观察变换

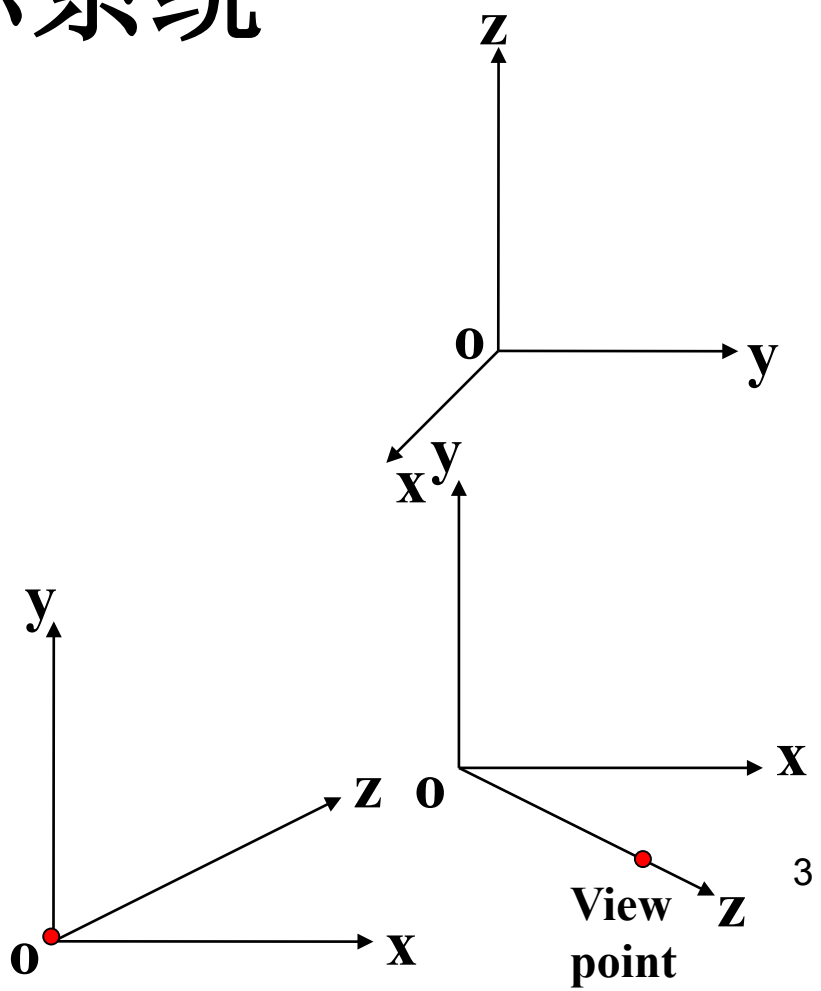
计算机图形学——原理、方法及应用（第4版），潘云鹤、童若锋、耿卫东、唐敏、童欣，高等教育出版社，2022。

观察

- 观察的目的是从特定的位置和方向显示物体　有很多参数需要设置。
- 观察指将物体从世界坐标系变换到观察坐标系，再变换到设备坐标系。
- 最终要投影将三维降到二维，等同于拍照。

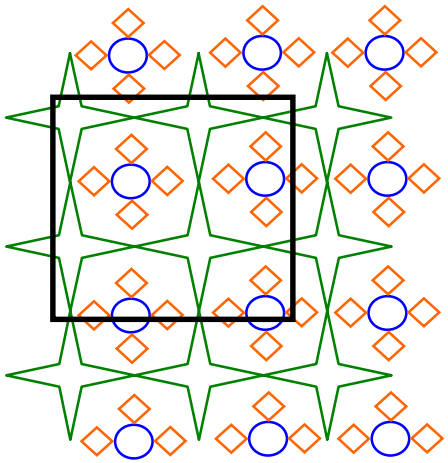
坐标系统

- **世界坐标系WCS**
  - *用于造型和摆放物体位置*
  - *右手系*
- **观察坐标系VCS**
  - *用于设定观察位置和方向*
  - *根据不同定义可以是左手系，目前使用右手系的更多*
- **设备坐标系DC**
- **标度化设备坐标系NDC**

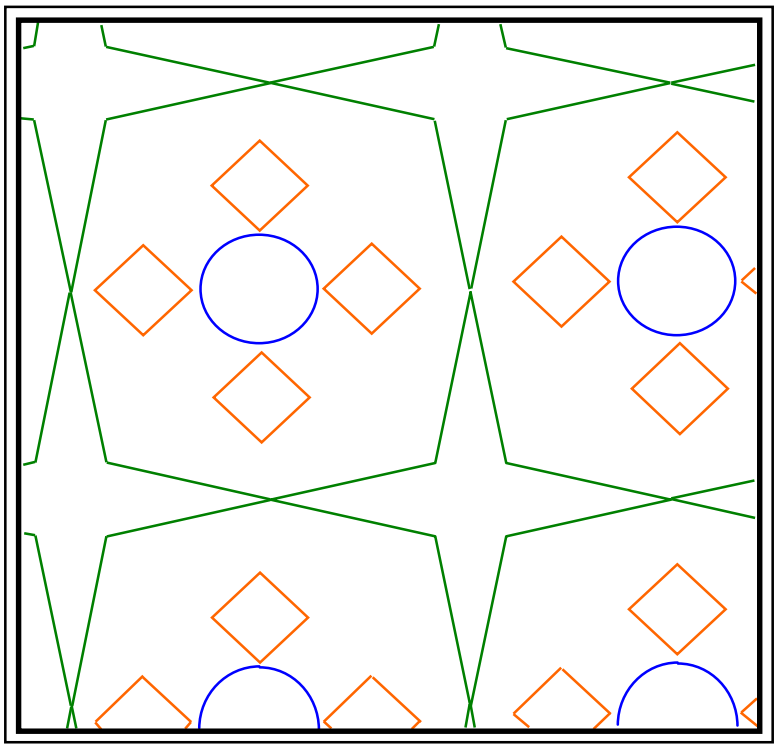


二维观察

- 将二维世界坐标系中的窗口映射到屏幕的某一区域



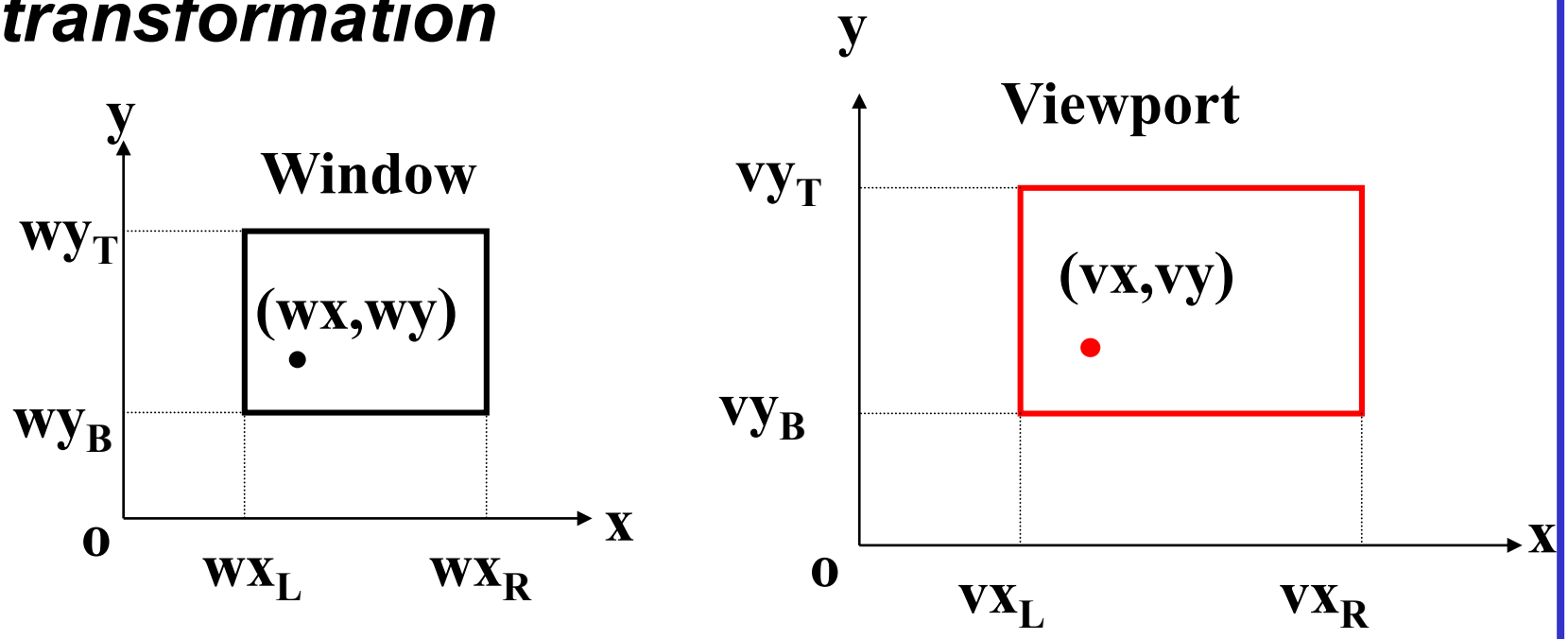
2D World



Screen

二维观察变换

- 将二维世界坐标系中的窗口***window*** 映射到屏幕的视口***viewport***，也称***window to viewport transformation***



二维观察变换

$$\begin{cases} \dfrac{vx - vx_L}{vx_R - vx_L} = \dfrac{wx - wx_L}{wx_R - wx_L} \\ \dfrac{vy - vy_B}{vy_T - vy_B} = \dfrac{wy - wy_B}{wy_T - wy_B} \end{cases}$$

$$\begin{cases} vx = \dfrac{vx_R - vx_L}{wx_R - wx_L} \bullet (wx - wx_L) + vx_L \\ vy = \dfrac{vy_T - vy_B}{wy_T - wy_B} \bullet (wy - wy_B) + vy_B \end{cases}$$

OpenGL 中的相应函数

gluOrtho2D(left, right, bottom, top)

Creates a matrix for projecting 2D coordinates onto the screen and multiplies the current matrix by it.

glViewport(x, y, width, height)

Define a pixel rectangle into which the final image is mapped.

(x, y) specifies the lower-left corner of the viewport.

(width, height) specifies the size of the viewport rectangle.

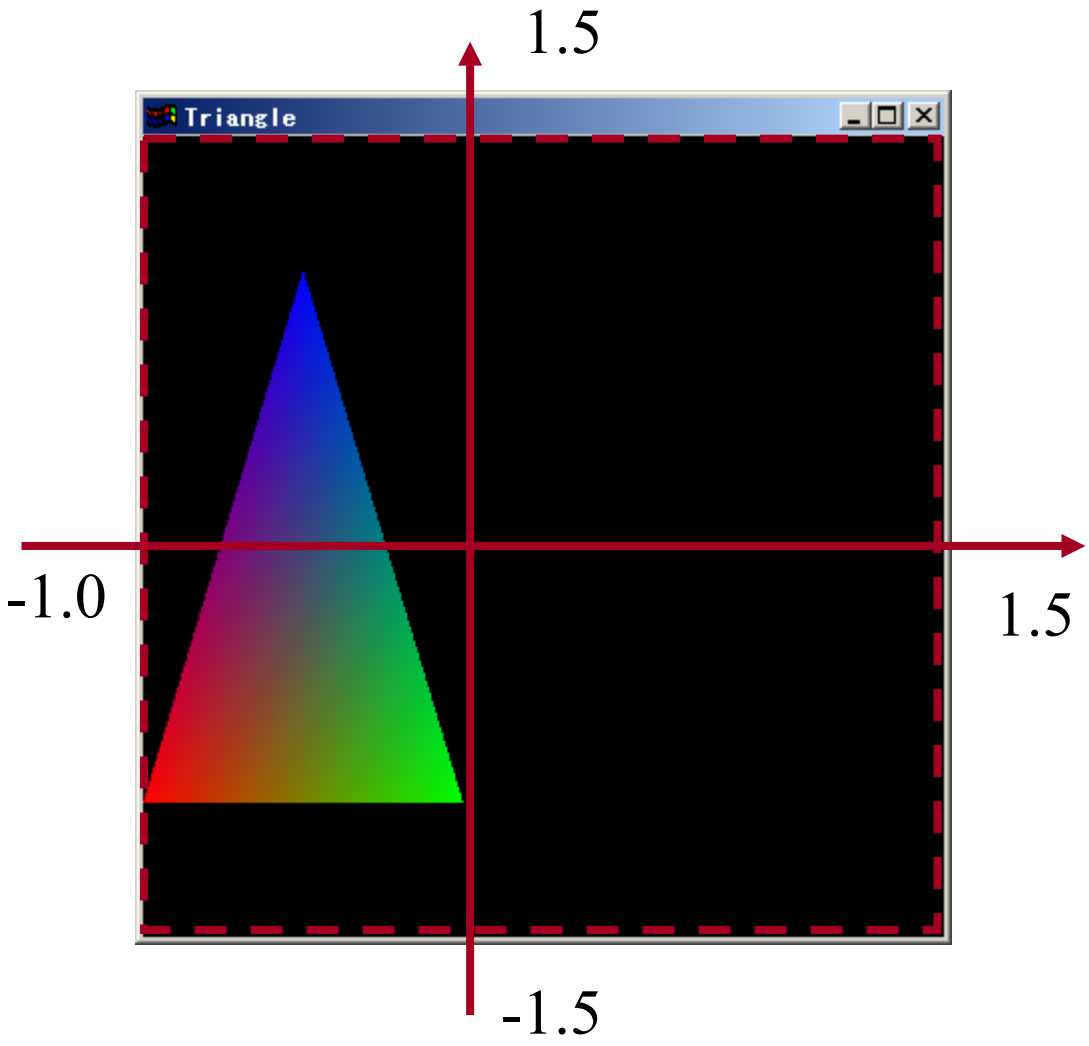
2D 显示实例

```
void myReshape(GLsizei w,GLsizei h)
{
    glViewport(0,0,w,h);//设置视口

    glMatrixMode(GL_PROJECTION);//指明当前矩阵为GL_PROJECTION
    glLoadIdentity();//将当前矩阵置换为单位阵

    //定义二维正视投影矩阵
    GLfloat ratio=(GLfloat)w/(GLfloat)h;
    if(w <= h)
        gluOrtho2D(-1.5,1.5,-1.5/ratio,1.5/ratio);
    else
        gluOrtho2D(-1.5*ratio,1.5*ratio,-1.5,1.5);

    glMatrixMode(GL_MODELVIEW);//指明当前矩阵为GL_MODELVIEW
}
```

2D 显示实例

```
void myDisplay(void)
{
    glClear(GL_COLOR_BUFFER_BIT);//buffer设置为颜色可写

    glBegin(GL_TRIANGLES);//开始画三角形
    glShadeModel(GL_SMOOTH);//设置为光滑明暗模式

    glColor3f(1.0,0.0,0.0);//设置第一个顶点为红色
    glVertex2f(-1.0,-1.0);//设置第一个顶点的坐标为（-1.0，-1.0）

    glColor3f(0.0,1.0,0.0);//设置第二个顶点为绿色
    glVertex2f(0.0,-1.0);//设置第二个顶点的坐标为（.0，-1.0）

    glColor3f(0.0,0.0,1.0);//设置第三个顶点为蓝色
    glVertex2f(-0.5,1.0);//设置第三个顶点的坐标为（-0.5，1.0）
    glEnd();//三角形结束

    glFlush();//强制OpenGL函数在有限时间内运行
}
```

2D 显示实例



几何变换 vs 观察变换

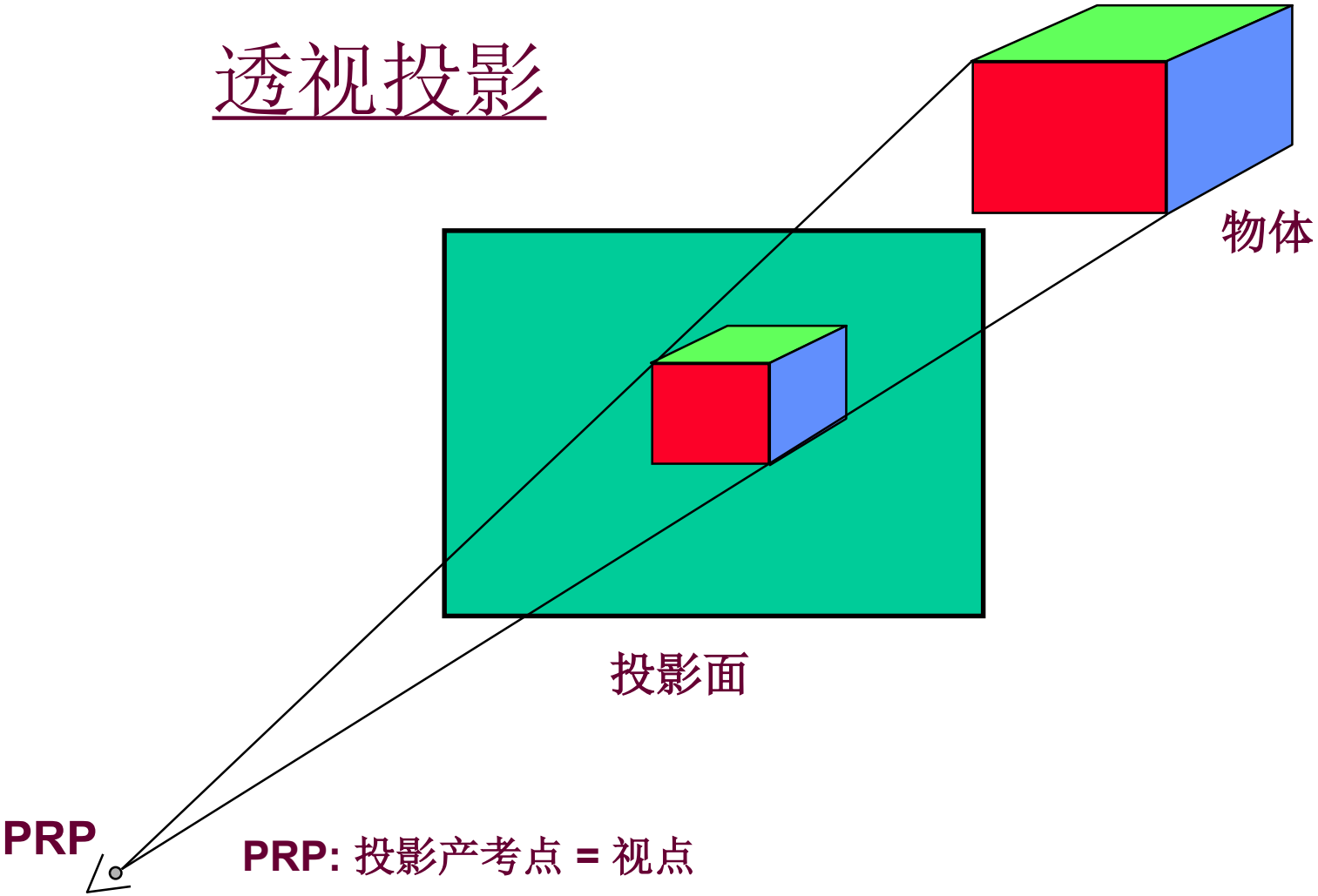
- *几何变换改变物体位置或朝向*
- *观察变换并不改动物体的真实位置，只是将其在不同坐标系统中表示*

变换流程

- **几何变换　Modeling transformation**
  - 在世界坐标系中移动摆放物体.
- **观察变换　Viewing transformation**
  - 摆放相机（设定观察坐标系）. 将物体从世界坐标系变换到观察坐标系World coordinate system(WCS) → View coordinate system(VCS)
- **投影变换　Projection transformation**
  - 将物体从观察坐标系VCS 投影到投影面
- **视口变换　Viewport transformation**
  - 缩放并平移到屏幕特定位置.

观察:

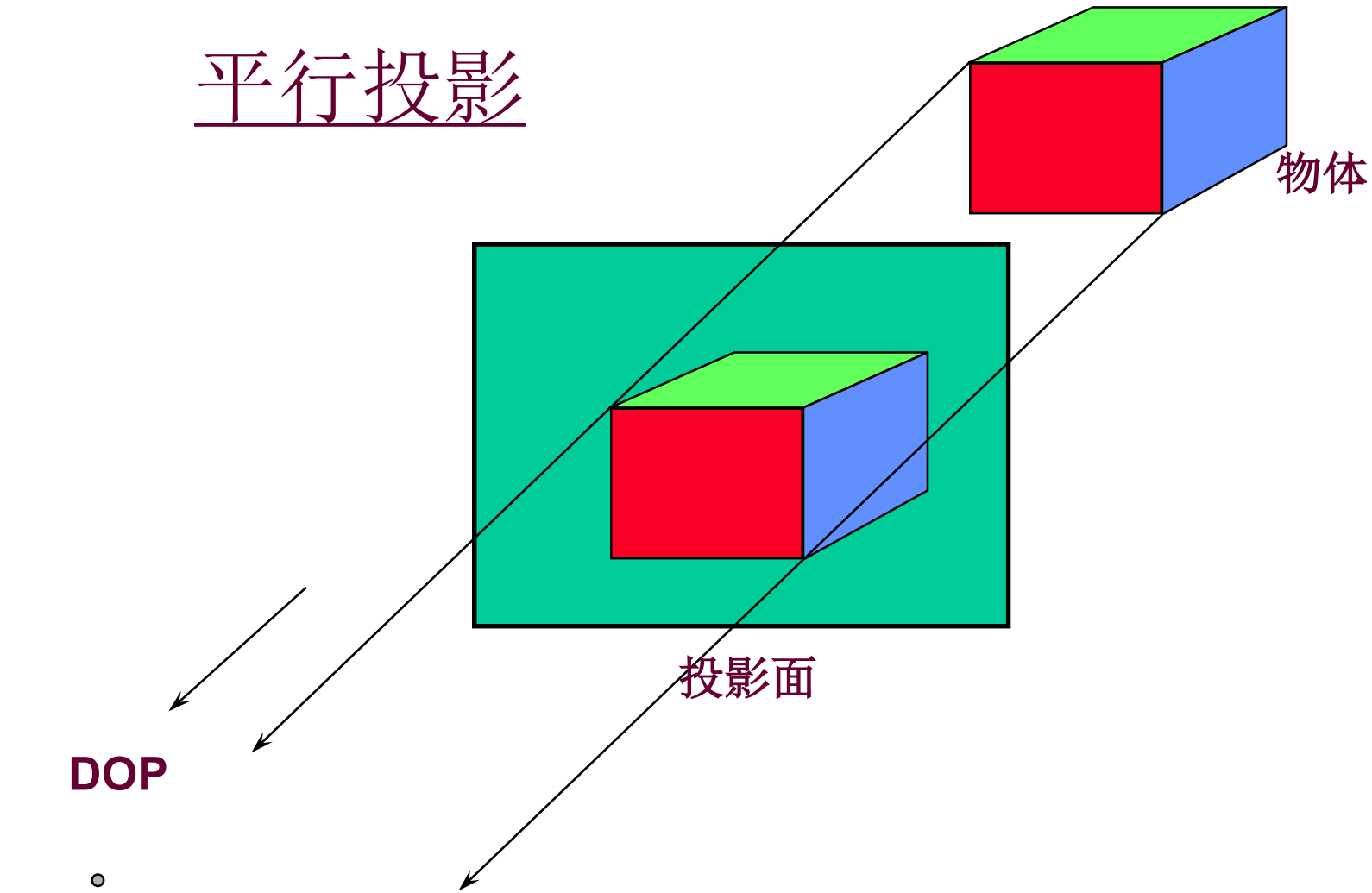
透视投影



Figures extracted from Angle's textbook

观察

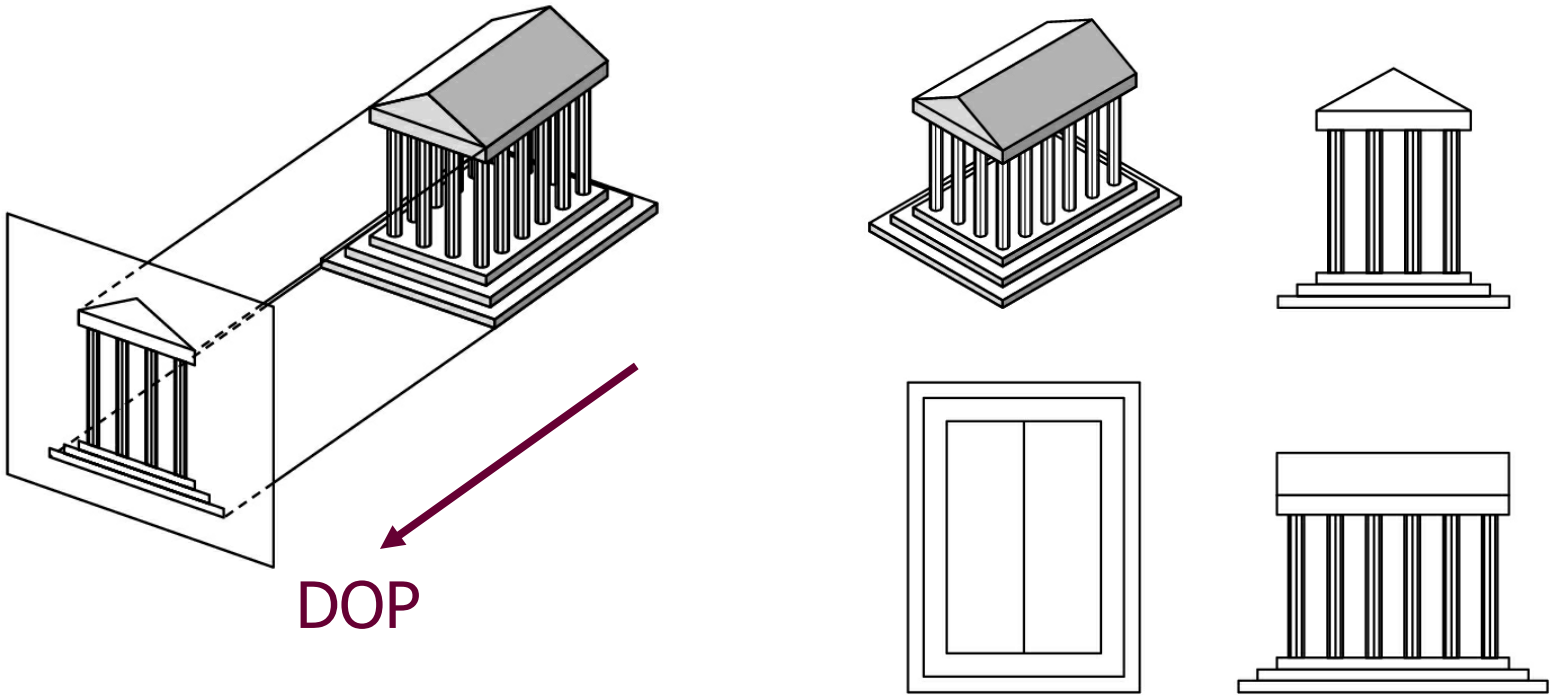
平行投影



DOP: 投影方向= 视点位于无穷远

Figures extracted from Angle's textbook

正投影



DOP 与投影面垂直

Figures extracted from Angle's textbook

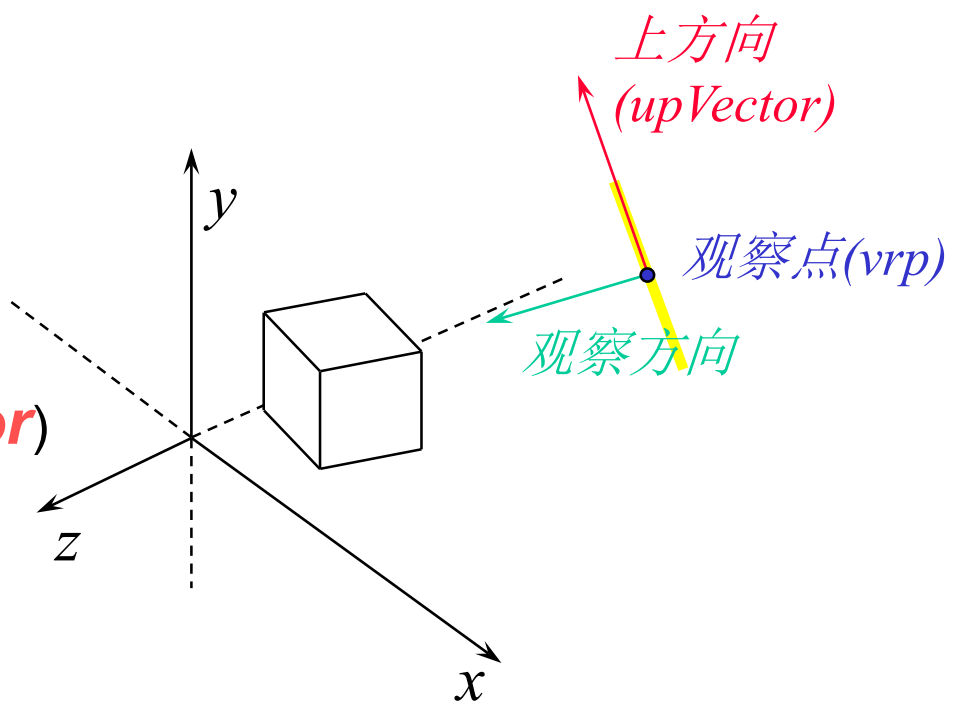
设定观察坐标系

需设定参数

观察点 (*view reference point*)

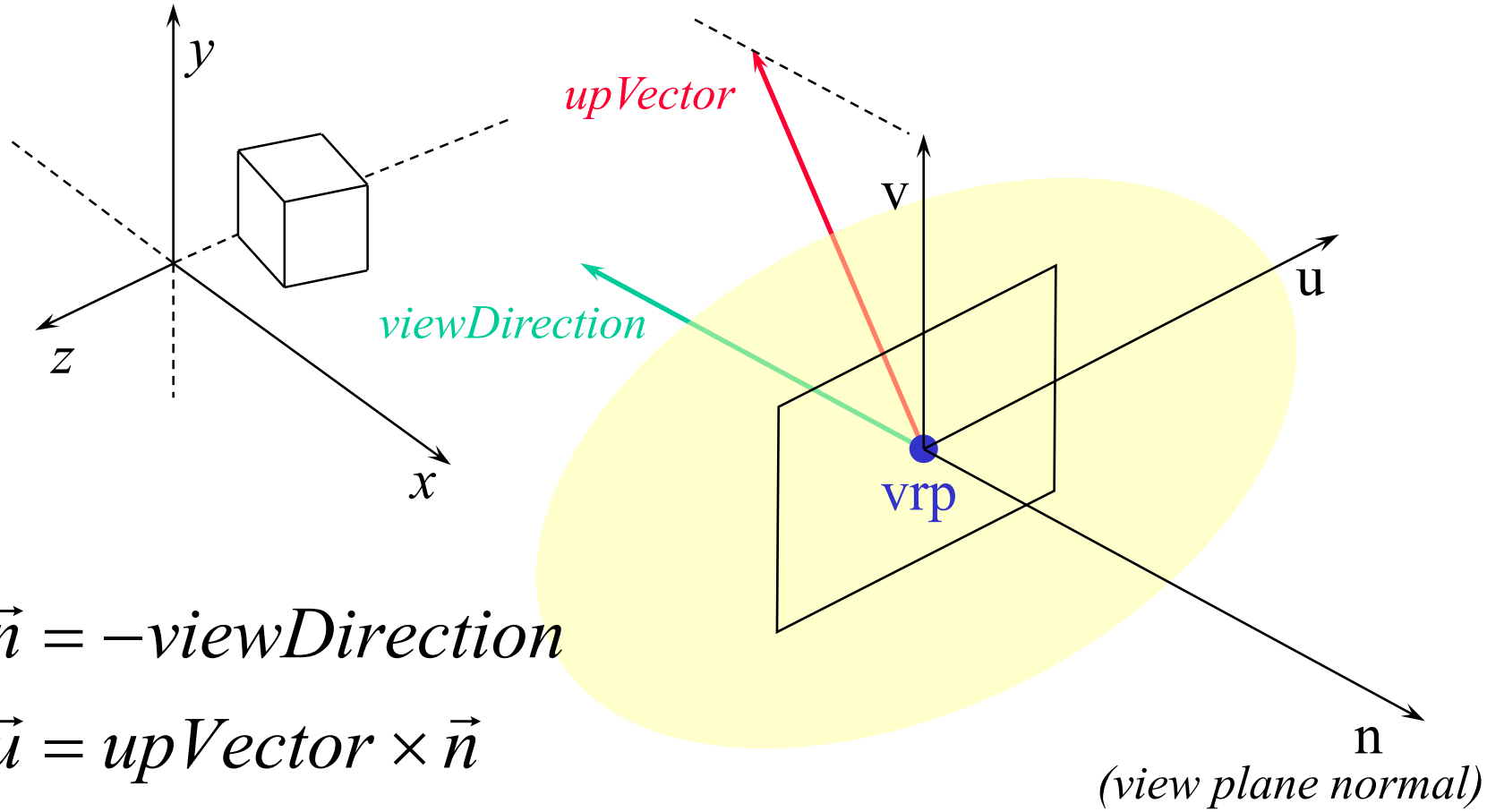
观察方向(***viewDirection***)

图像向上的方向(***upVector***)



所有参数是在世界坐标系中的值

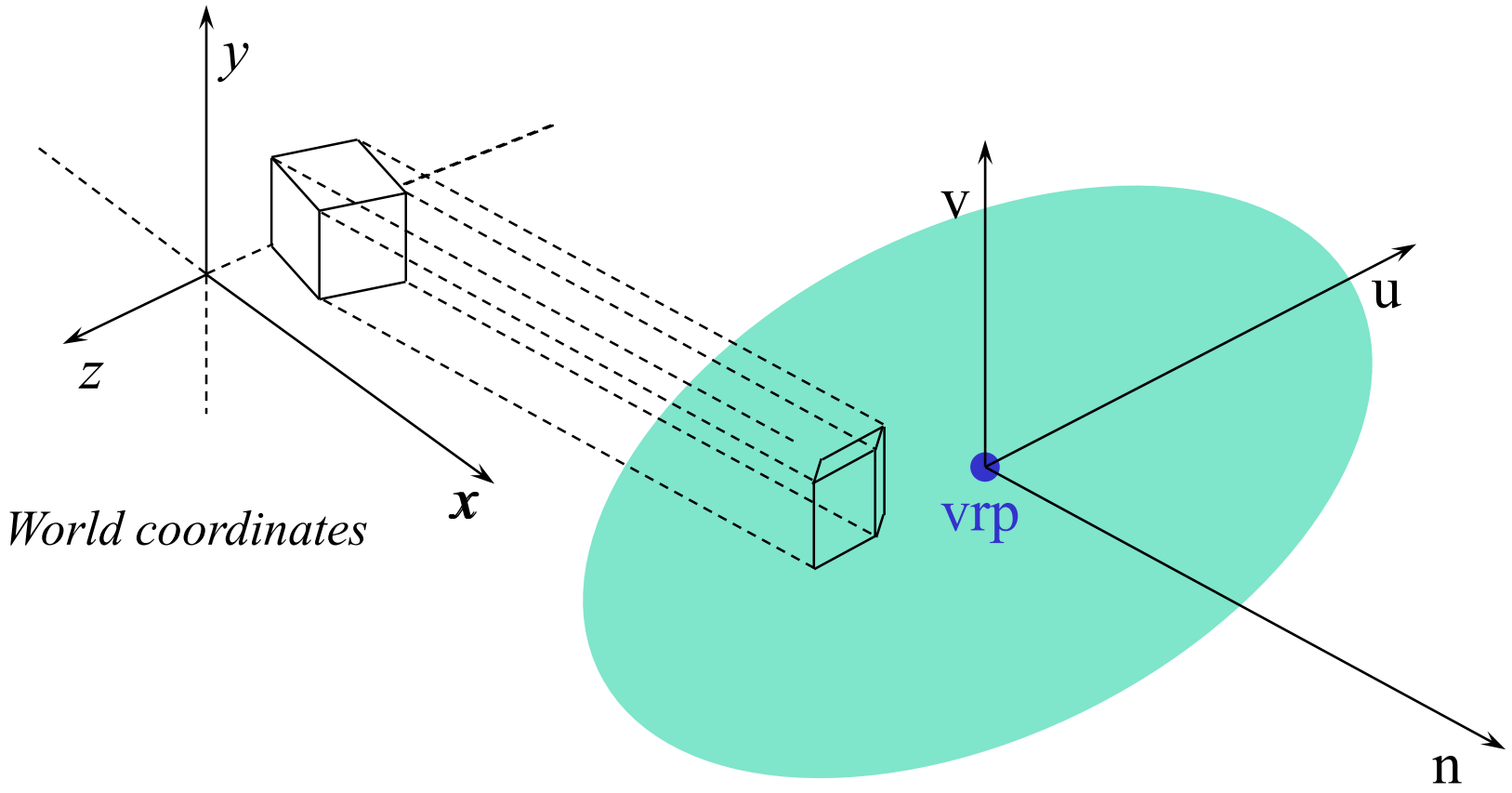
观察坐标系的计算



$$\vec{n} = -viewDirection$$

$$\vec{u} = upVector \times \vec{n}$$

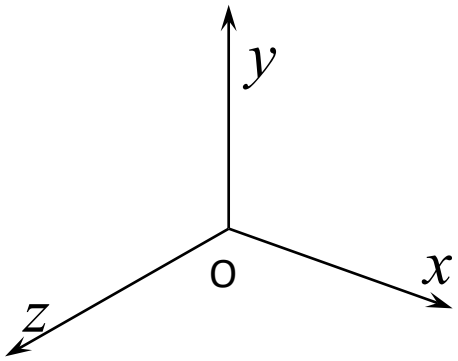
$$\vec{v} = \vec{n} \times \vec{u}$$

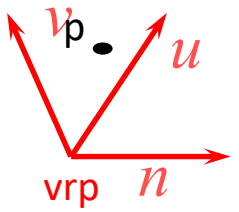


- 由于投影方向在观察坐标系中与坐标轴方向垂直，将物体坐标都变换到观察坐标系后，投影变得简单（相当于直接去掉Z值）。

观察变换矩阵

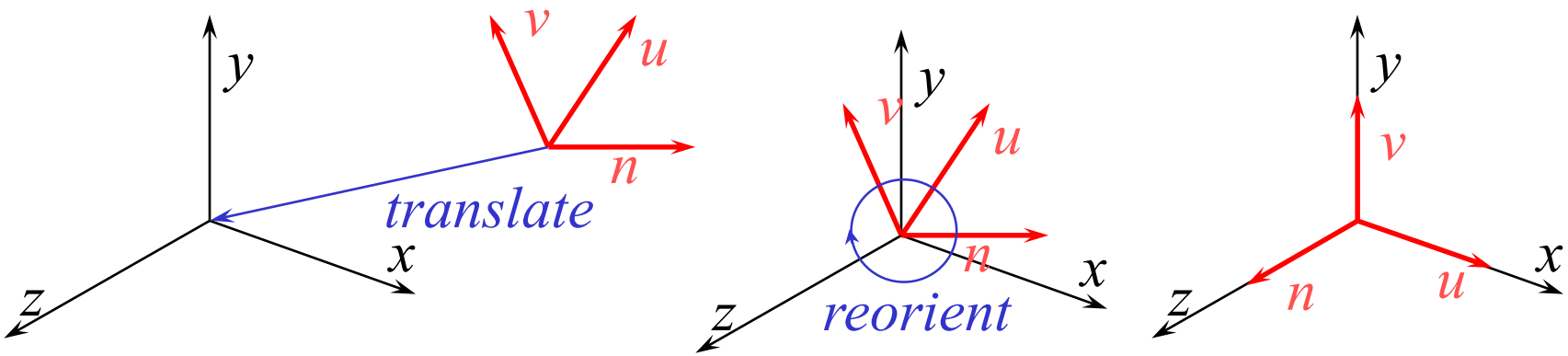
观察变换矩阵M 将点的坐标从世界左边系变换到观察坐标系





若 $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ 为点在世界坐标系中的坐标，$\begin{bmatrix} \text{new}x \\ \text{new}y \\ \text{new}z \\ 1 \end{bmatrix}$ 为它在观察坐标系中的点，则，$\begin{bmatrix} \text{new}x \\ \text{new}y \\ \text{new}z \\ 1 \end{bmatrix} = M \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ 问题：M如何求。

观察变换矩阵



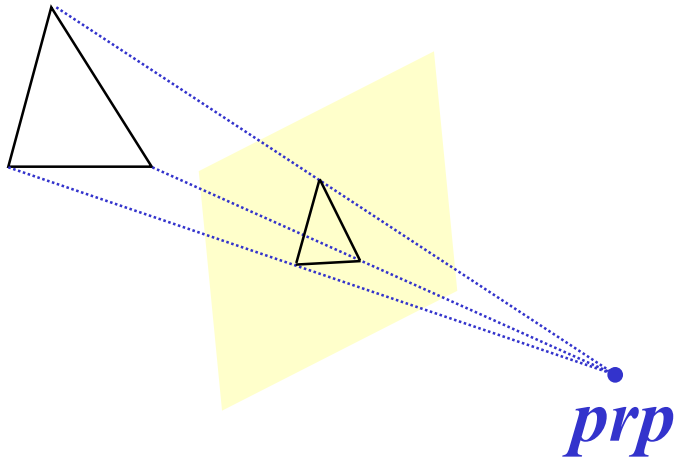
$$M = \begin{bmatrix} u_x & u_y & u_z & -\vec{u} \bullet vrp \\ v_x & v_y & v_z & -\vec{v} \bullet vrp \\ n_x & n_y & n_z & -\vec{n} \bullet vrp \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

为什么？

透视投影

- The points are transformed to the view plane along lines that converge to a point called ***projection reference point*** (***prp***) or ***center of projection*** (***cop***)

- ***prp*** is specified in terms of the viewing coordinate system

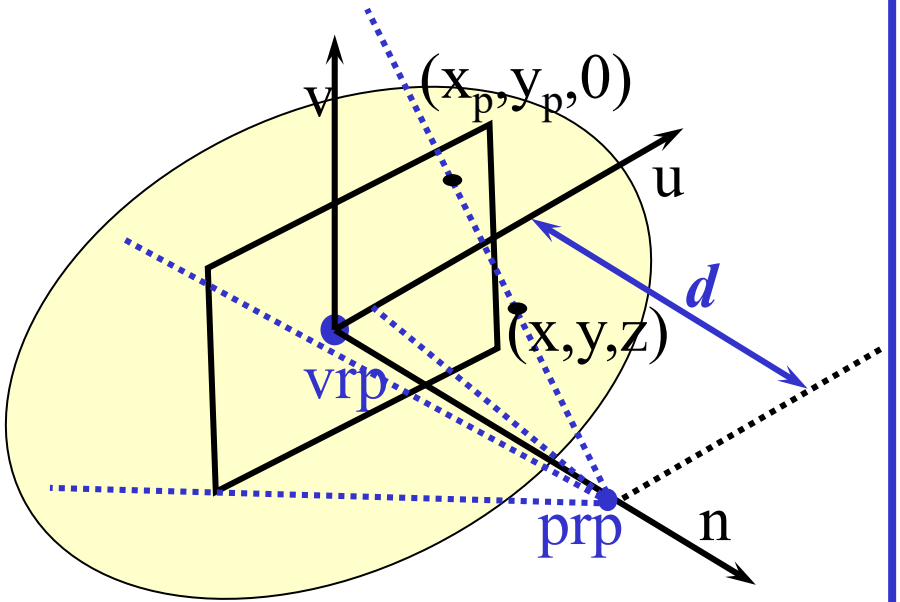


透视投影矩阵

- ***prp*** 是投影中心，设其与投影面的距离为 ***d***.

则透视投影矩阵为

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/d & 1 \end{bmatrix}$$



为什么?

$$\frac{x}{x_p} = \frac{y}{y_p} = \frac{d-z}{d}$$

$$x, y, 0, -z/d+1=\frac{d-z}{d}$$

观察＋透视投影后的矩阵

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/d & 1 \end{bmatrix} \times \begin{bmatrix} u_x & u_y & u_z & -\vec{u} \bullet vrp \\ v_x & v_y & v_z & -\vec{v} \bullet vrp \\ n_x & n_y & n_z & -\vec{n} \bullet vrp \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

然后进行viewport transformation变换到标度化设备坐标系。

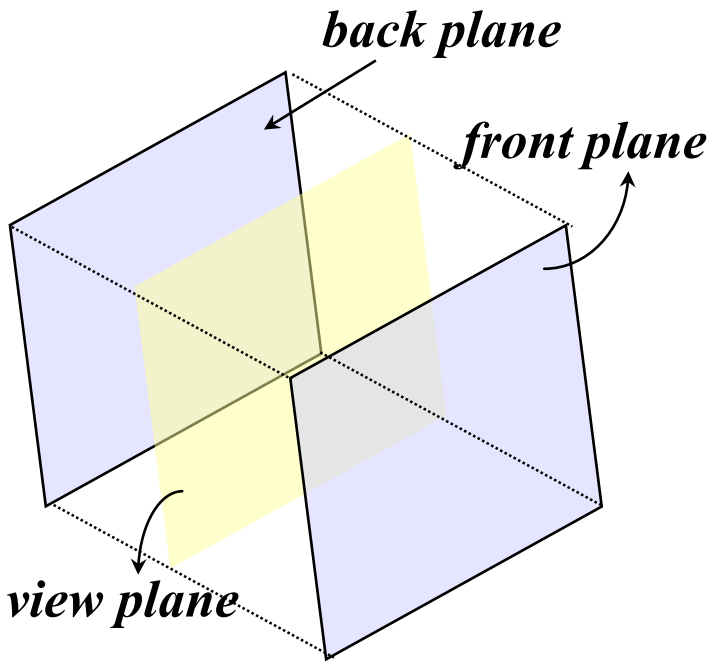
平行投影

观察＋平行投影后的变换矩阵

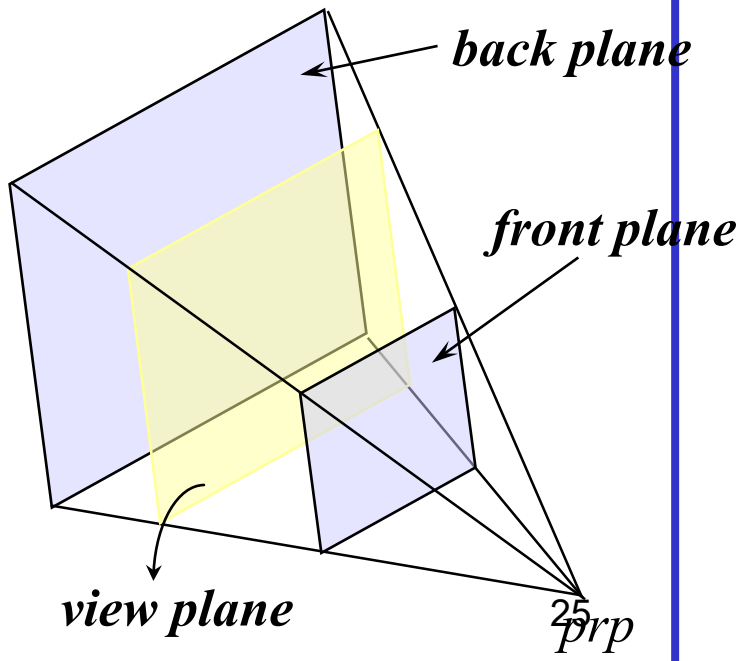
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} u_x & u_y & u_z & -\vec{u} \bullet vrp \\ v_x & v_y & v_z & -\vec{v} \bullet vrp \\ n_x & n_y & n_z & -\vec{n} \bullet vrp \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

然后进行viewport transformation变换到标度化设备坐标系。

裁剪



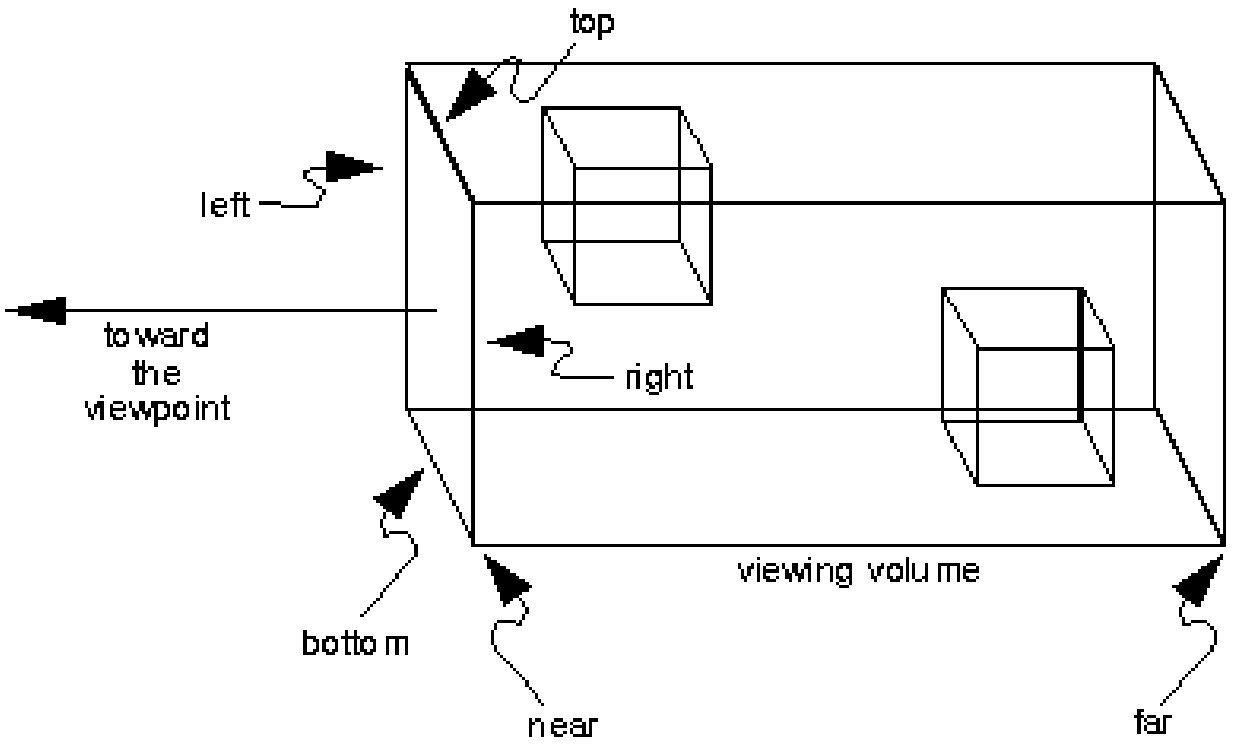
正投影留下长方体内的物体



透视投影留下四棱柱内的物体

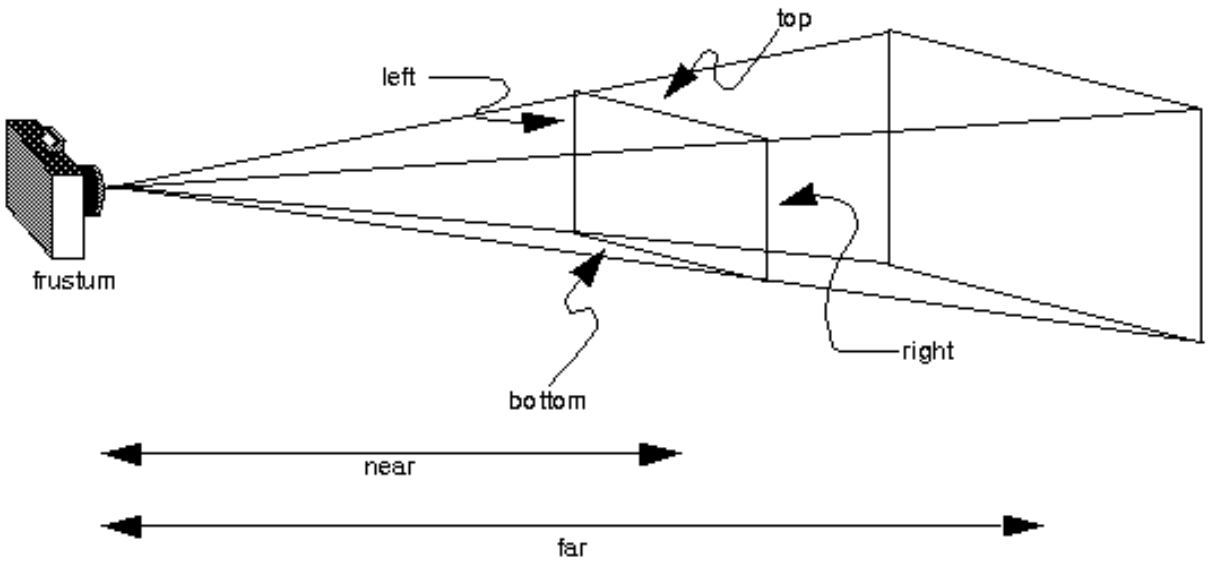
OpenGL 正投影

glOrtho(left, right, bottom, top, near, far);



OpenGL 透视

glFrustum(left, right, bottom, top, near, far);



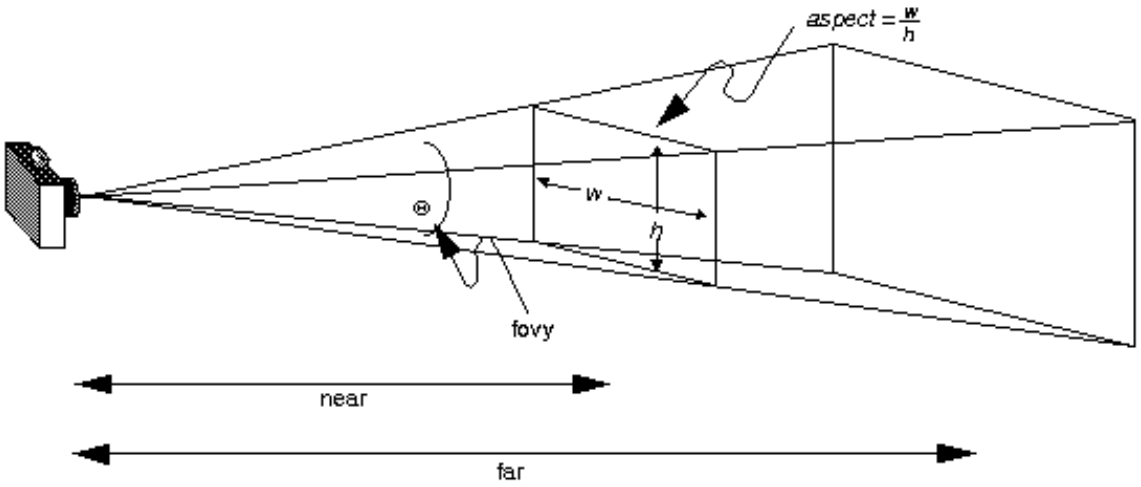
glMatrixMode(GL_PROJECTION);

glLoadIdentity( );

glFrustum(left, right, bottom, top, near, far);

OpenGL 透视

gluPerspective(fovy, aspect, near, far);



fov 是上下平面间的角度

Glu中的观察坐标系设定

```
gluLookAt(eyex, eyey, eyez, atx, aty, atz,
          upx, upy, upz);
```

`eyex, eyey, eyez` **specify the position of the eye point and are mapped to the origin.**

`atx, aty, atz` **specify a point being looked at, which will be rendered in center of view port. It is mapped to the -z axis.**

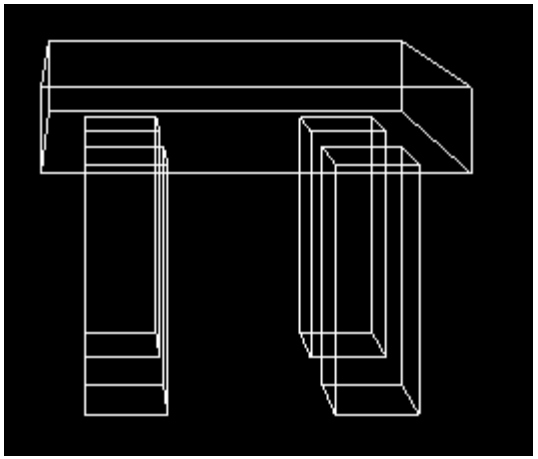
`upx, upy, upz` **specify components of the camera up vector.**

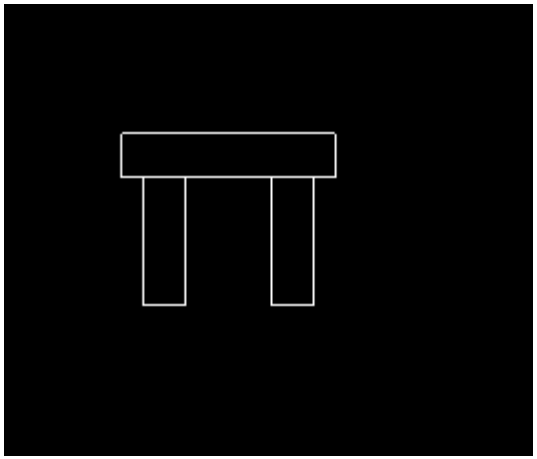
gluLookAt

```
{
    float forward[3], side[3], up[3];
    GLfloat m[4][4];

    forward[0] = centerx - eyex;
    forward[1] = centery - eyey;
    forward[2] = centerz - eyez;

    up[0] = upx;
    up[1] = upy;
    up[2] = upz;

    normalize(forward);

    /* Side = forward x up */
    cross(forward, up, side);
    normalize(side);

    /* Recompute up as: up = side x forward
       */
    cross(side, forward, up);

    __gluMakeIdentityf(&m[0][0]);
    m[0][0] = side[0];
    m[1][0] = side[1];
    m[2][0] = side[2];

    m[0][1] = up[0];
    m[1][1] = up[1];
    m[2][1] = up[2];

    m[0][2] = -forward[0];
    m[1][2] = -forward[1];
    m[2][2] = -forward[2];

    glMultMatrixf(&m[0][0]);
    glTranslated(-eyex, -eyey, -eyez);
}
```

实验3

- 实现正投影和透视投影
- 交互设定观察坐标系





谢　　谢