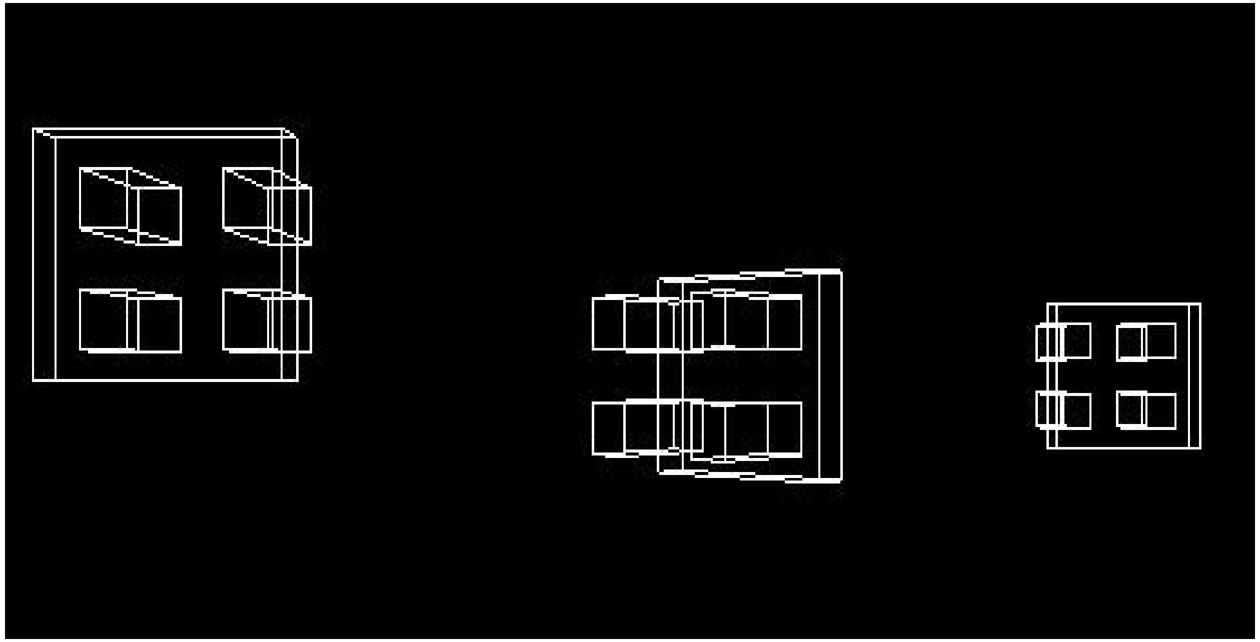
投影变换

- 透视投影

$$\mathbf{M}_{\text{per}} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{l+r}{l-r} & 0 \\ 0 & \frac{2n}{t-b} & \frac{b+t}{b-t} & 0 \\ 0 & 0 & \frac{f+n}{n-f} & \frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

实验内容

- 使用透视变换
- 绘制三维的会动的后三位学号，要求左边的沿着y轴上下运动，中间的绕着y轴正方向旋转，右边的沿着z轴缩放运动



实验内容

- Hint
  - 在reshape函数中将正投影的glOrtho()改为使用透视投影的gluPerspective()
  - 使用glutWireCube(1.0)函数可以绘制一个边长为1.0的线框正方体
- Bonus
  - 不直接调用glutWireCube绘制六面体