華東理工大學

计算机图形学

2023年10月

奉贤校区

07

三维变换和三维观察

3D Transformation and View

绘制流水线

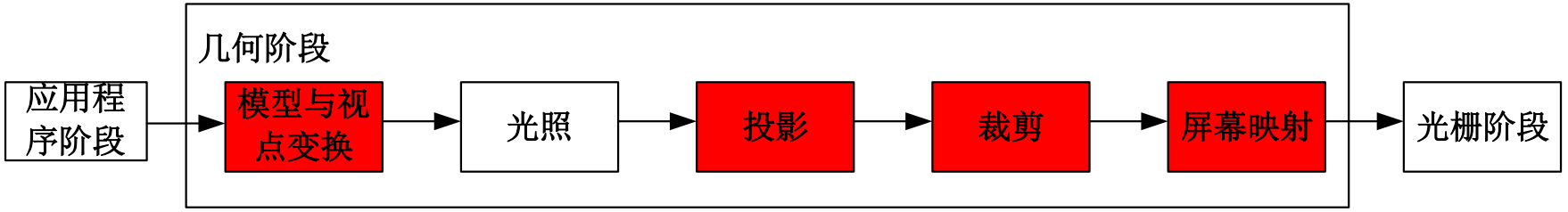
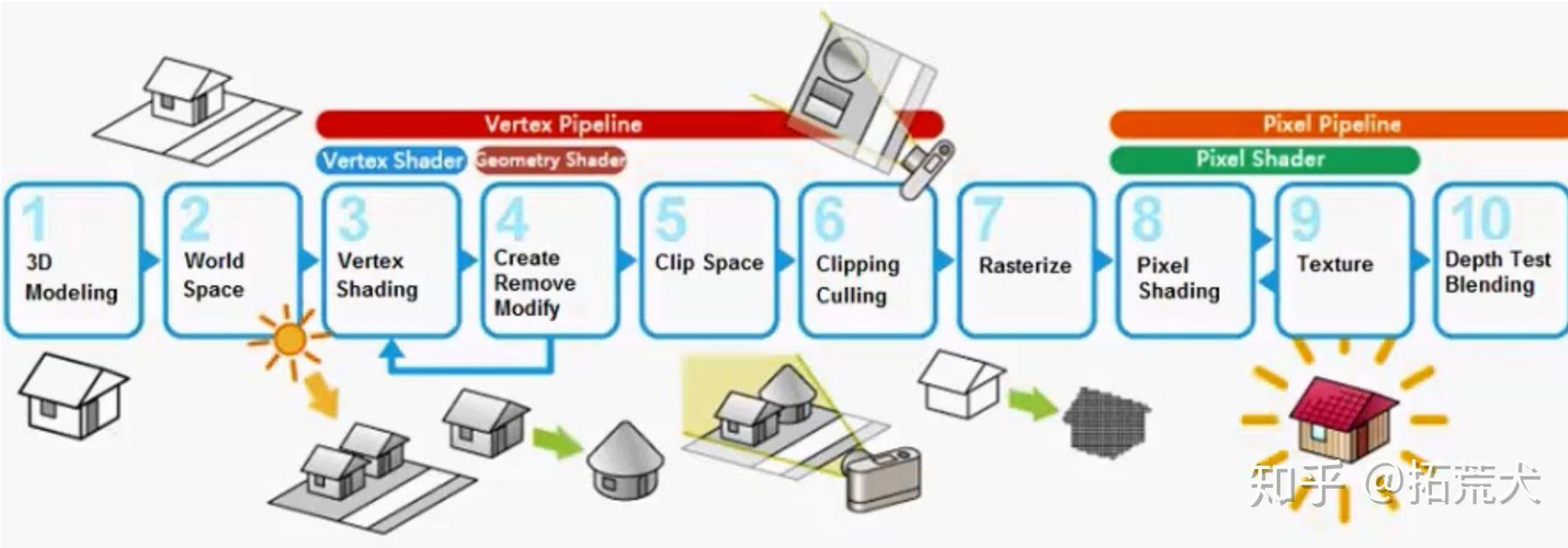
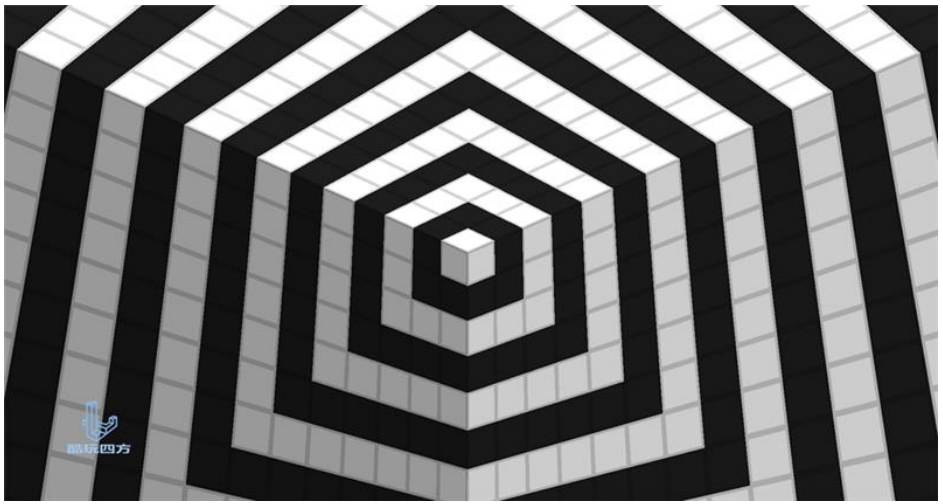


图2.22 绘制流水线的结构


Vertex Pipeline
Vertex Shader
Geometry Shader
Pixel Pipeline
Pixel Shader
1 3D Modeling
2 World Space
3 Vertex Shading
4 Create Remove Modify
5 Clip Space
6 Clipping Culling
7 Rasterize
8 Pixel Shading
9 Texture
10 Depth Test Blending
知乎 @拓荒犬

本章要点

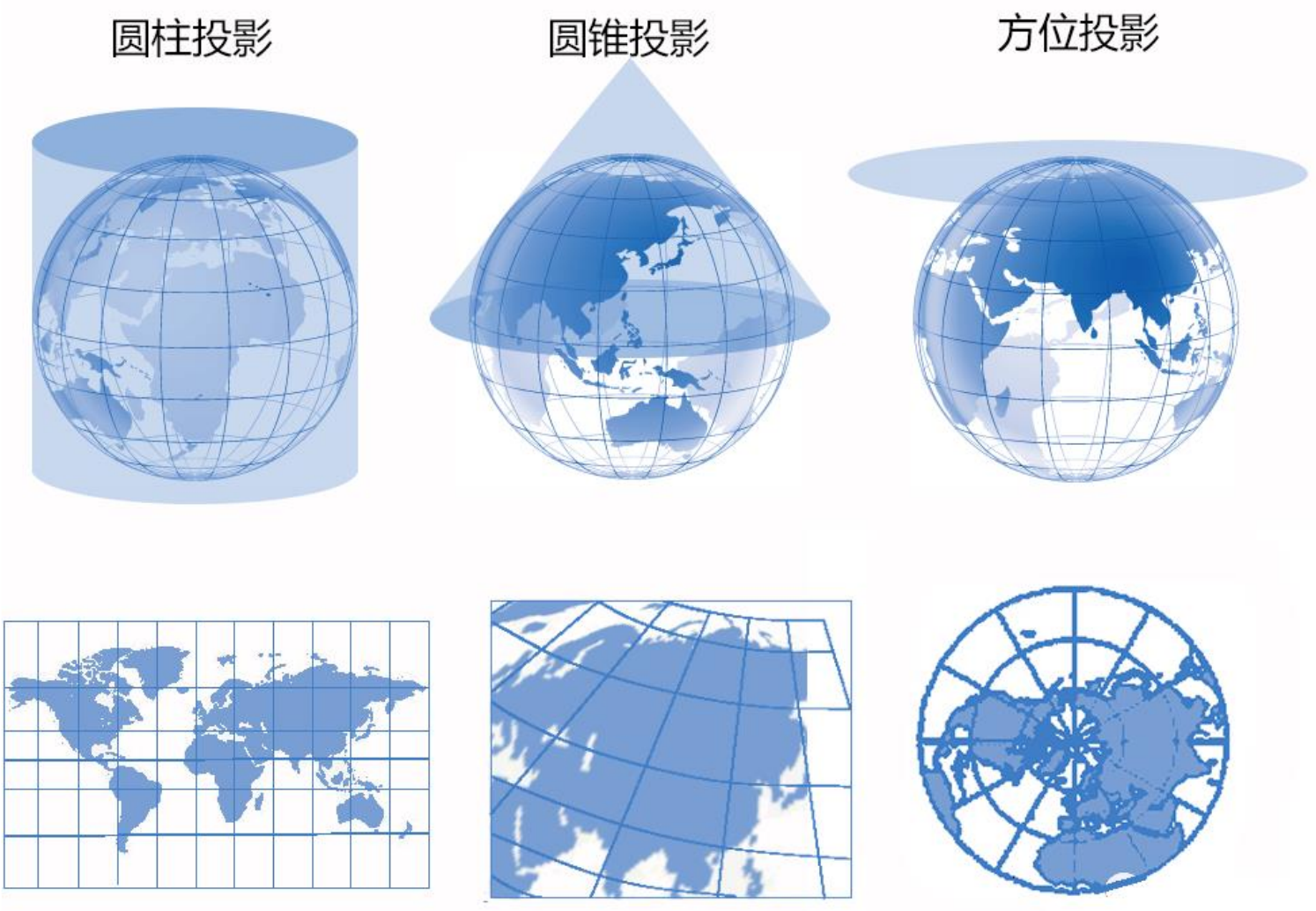
- 如何对三维图形进行方向、尺寸和形状方面的变换;
- 如何进行投影变换;（三维->二维）
- 如何方便地实现在显示设备上对三维图形进行观察;
- OpenGL中的观察变换

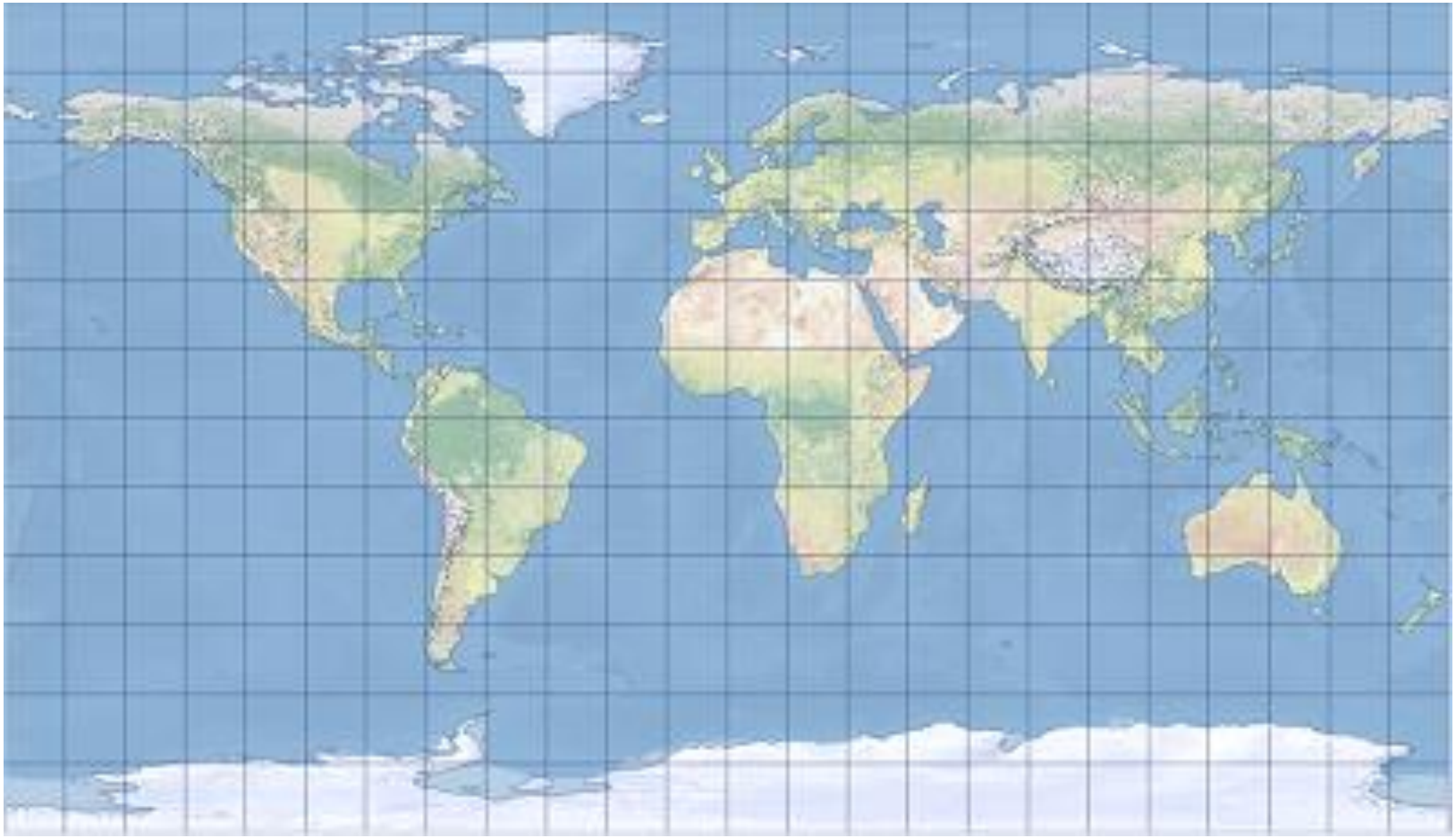


问题

1. 为什么要进行投影操作？
2. 三视图常用在哪个领域？
3. 有没有非平面几何投影？举例

地图投影

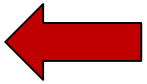




- 圆柱投影或墨卡托投影

三维观察变换

1. 观察坐标系
2. 观察空间
3. 三维观察流程
4. 三维裁剪



观察坐标系

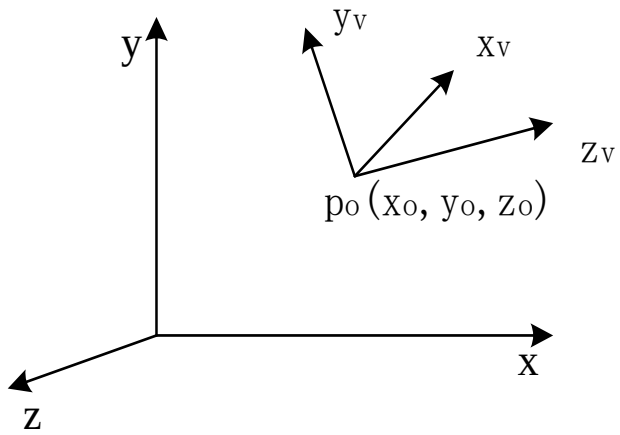


图7.27 用户坐标系与观察坐标系

- 观察参考坐标系（View Reference Coordinate）
- 观察参考点（View Reference Point）

观察坐标系

- 观察平面（View Plane），即投影平面。

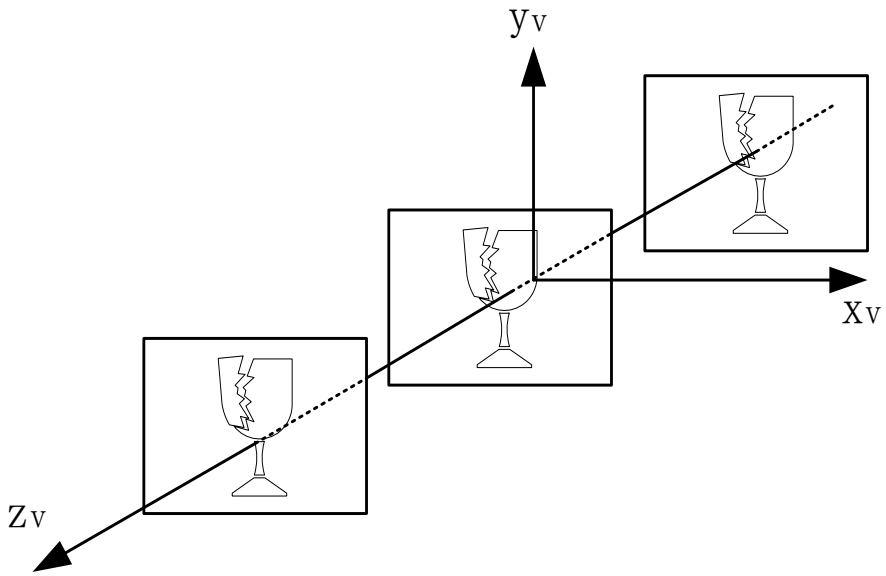
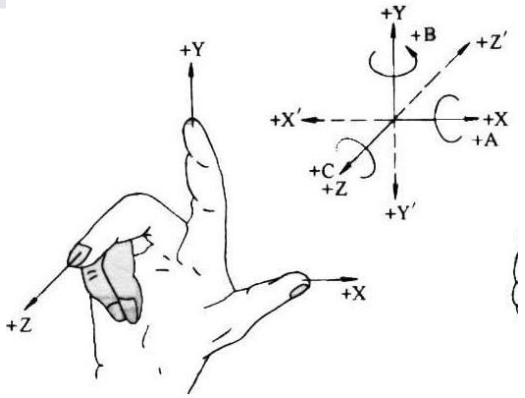


图7.28 沿zv轴的观察平面

观察坐标系

- 观察坐标系（UVN坐标系）的建立

（1）法矢量N



右手坐标系

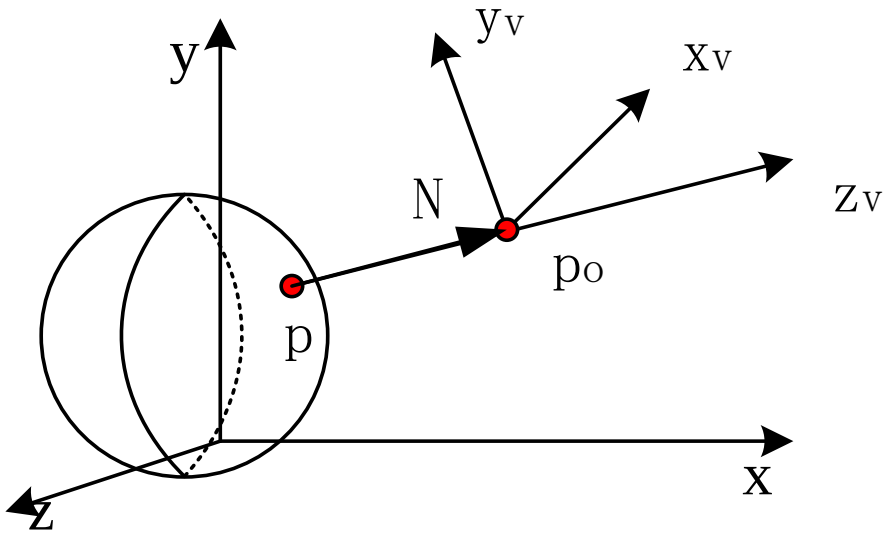


图7.28 法矢量N的定义

(2) 法矢量V的建立

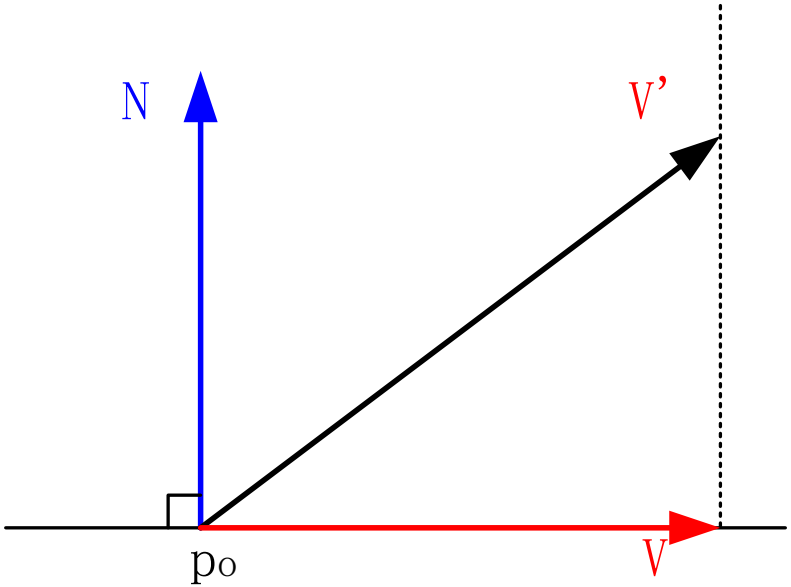
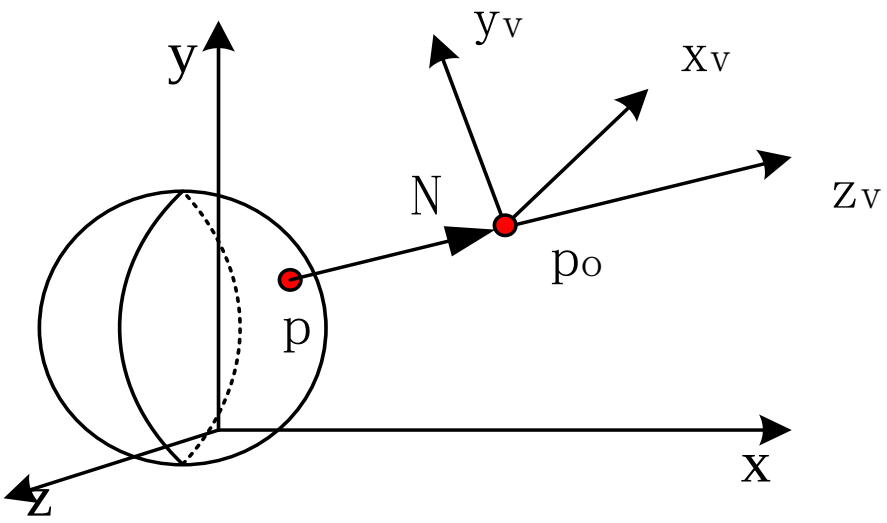


图7.29 法矢量V的定义

(3) 法矢量U

观察坐标系

- 通过改变观察参考点的位置或改变N的方向可以使用户在不同的距离和角度上观察三维形体。

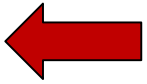


UVN
XYZ

图7.30 三维观察

三维观察变换

1. 观察坐标系
2. 观察空间
3. 三维观察流程
4. 三维裁剪



观察空间

相当于二维的观察窗口，只有观察空间内的图形才会被现实到输出设备上，观察空间外的图形不显示。

- 观察窗口：

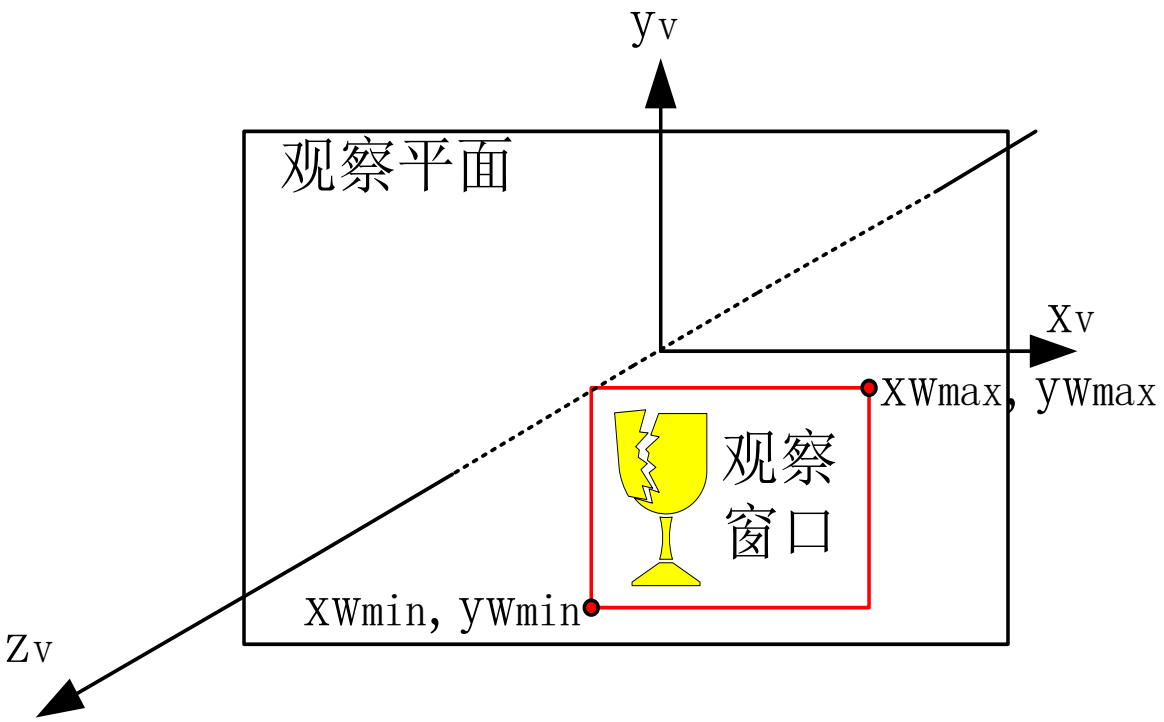


图7.31 观察窗口

观察空间

□观察空间：将观察窗口沿投影方向作平移运动产生的三维形体。

□观察空间的大小和形状依赖于窗口的大小及投影类型。

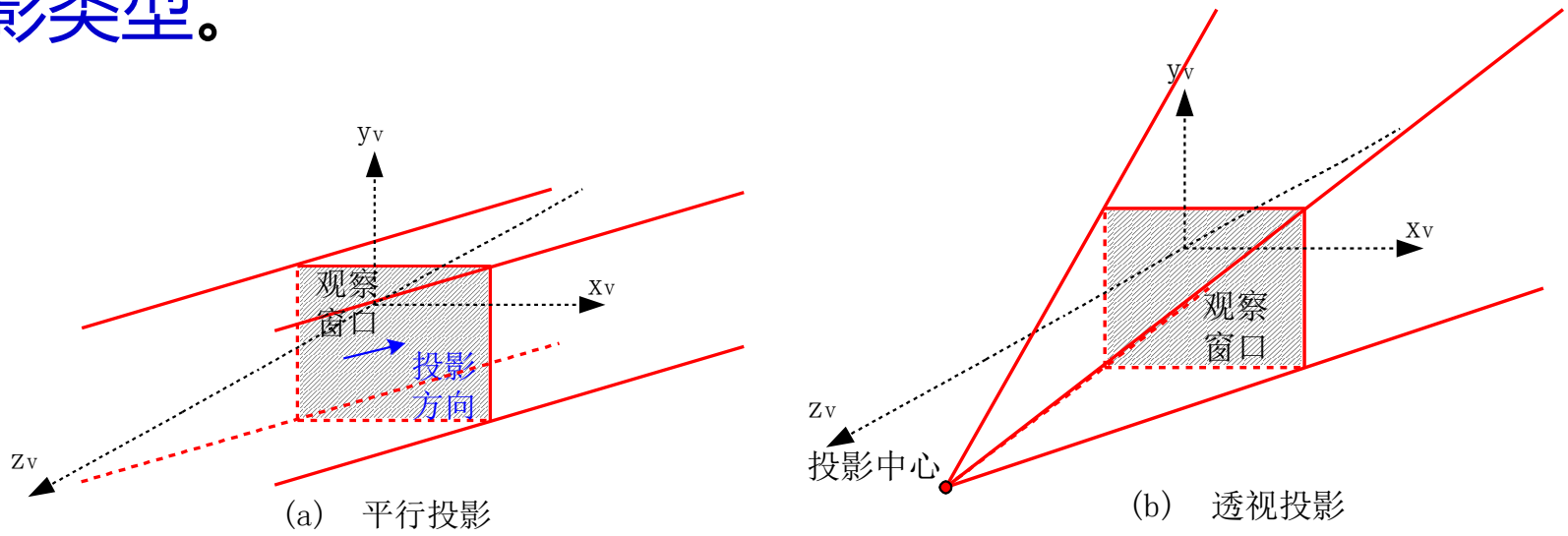


图7.32 观察空间

观察空间

- 无限观察空间、有限观察空间
- 前后截面：$Z=Z_{front}$，$Z=Z_{back}$

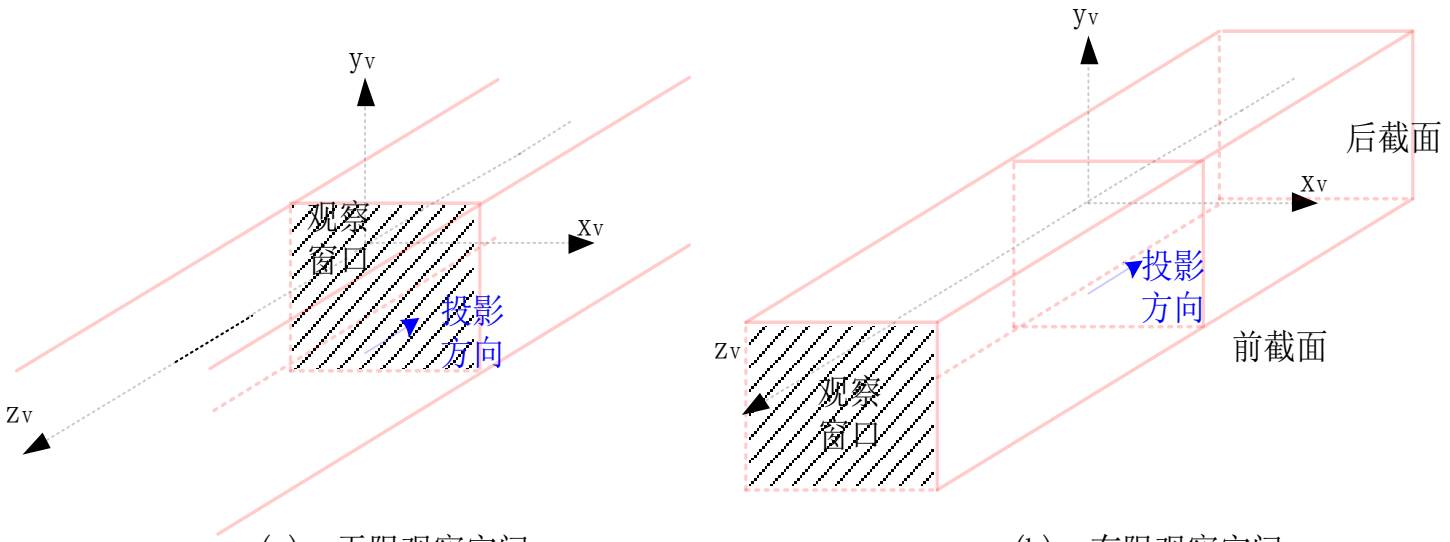


图7.33 平行投影观察空间

观察空间

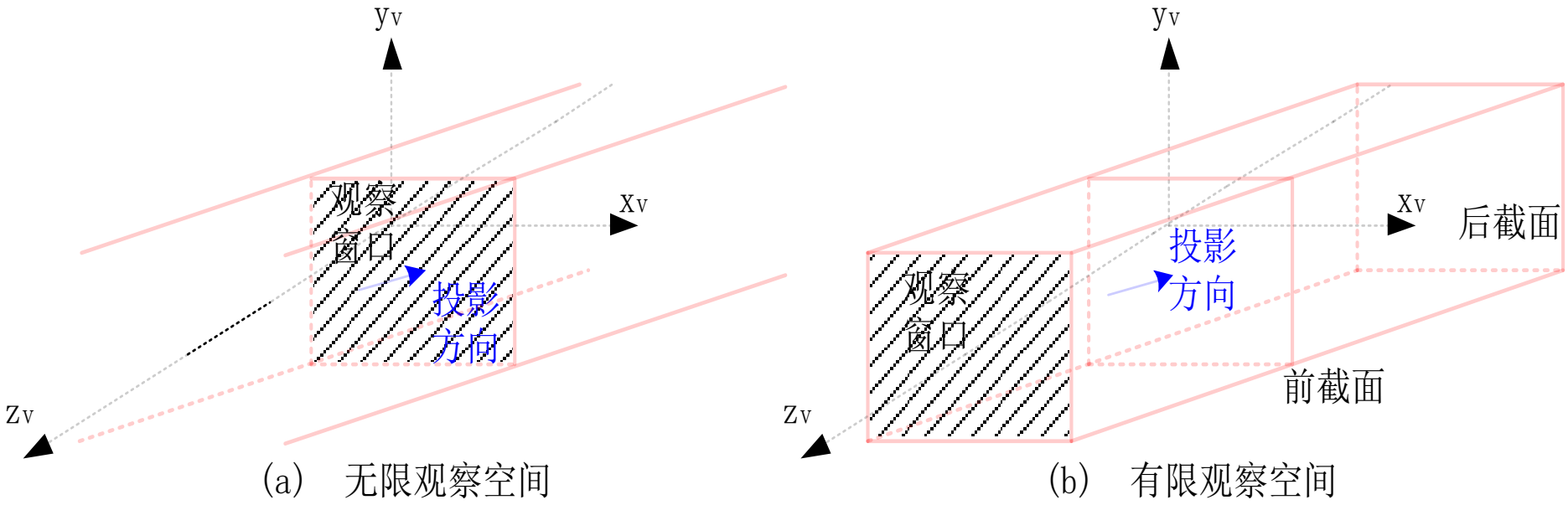


图7.34 斜投影观察空间

观察空间

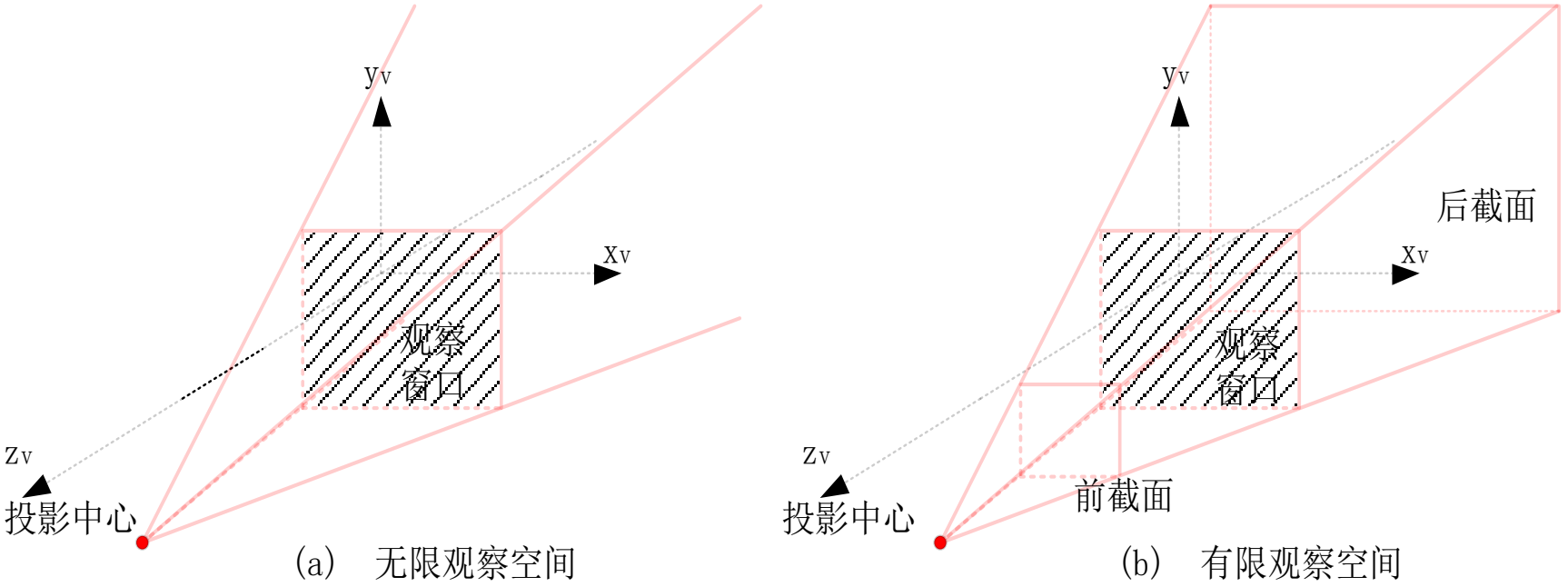


图7.35 透视投影观察空间

- 需注意，对于透视投影，**前截面**必须在投影中心和后截面之间。

观察空间

- 观察平面和前后截面的有关位置取决于要生成的窗口类型及特殊图形包的限制。

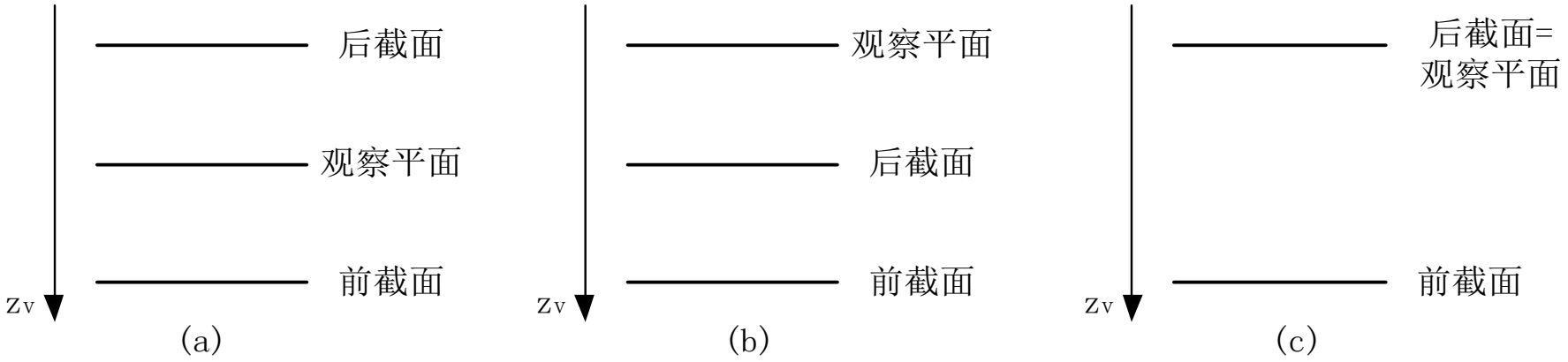
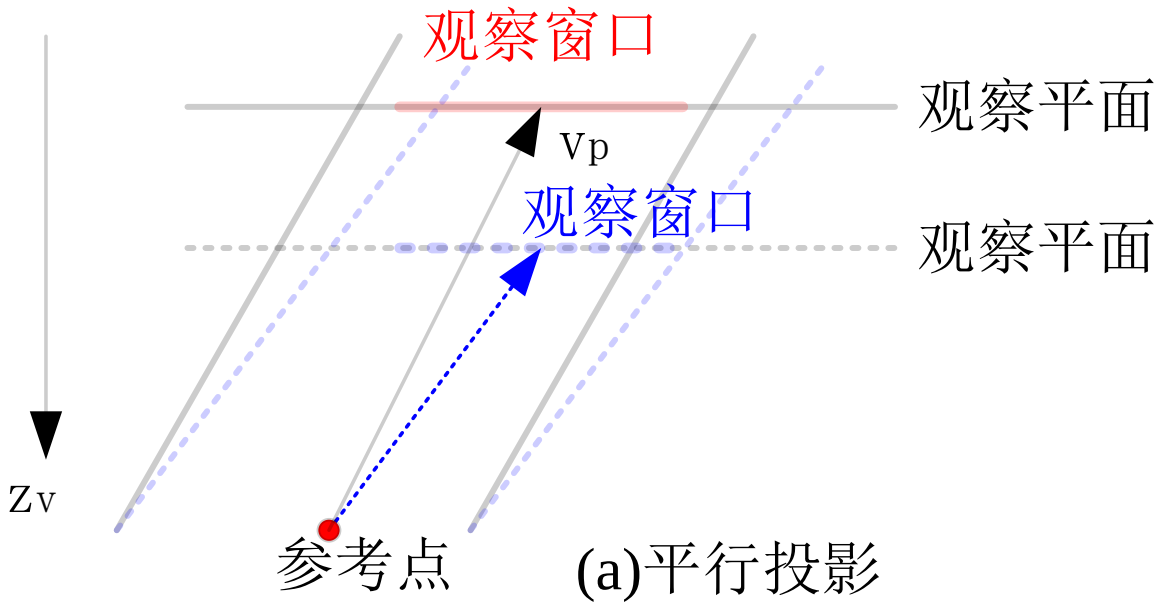
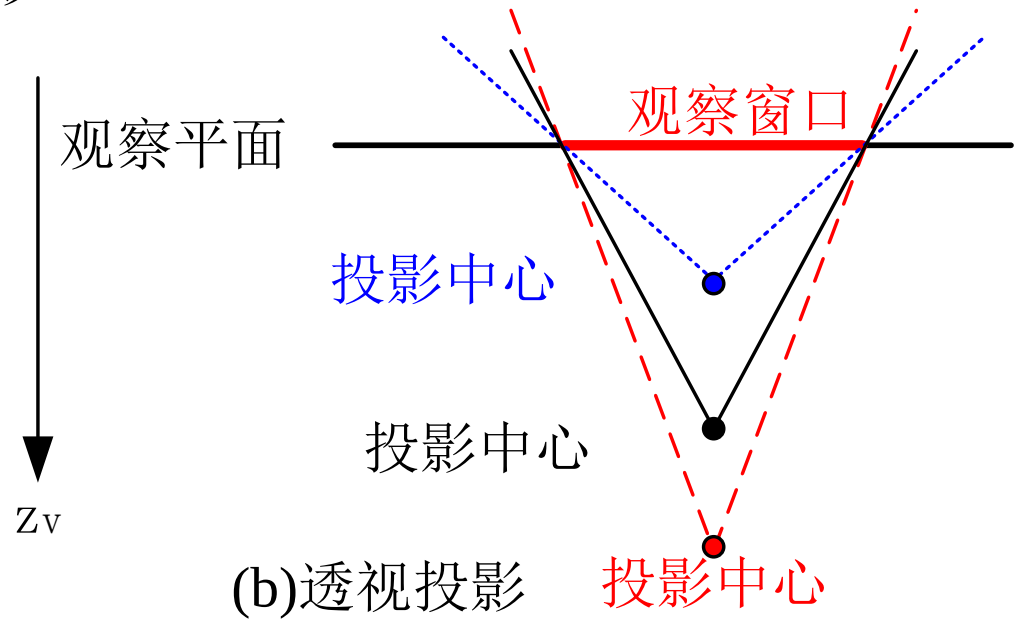


图7.36 观察平面的位置



(a)平行投影



(b)透视投影

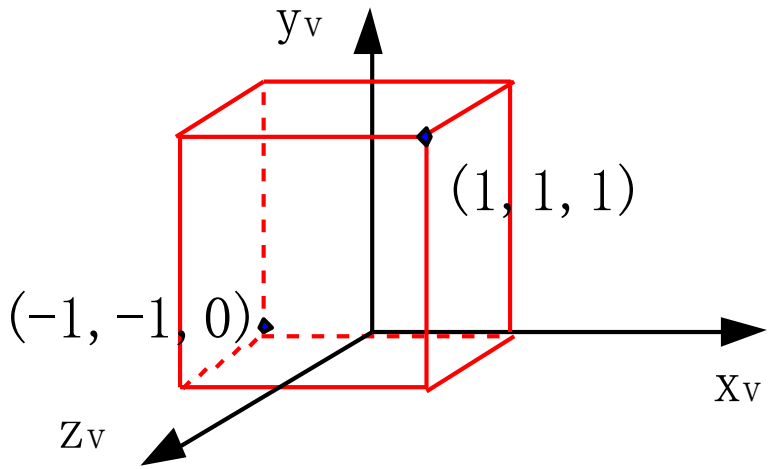
规范化观察空间

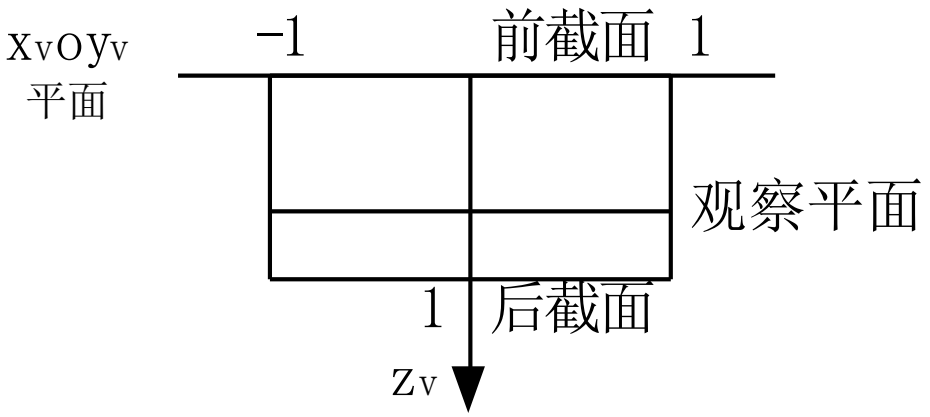
- 平行投影的规范化观察空间定义为：

$$x_v = 1, x_v = -1$$

$$y_v = 1, y_v = -1$$

$$z_v = 0, z_v = 1$$





(a) 平行投影的规范化观察空间

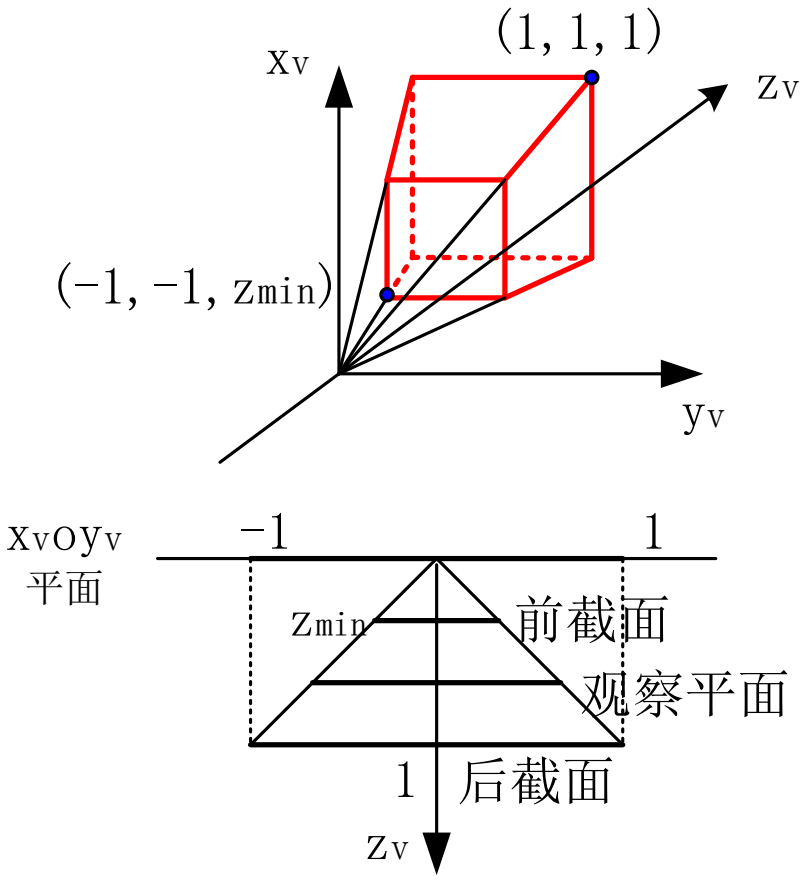
观察空间

- 透视投影的规范化观察空间为：

$$x_v = z_v, x_v = -z_v$$

$$y_v = z_v, y_v = -z_v$$

$$z_v = z_{\min}, z_v = 1$$



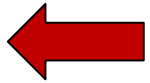
(b) 透视投影的规范化观察空间

问题

- 图形学的观察空间与照相机的观察空间的区别在于？

三维观察变换

1. 观察坐标系
2. 观察空间
3. 三维观察流程
4. 三维裁剪



三维观察流程

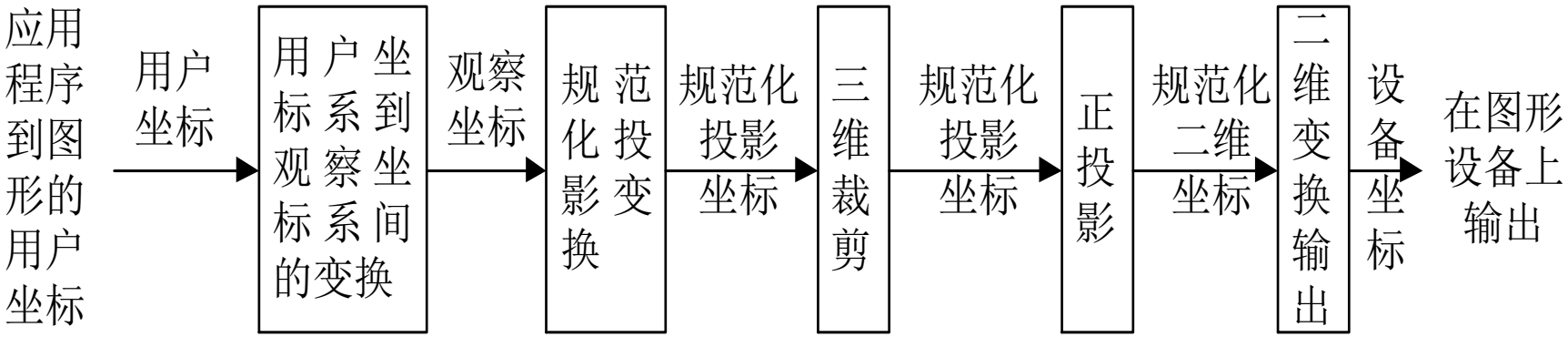


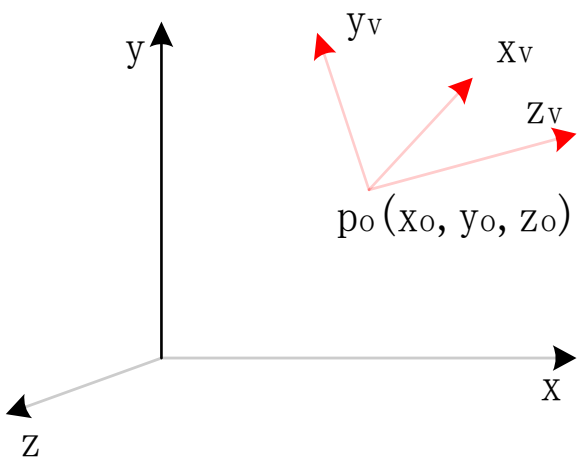
图7.37 三维观察流程

用户坐标系到观察坐标系变换

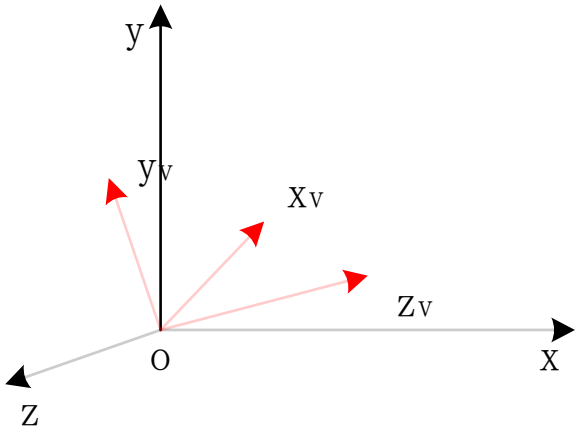
具体变换步骤：

(1) 平移观察参考点到用户坐标系原点；

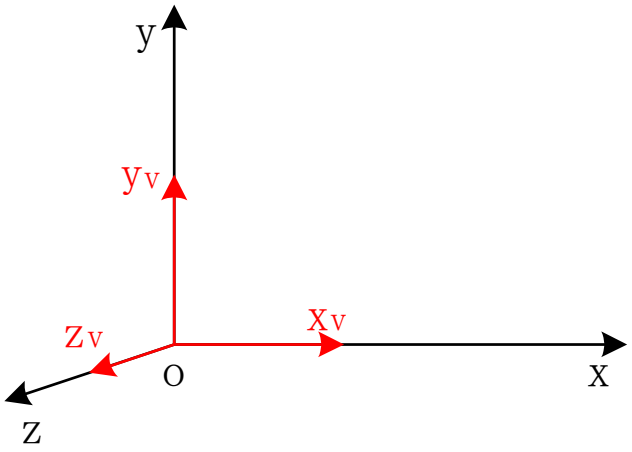
(2) 进行旋转变换分别让x_v、y_v和z_v轴对应到用户坐标系中的x、y和z轴。



(a) 用户坐标系与观察坐标系



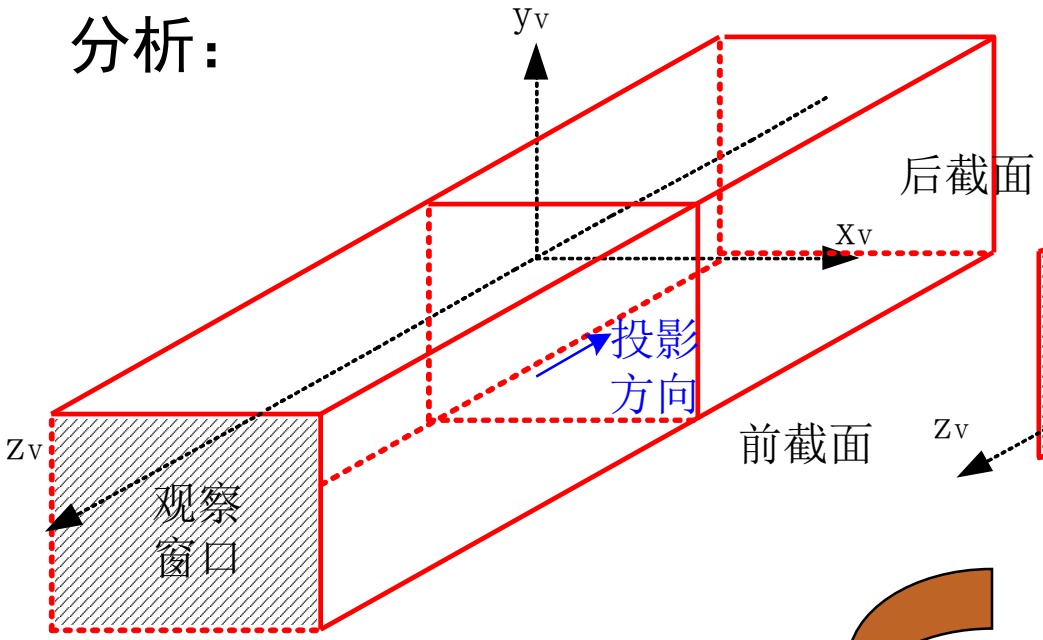
(b) 平移观察坐标系



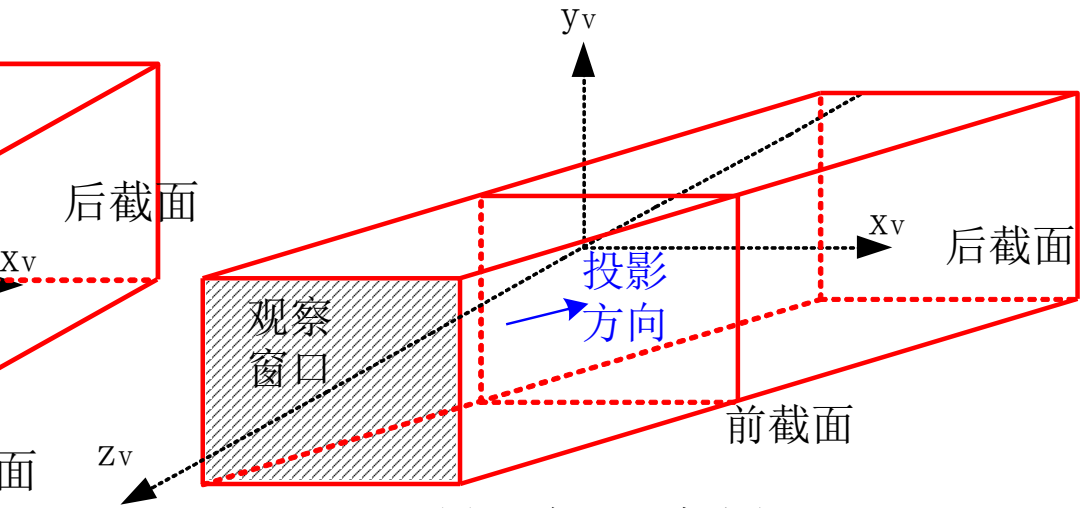
(c) 旋转观察坐标系

平行投影的规范化投影变换

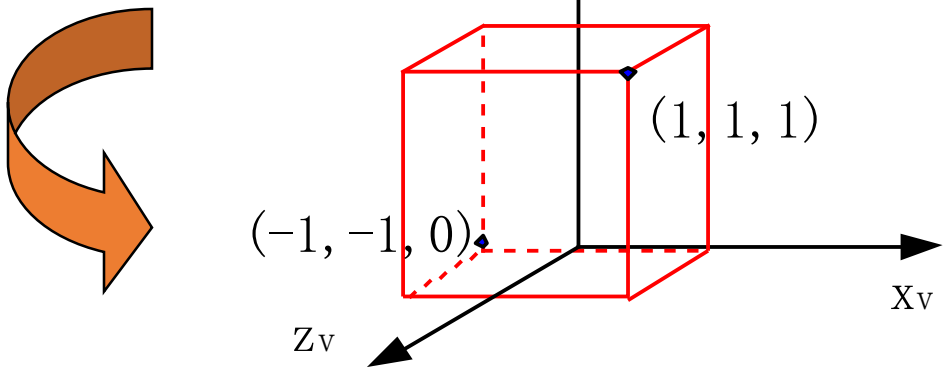
分析：



(b) 有限观察空间



(b) 有限观察空间



平行投影的规范化投影变换

- 观察窗口：左下角点（xw_{min}，yw_{min}）

  右上角点（xw_{max}，yw_{max}）

- 参考点：（x_{prp}，y_{prp}，z_{prp}）

- 前后截面：$Z=Z_{front}$，$Z=Z_{back}$

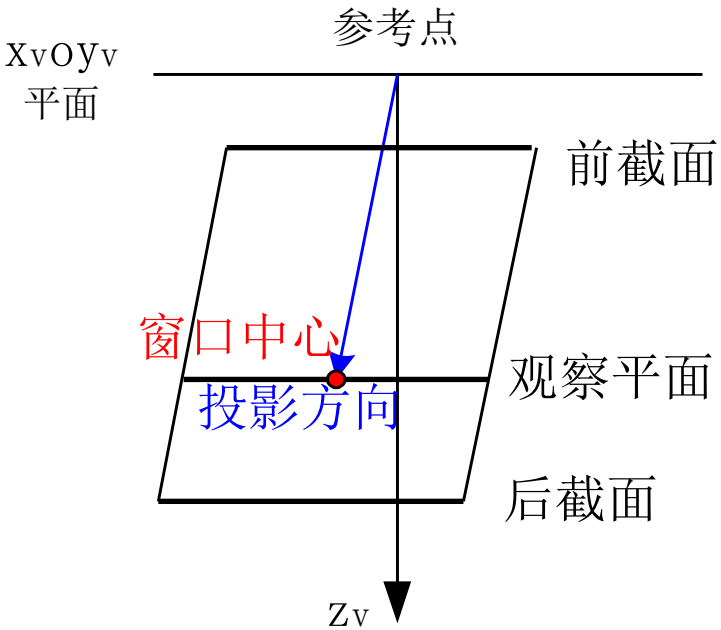
- 观察平面：$Z=Z_{vp}$

- 投影方向为从参考点到观察窗口中心点的坐标矢量。

平行投影的规范化投影变换

- 平行投影的规范化投影变换可由以下三步组成。

(1)将投影中心平移到观察坐标系原点；

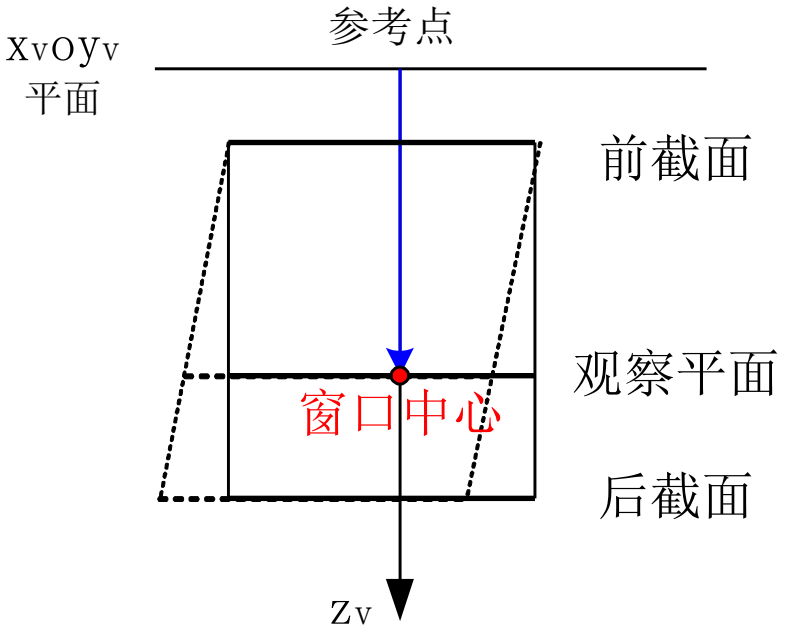


(a) 平移变换

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_{prp} & -y_{prp} & -z_{prp} & 1 \end{bmatrix}$$

平行投影的规范化投影变换

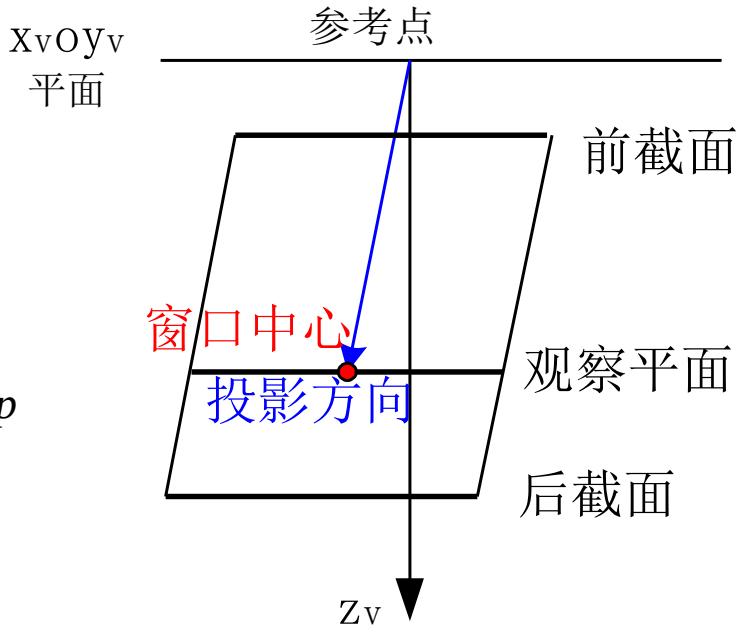
(2)对坐标系进行错切变换，使参考点和窗口中心的连线错切到z_v轴；



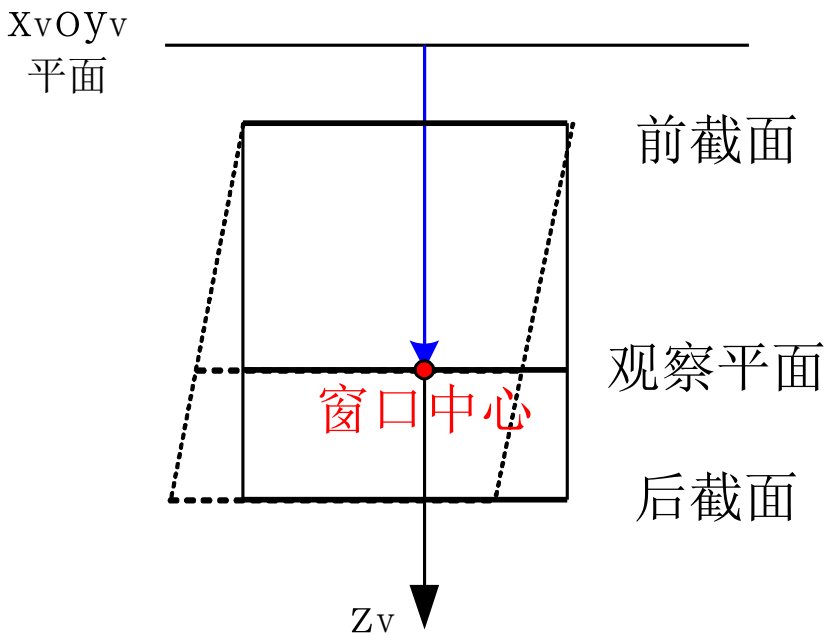
(b) 错切变换

口平移变换后，窗口中心点的坐标

$$\begin{cases} x_{cw} = (xw_{\min} + xw_{\max})/2 - x_{prp} \\ y_{cw} = (yw_{\min} + yw_{\max})/2 - y_{prp} \\ z_{cw} = z_{vp} - z_{prp} \end{cases}$$



(a) 平移变换



(b) 错切变换

解得：

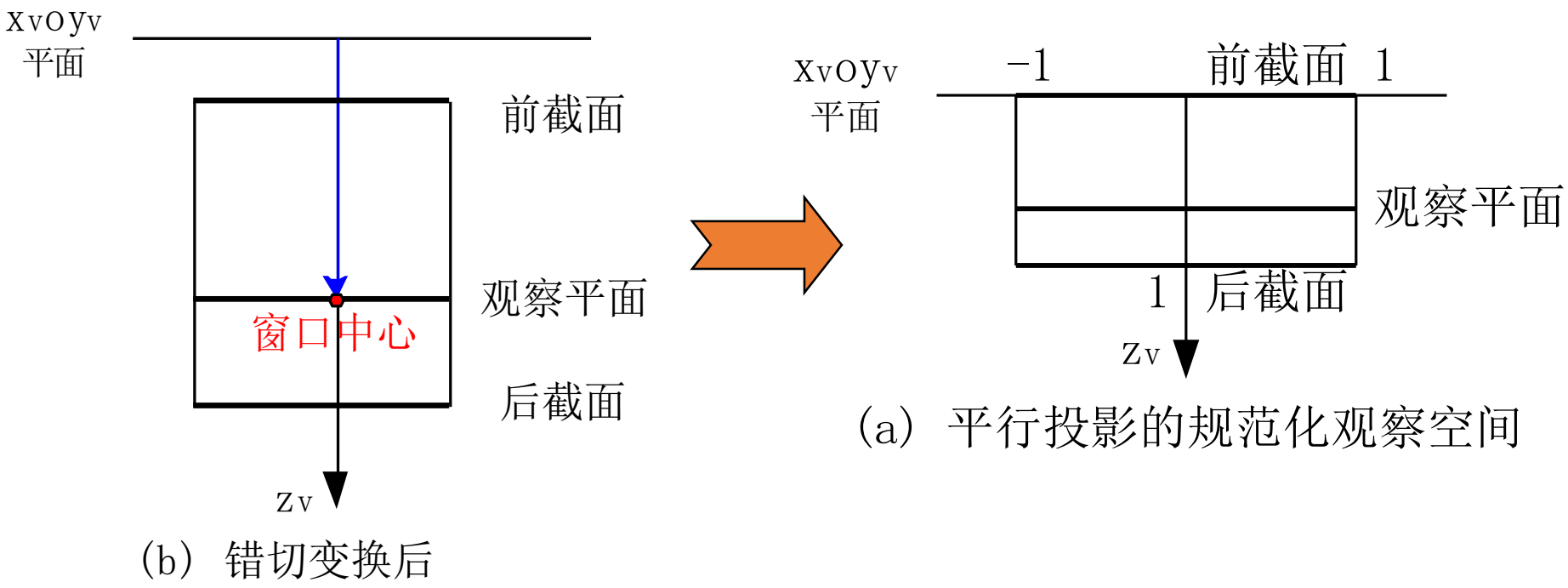
$$a = -x_{cw} / z_{cw}$$

$$b = -y_{cw} / z_{cw}$$

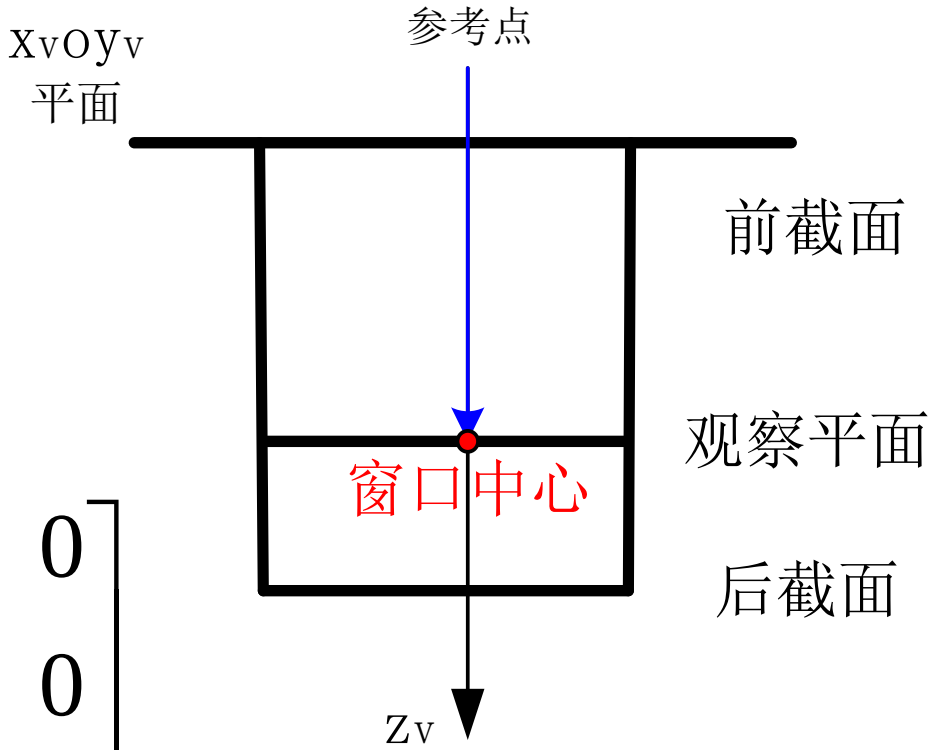
$$[0 \quad 0 \quad z_{cw} \quad 1] = [x_{cw} \quad y_{cw} \quad z_{cw} \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

平行投影的规范化投影变换

(3)进行坐标的归一化变换；

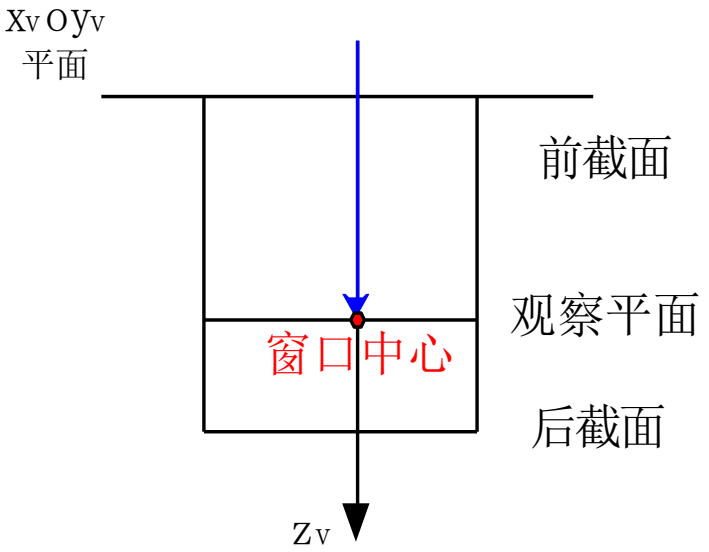


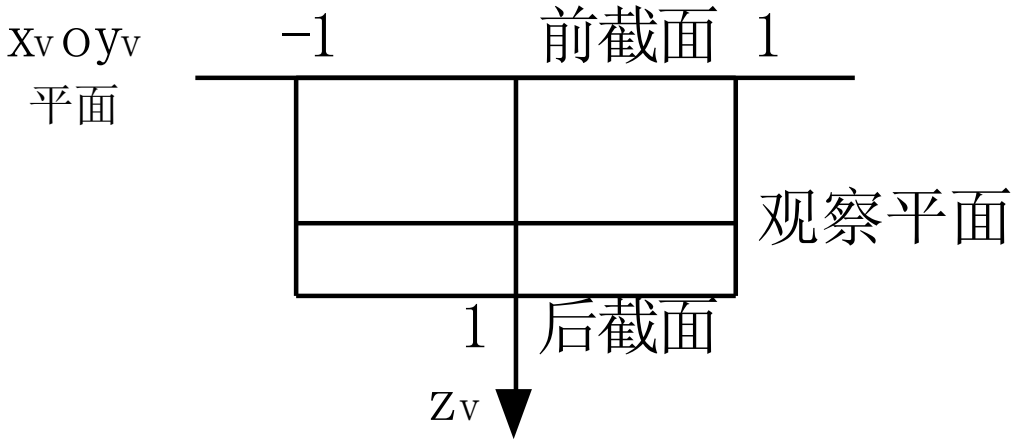
a) 平移变换



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -(z_{front} - z_{prp}) & 1 \end{bmatrix}$$

b）比例变换



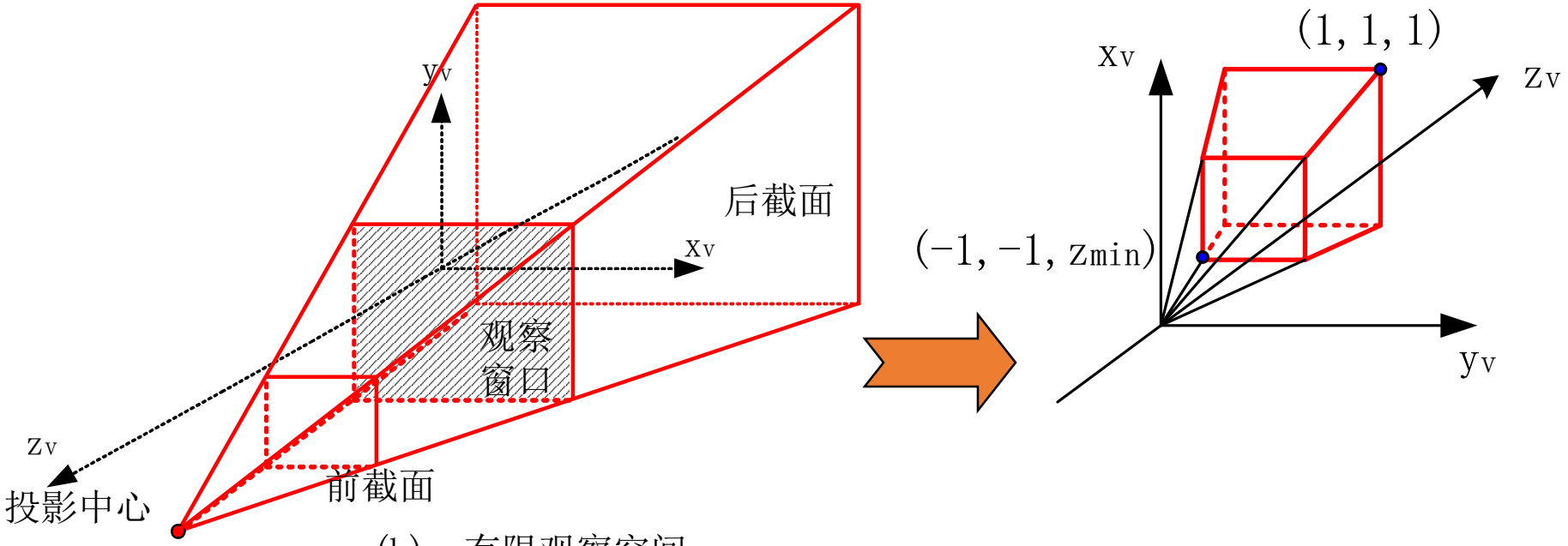


$$\begin{bmatrix} \frac{2}{xw_{\max}-xw_{\min}} & 0 & 0 & 0 \\ 0 & \frac{2}{yw_{\max}-yw_{\min}} & 0 & 0 \\ 0 & 0 & \frac{1}{z_{back}-z_{front}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

透视投影的规范化投影变换

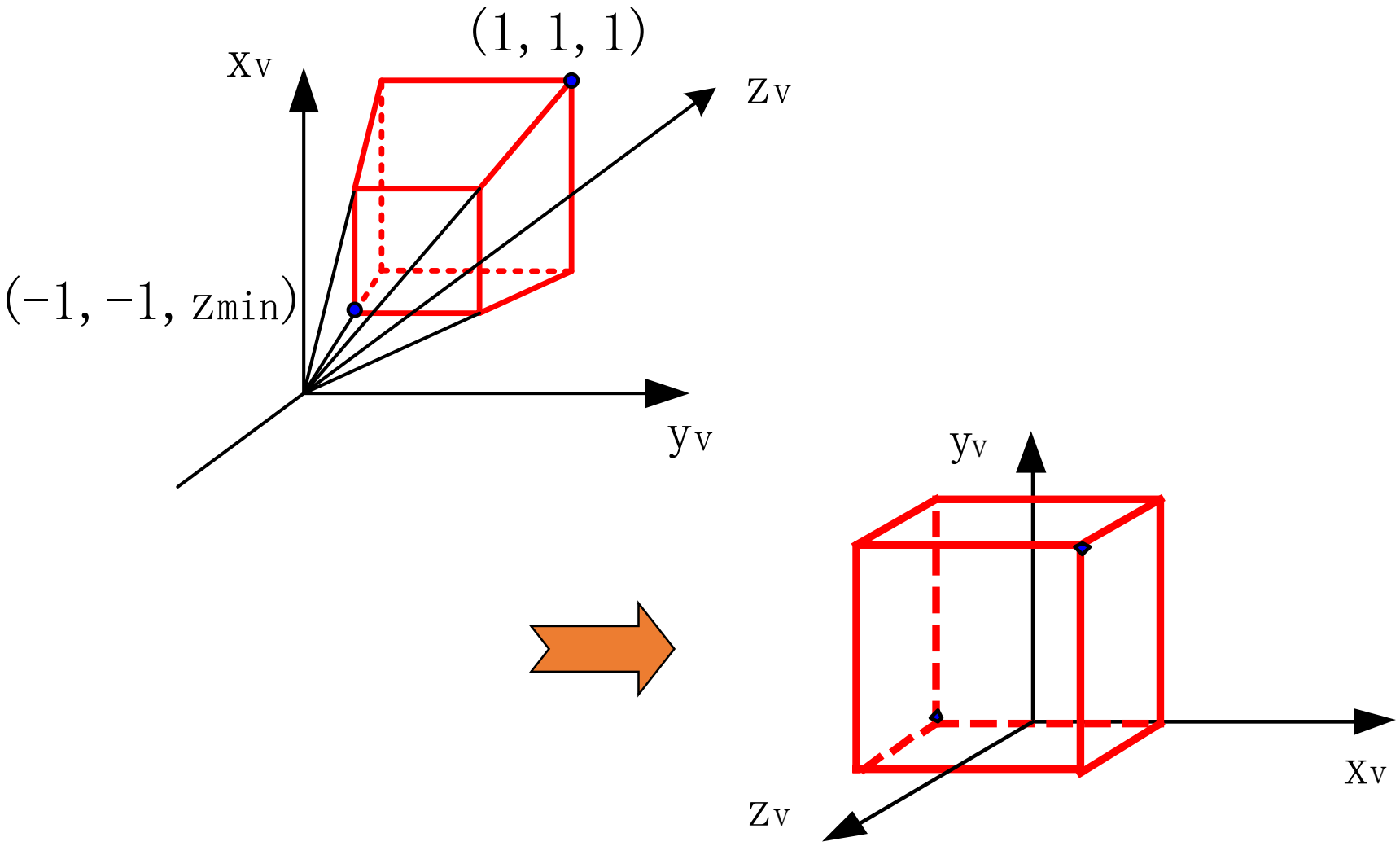
分析：透视投影的规范化投影变换分两步进行

(1)



(b)　有限观察空间

(2)



透视投影的规范化投影变换

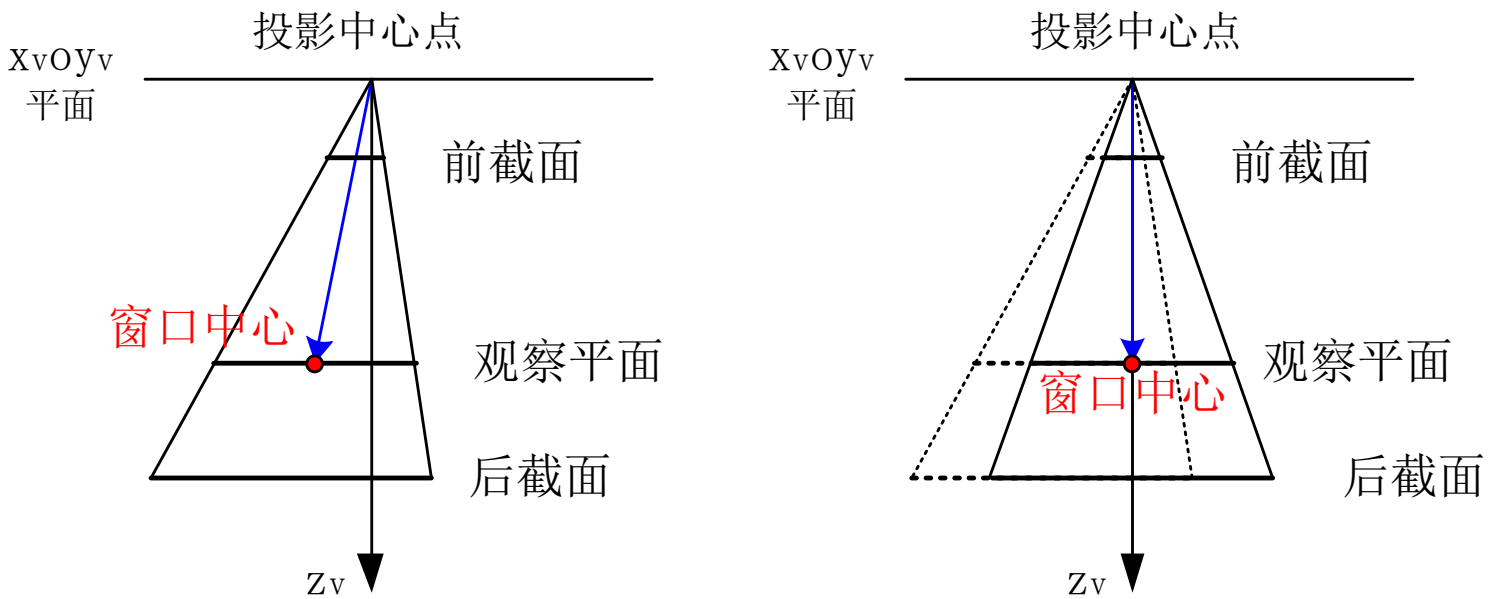
- **观察窗口：** 左下角点（xw_{min}，yw_{min}，0）

  右上角点（xw_{max}，yw_{max}，0）
- **参考点：** （x_{prp}，y_{prp}，z_{prp}）
- **前后截面：** $Z=Z_{front}$，$Z=Z_{back}$
- **观察平面：** $Z=Z_{vp}$

变换步骤：

(1) 将投影中心平移到观察坐标系原点

(2) 对坐标系进行错切变换



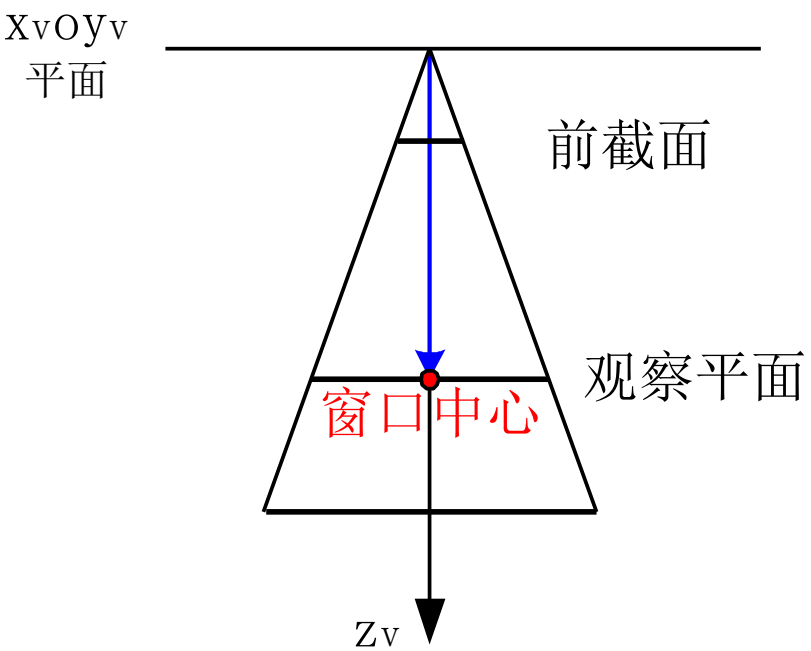
(a) 平移变换　　(b) 错切变换

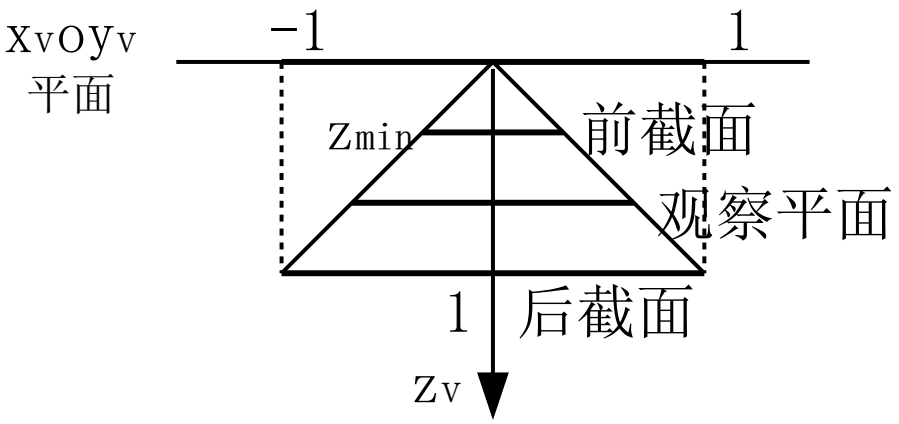
透视投影的规范化投影变换

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{x_{cw}}{z_{cw}} & -\frac{y_{cw}}{z_{cw}} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

透视投影的规范化投影变换

(3) 进行比例变换。

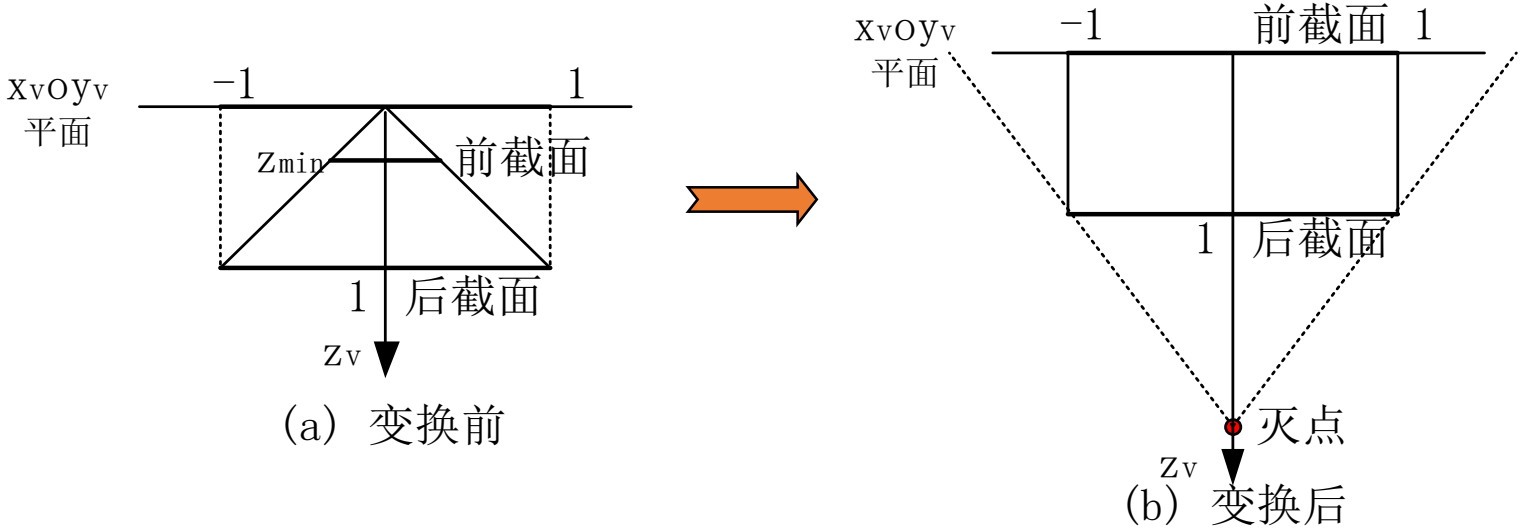




$$\begin{bmatrix} \frac{2}{xw_{\max}-xw_{\min}}\cdot\frac{z_{vp}-z_{prp}}{z_{back}-z_{prp}} & 0 & 0 & 0 \\ 0 & \frac{2}{yw_{\max}-yw_{\min}}\cdot\frac{z_{vp}-z_{prp}}{z_{back}-z_{prp}} & 0 & 0 \\ 0 & 0 & \frac{1}{z_{back}-z_{prp}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

透视投影的规范化投影变换

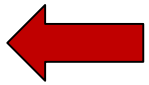
(4) 将图7-34(b)所示的透视投影的规范化观察空间变换为图7-34(a)的平行投影的规范化观察空间。



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{1}{1-z_f} & 0 \\ 0 & 0 & -\dfrac{z_f}{1-z_f} & 0 \end{bmatrix}$$

三维观察变换

1. 观察坐标系
2. 观察空间
3. 三维观察流程
4. 三维裁剪



三维裁剪

- 三维裁剪保留所有在观察空间内的图形以便在输出设备中显示，所有在观察空间外的图形被丢弃。
- 三维直线段的裁剪
- 多边形面的裁剪

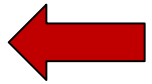
三维裁剪

四维齐次坐标表示的图形裁剪：

- 将齐次坐标转换为三维坐标，在三维空间中关于规范化观察空间剪裁；
- 直接在齐次坐标空间中进行裁剪。

OpenGL中的变换

1. 变换种类
2. 模型视图矩阵
3. 矩阵操作
4. 投影变换

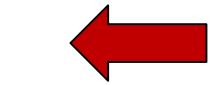


变换种类

- 视图变换：指定观察者或摄影机的位置；
- 模型变换：在场景中移动对象；
- 模型视图变换：描述视图变换与模型变换的对偶性；
- 投影变换：对视见空间进行修剪和改变大小；
- 视见区变换：对窗口的最终输出进行缩放；

OpenGL中的变换

1. 变换种类
2. 模型视图矩阵
3. 矩阵操作
4. 投影变换



模型视图矩阵

- 平移

  void glTranslated(f)(GLdouble x，GLdouble y，GLdouble z)；

- 旋转

  void glRotated(f)(GLdouble angle，GLdouble x，GLdouble y，GLdouble z)；

- 比例

  void glScaled(f)(GLdouble x，GLdouble y，GLdouble z)；

模型视图矩阵

- 视图变换函数（定义观察坐标系）

void gluLookAt（GLdouble eyex，GLdouble eyey，GLdouble eyez，GLdouble centerx，GLdouble centery，GLdouble centerz，GLdouble upx，GLdouble upy，GLdouble upz）；

模型视图矩阵

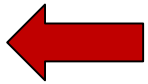
- OpenGL中使用列向量矩阵

$$p'=\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=T_{3D}\cdot p=\begin{bmatrix}a&d&h&l\\b&e&h&m\\c&f&j&n\\p&q&r&s\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

$$P'=T\cdot P=(T_n\cdot T_{n-1}\cdot T_{n-2}\cdots T_1)\cdot P\qquad(n>1)$$

OpenGL中的变换

1. 变换种类
2. 模型视图矩阵
3. 矩阵操作
4. 投影变换



矩阵操作

glMatrixMode(GLenum mode)；

参数mode用于确定将哪个矩阵堆栈用于矩阵操作。

GL_MODELVIEW：模型视图矩阵堆栈

GL_PROJECTION：投影矩阵堆栈

GL_TEXTURE：纹理矩阵堆栈

投影变换

- OpenGL中只提供了两种投影方式，一种是平行投影（正射投影），另一种是透视投影。在投影变换之前必须指定当前处理的是投影变换矩阵：

```
glMAtrixMode(GL_PROJECTION);
glLoadIdentity();
```

投影变换

- 平行投影：视景体是一个矩形的平行管道，也就是一个长方体，其特点是无论物体距离相机多远，投影后的物体大小尺寸不变。

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

投影变换

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

- **一个特殊的正射投影函数，主要用于二维图像到二维屏幕上的投影。其near和far缺省值分别为-1.0和1.0，所有二维物体的Z坐标都为0.0。因此它的裁剪面是一个左下角点为（left，bottom）、右上角点为（right，top）的矩形。**

透视投影

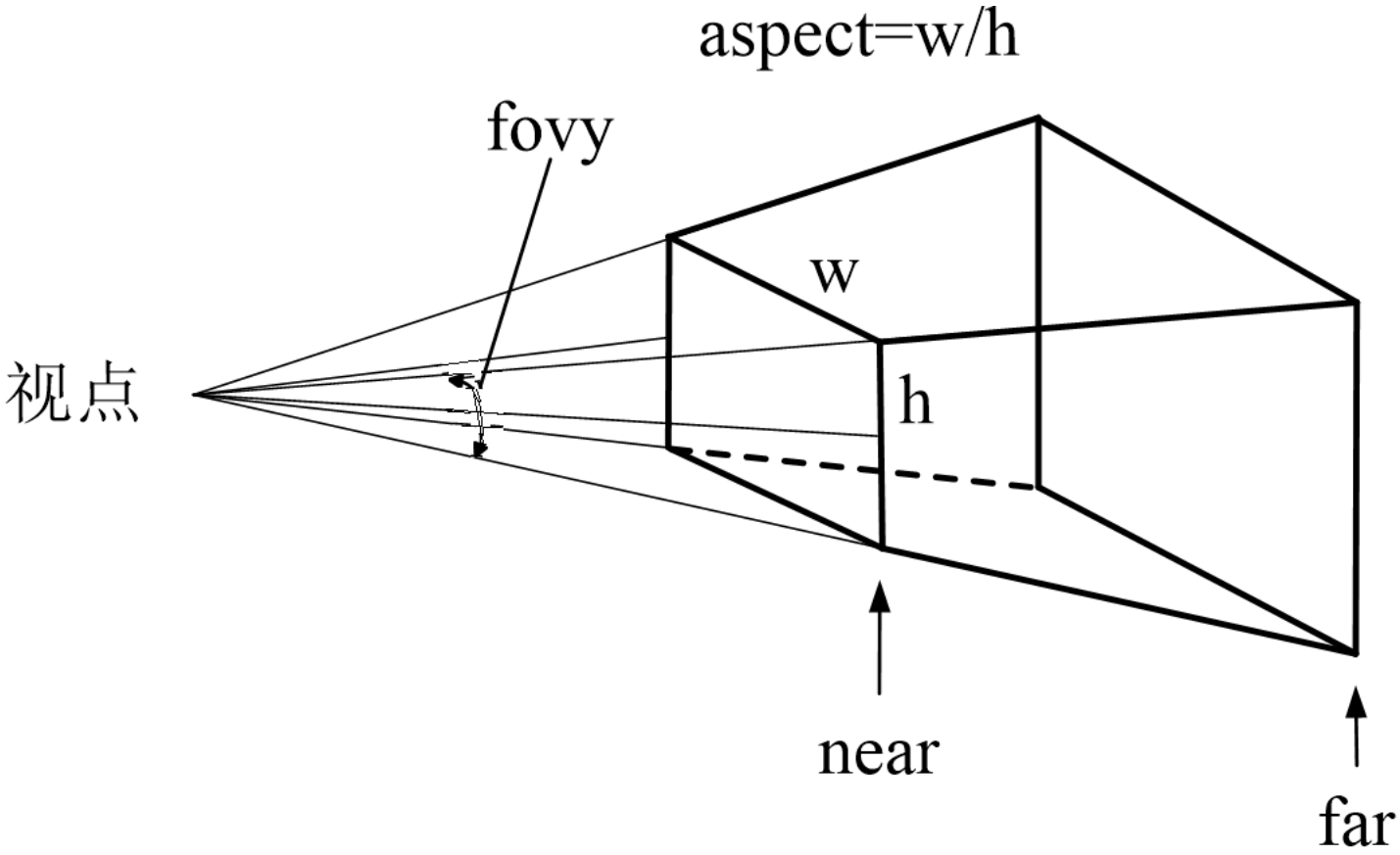
void glFrustum (GLdouble left,GLdouble Right,GLdouble bottom,GLdouble top,GLdouble near,GLdouble far);

此函数创建一个透视投影矩阵，并且用这个矩阵乘以当前矩阵。它的参数只定义近裁剪平面的左下角点和右上角点的三维空间坐标，即（left，bottom，-near）和（right，top，-near）；最后一个参数far是远裁剪平面的Z负值，其左下角点和右上角点空间坐标由函数根据透视投影原理自动生成。

void gluPerspective (GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);

它也创建一个对称透视视景体，但它的参数定义于前面的不同，其操作是创建一个对称的透视投影矩阵，并且用这个矩阵乘以当前矩阵。参数fovy定义视野Y-Z平面Y方向张开的角度，范围是[0.0，180.0]；参数aspect是投影平面的纵横比（宽度与高度的比值）；参数zNear和zFar分别是远近裁剪面沿Z负轴到视点的距离。

投影变换



fovy: Specifies the field of view angle, in degrees, in the y direction.

高级矩阵操作

```
glfloat m[] = { 1.0f, 0.0f, 3.0f, 0.0f,
                0.0f, 1.0f, 0.0f, 1.0f,
                0.0f, 0.0f, 1.0f, 1.0f,
                0.0f, 0.0f, 0.0f, 1.0f, };
glMatrixMode(GL_MODELVIEW);
glLoadMatrixf(m);
   glMultiMatrixf(m);
```

作业

7.14 上传代码到s.ecust.edu.cn