2018-2019年度第二学期 00106501

计算机图形学

童 伟 华　管理科研楼1205室

E-mail: tongwh@ustc.edu.cn

中国科学技术大学　数学科学学院

http://math.ustc.edu.cn/

第二节　计算机视图

计算机视图

- 视图有三个功能，都在流水线体系中实现
  - 定位照相机
    - 设置模型－视图矩阵
  - 设置镜头
    - 设置投影矩阵
  - 裁剪
    - 设置视景体

合成照相机

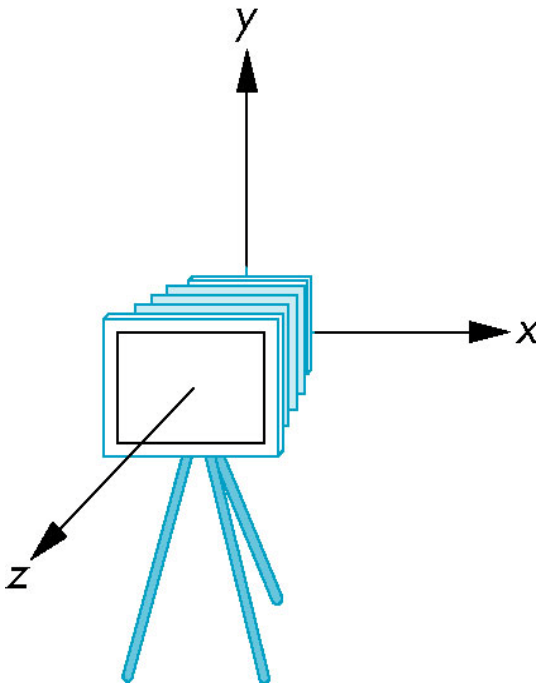
- 计算机视图是基于合成照相机的，原则上可以实现所有的经典视图
- 所有的经典视图是基于对象、观察者和投影线之间的紧密联系的，而在计算机图形学中强调的则是对象定义与照相机定义之间的独立性
- 在OpenGL中可以指定采用的是透视投影或者正交投影，但在透视投影中OpenGL并不知道什么是单点、两点或三点透视
  - 为了实现这些细节需要知道对象与照相机之间的关系

最终的选择

- 偏好对象定义与照相机定义之间的独立性
- 如果应用程序需要特定类型的视图，那么应当仔细确定照相机相对于对象的位置
  - 因为在透视投影中，从射影几何的角度来说，肯定有三个灭点，只是有些灭点在无穷远

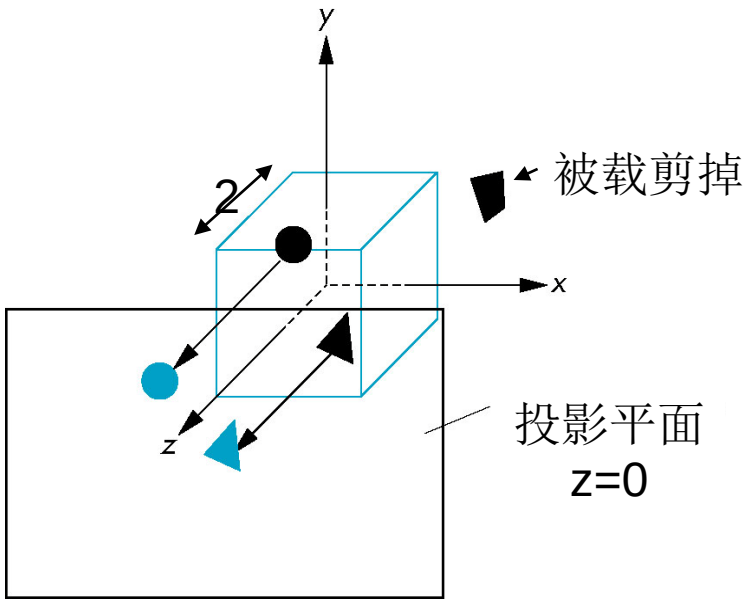
OpenGL中的照相机

- 在OpenGL中，初始的世界标架和照相机标架相同
  - 初始的模型－视图矩阵是单位阵
- 照相机位于原点，并指向z轴的负向
- OpenGL也指定了默认的视景体，它是一个中心在原点的边长为2的立方体
  - 缺省的投影矩阵是单位阵



缺省投影

- 默认的投影是正交投影

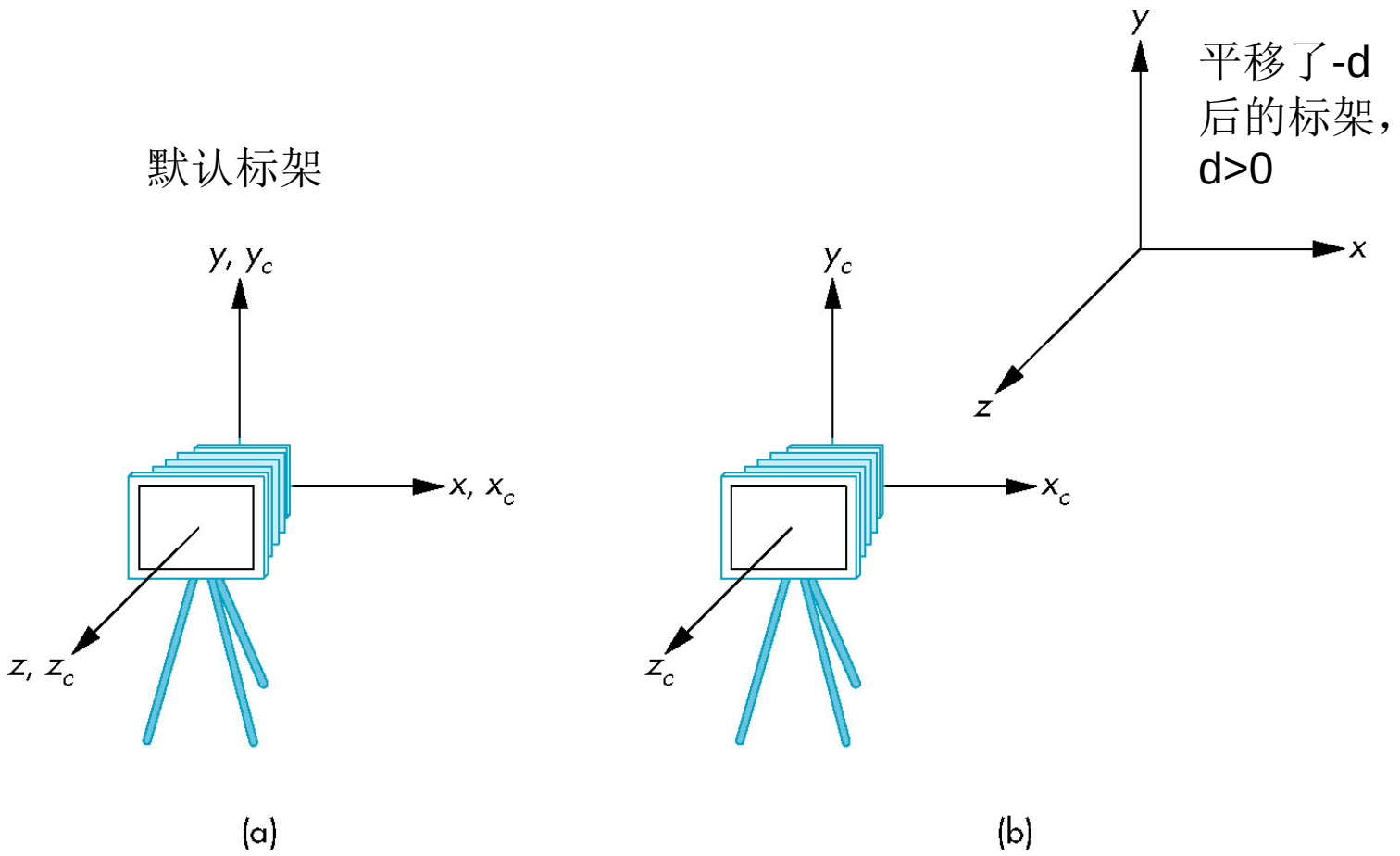


- 在缺省的照相机设置下，为了使定义的对象可见，只要使对象的位置和尺寸与默认视景体相匹配
  - 通常可以对数据进行适当的平移和各向同性放缩

移动照相机标架

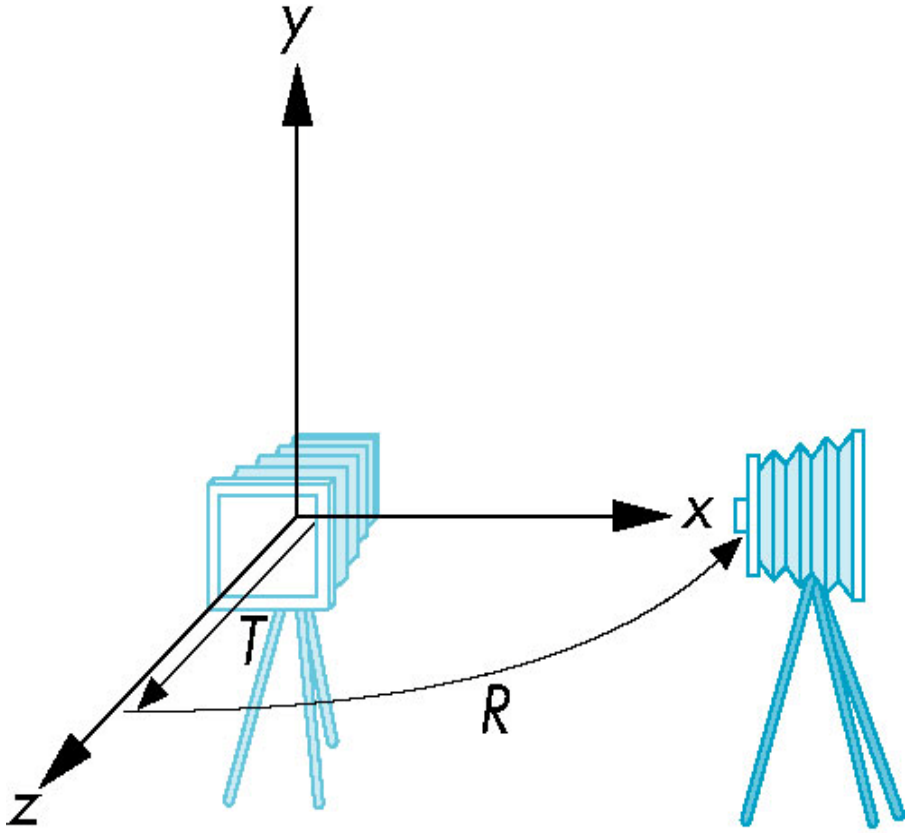
- 如果想看到具有更大的正z坐标的对象，我们可以
  - 把照相机沿z轴正向移动
    - 平移照相机标架
  - 把对象沿z轴负向移动
    - 移动世界标架
- 两者是完全等价的，都是由模型－视图矩阵确定的
  - 需要平移glTranslated(0.0, 0.0, -d);
  - 此处d＞0

移动照相机标架



移动照相机标架

- 可以利用一系列旋转和平移把照相机定位到任意位置
- 例如，为了得到侧视图
  - 旋转照相机: R
  - 把照相机从原点移开: T
  - 模型－视图矩阵C = TR



OpenGL代码

- 注意最后指定的变换是最先被应用的变换

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslated(0.0, 0.0, -d);
glRotated(-90.0, 0.0, 1.0, 0.0);
```

模型－视图矩阵

- 模型－视图矩阵是OpenGL状态的一部分
- 任何时刻的模型－视图矩阵包含了照相机标架与世界标架的位置关系
- 虽然表面上看把模型与视图矩阵结合为一个矩阵会导致一些混淆，但仔细体会这种流水线体系就会发现其中的优势
- 如果把照相机也看作具有几何属性的对象，那么改变对象位置和定向的变换当然对照相机的位置和定向相对于其它对象也发生改变
- 可以认为在定义真正对象之前的模型－视图变换是定位照相机

如何构造等角投影

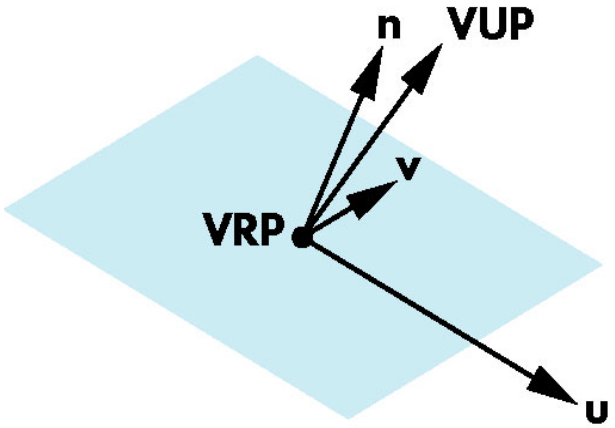
- 假设从中心在原点的立方体开始，立方体平行于坐标轴
- 希望移动照相机得到该立方体的等角投影
  - 首先绕y轴旋转45度
  - 然后绕x轴旋转35.26度
  - 最后从原点移开
- 构造方法复杂

两个视图API

- 为了实现某种投影，需要经过复杂的计算得到变换的构成
- 可以采用在PHIGS和GKS-3D中的方法定位照相机
- 在世界标架中描述照相机的位置
- 投影的类型是由在OpenGL中等价的投影矩阵确定的
  - 视图过程的这部分操作也称为规范化变换 (normalization transformation)
  - 用标架中的变换实现

照相机参数

- 照相机的默认位置在原点，指向z轴负向
- 所期望的位置称为视图参考点(view-reference point, VRP)
- 照相机定向
  - 视图平面法向n(view-plane normal, VPN)
  - VUP (view-up vector)

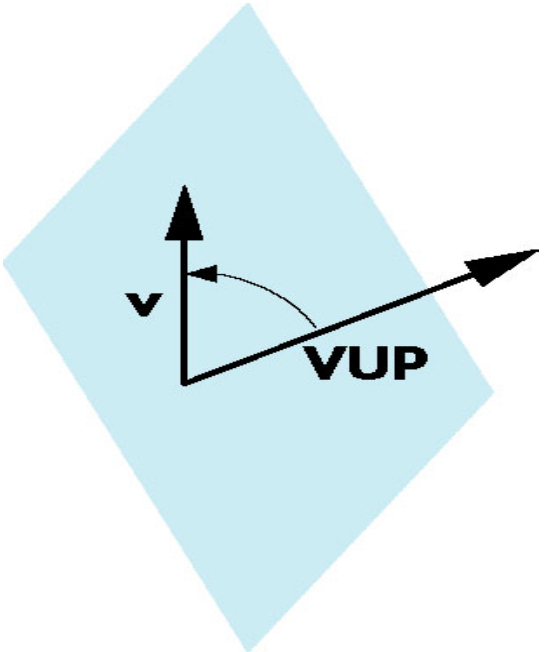


VPN & VUP

- VPN给出投影面的方向，即平面的法向
- 只有平面的定向不能完全确定照相机的定向
  - 照相机还可以绕VPN方向旋转
- 只有给出了VUP，才完全确定了照相机的方向

VUP

- 不必要求VUP向量必定平行于投影面
- 把VUP投影到投影平面上得到上方向量v
  - v与n正交
  - 设u = v x n
  - 由(u, v, n)构成的直角坐标系称为视图坐标系
  - 加上VRP，构成照相机标架
  - 实现这种变换的矩阵称为视图定向矩阵



视图定位矩阵

- 设VRP点$p = [x,y,z,1]^T$，视平面法向$n = [n_x,n_y,n_z,0]^T$，上方向量为$v_{up} = [v_{upx}, v_{upy}, v_{upz}, 0]^T$.
- $v = v_{up} - (v_{up} \cdot n)/(n \cdot n)n$，把v，n单位化
- $u = v \times n$
- 视图定位矩阵为

$$\mathbf{V} = \begin{bmatrix} u_x & u_y & u_z & -xu_x - yu_y - zu_z \\ v_x & v_y & v_z & -xv_x - yv_y - zv_z \\ n_x & n_y & n_z & -xn_x - yn_y - zn_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

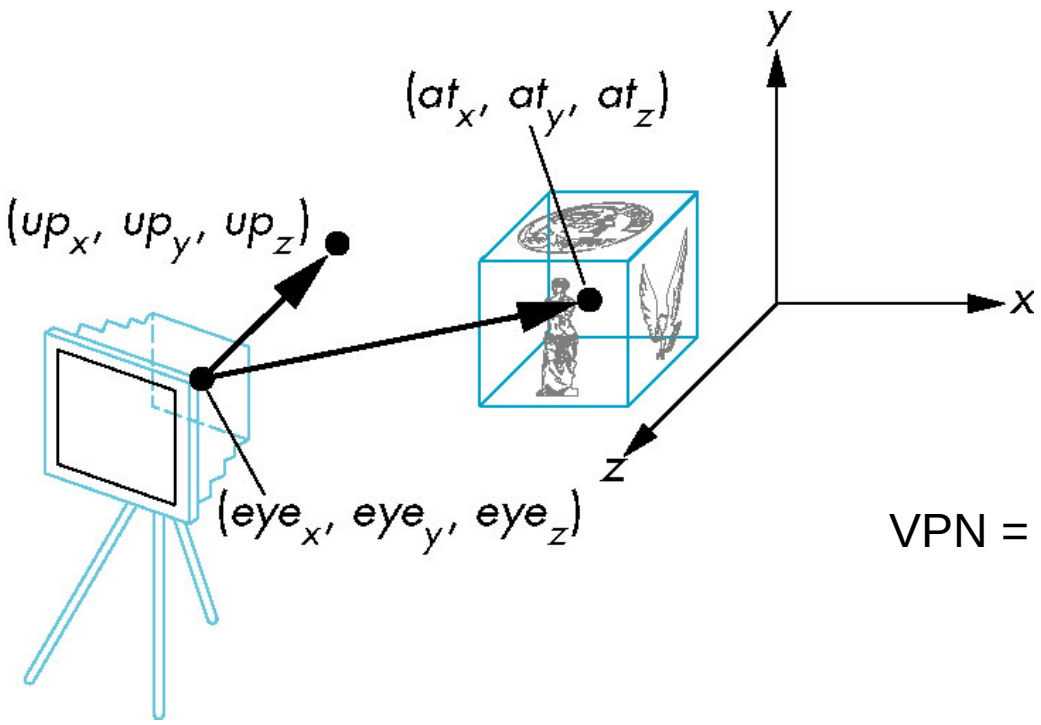
LookAt()函数

- 在GLU库中包含了函数gluLookAt()，提供了创建定位照相机所用的模型－视图矩阵的简单方法
- 注意在设置中需要一个向上的方向
- 需要初始化，即上载单位阵
  - 也可以与模型变换复合在一起
- 例如：平行于轴的立方体的等角投影

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(1.,1.,1.,0.,0.,0.,0.,1.,0.);
```

gluLookAt()

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



VPN = eye - at

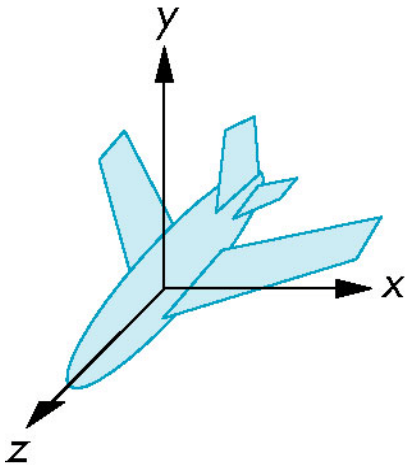
gluLookAt()与其它变换

- 用户可以自己定义模型－视图矩阵实现同样的功能
- 但是从概念上可以把gluLookAt()作为照相机的定位，而把后续的其它变换作为对象的定位

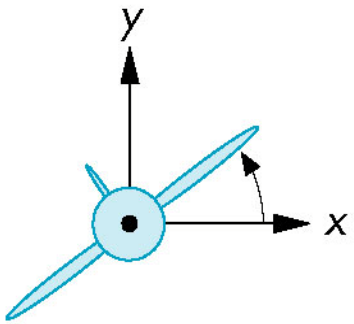
其它视图API

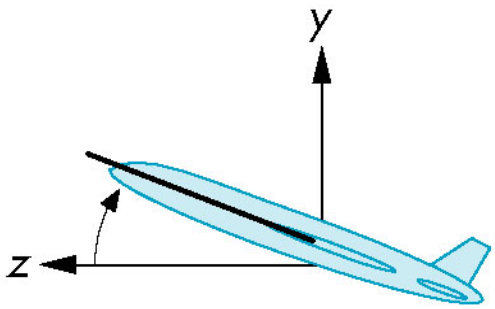
- 在OpenGL中gluLookAt()函数是唯一的专门用来定位照相机的函数
- 在许多实际应用中，已有的视图定义方法不适用
  - 例如飞机操纵模拟

飞机操纵模拟

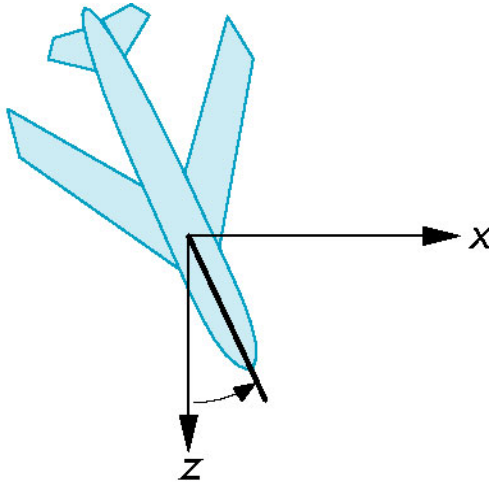


侧滚
Roll





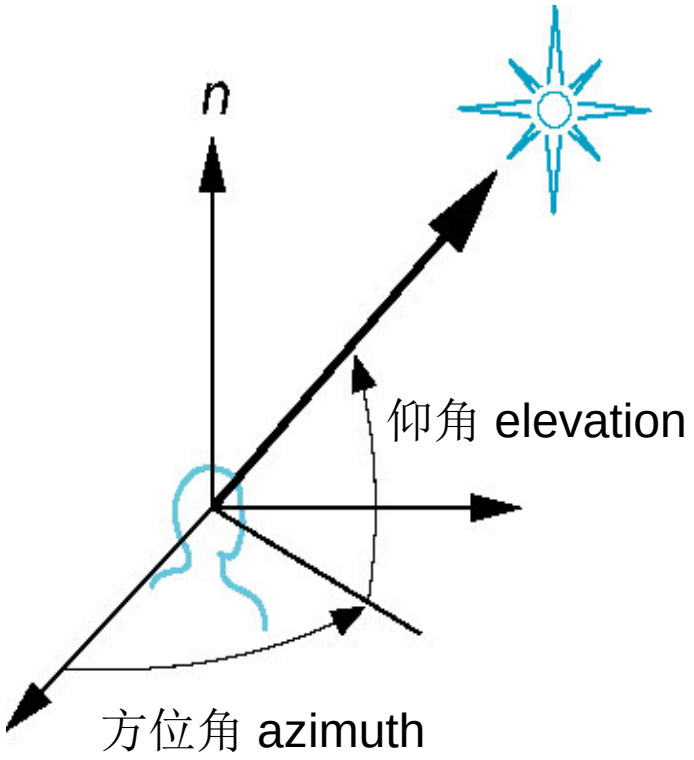
倾斜
Pitch



偏转
Yaw

星空观察

- 采用极坐标
- 点在天空中的位置指定
  - 方向由方位角和仰角定义
  - 观察者所在的平面，即法向n
  - 照相机还可以绕方向旋转，称为扭转角



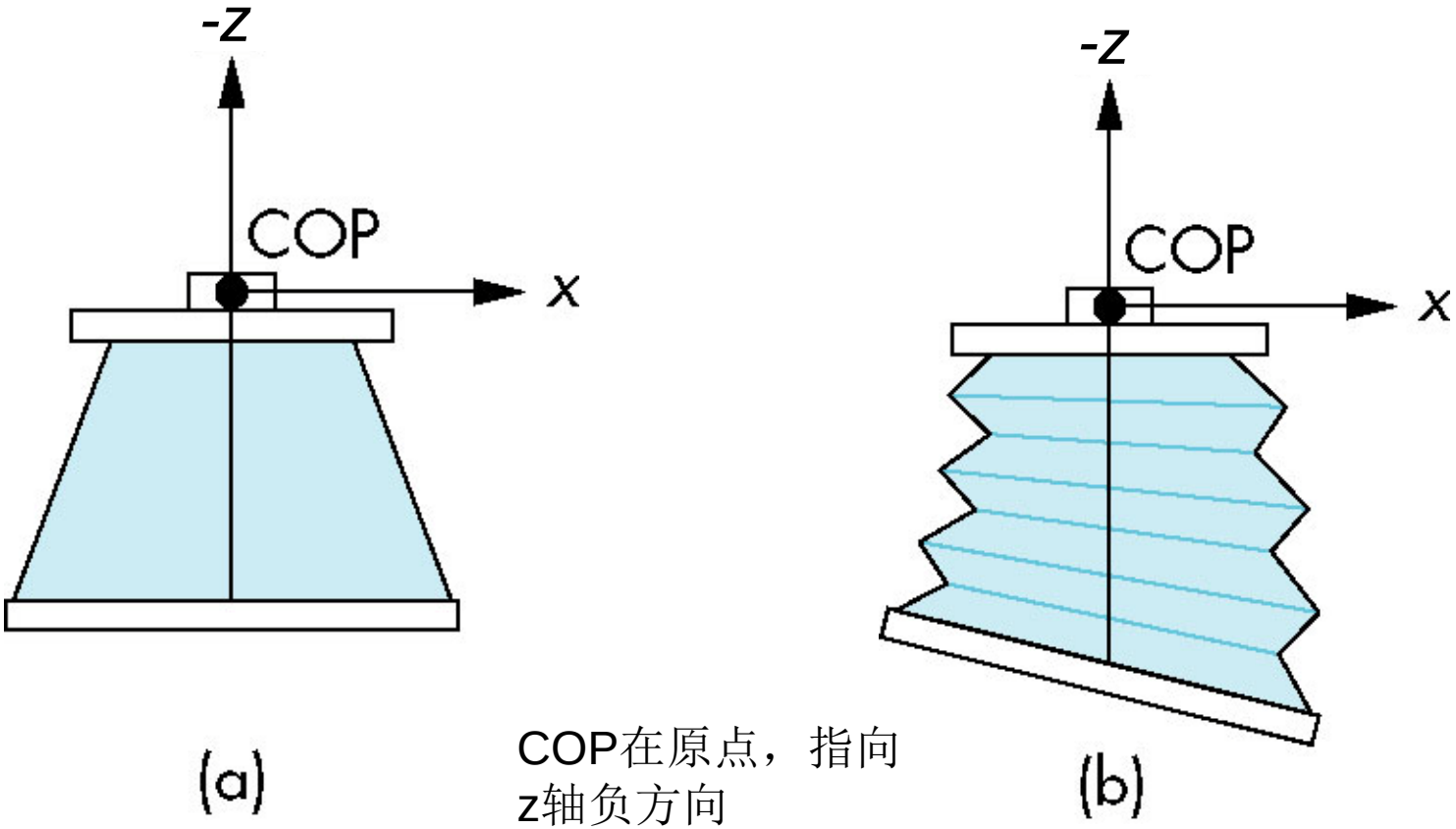
物理照相机

- 在定位后仍然可以选择镜头
- 镜头与胶卷的大小结合在一起确定在照相机前面多大范围的对象出现在最终的照片上
- 宽角镜头可以使离照相机近的对象看起来比离照相机远的对象夸张得大，远距镜头则会近似得到平行投影的效果

计算机中的照相机

- 可以选择投影的类型和视图参数模拟镜头的远近
- 绝大多数API提供不同的函数用于定义平行投影和透视投影
- OpenGL就是如此，虽然两者是在同一个流水线体系中实现
  - 用glLoadMatrix设置投影矩阵

两种透视投影模型



COP在原点，指向z轴负方向

投影与规范化

- 在视点（照相机）标架中默认的投影是正交投影
- 对于在默认视景体内的点，$x_p = x, y_p = y, z_p = 0$
- 大多数图形系统应用视图规范化的过程
  - 通过由投影矩阵确定的变换把所有其它的视图转化为默认视图
  - 从而可以对所有的视图采用同样的流水线体系

齐次坐标表示

- 默认的正交投影矩阵

$$\mathbf{P}_p = \mathbf{M}\mathbf{p}$$

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x_p = x$$

$$y_p = y$$

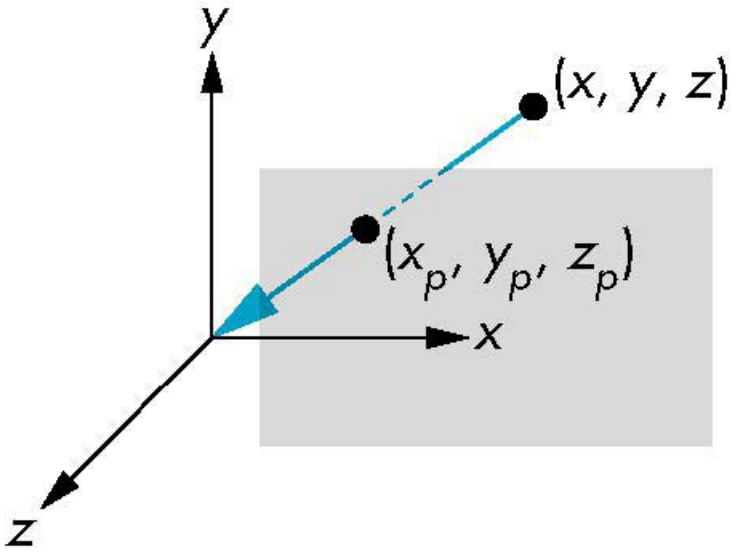
$$z_p = 0$$

$$w_p = 1$$

在实际应用中可以令M = I, 然后把对角线第三个元素置为零。

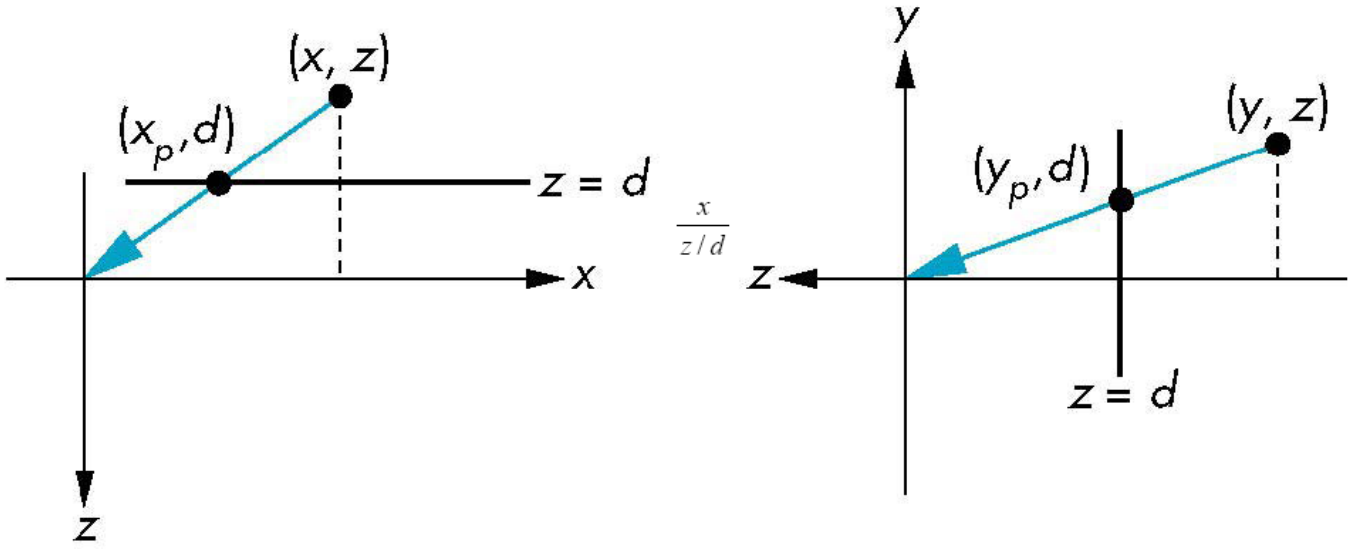
简单透视

- 只考虑两种透视投影模型的第一种
- 投影中心在原点
- 投影平面为$z = d, d < 0$



透视方程

- 考虑顶部与侧边视图



$$x_p = \frac{x}{z/d} \qquad y_p = \frac{y}{z/d} \qquad z_p = d$$

齐次坐标形式

$$\mathbf{p} = \mathbf{M}\mathbf{q}$$

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \qquad \mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \Rightarrow \quad \mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

透视除法

- 如果$w \neq 1$, 那么必须从齐次坐标中除以w而得到所表示的点
- 这就是透视除法，结果为

$$x_p = \frac{x}{z/d}, \quad y_p = \frac{y}{z/d}, \quad z_p = d$$

上述方程称为透视方程

- 后面会用OpenGL函数考虑相应的裁剪体

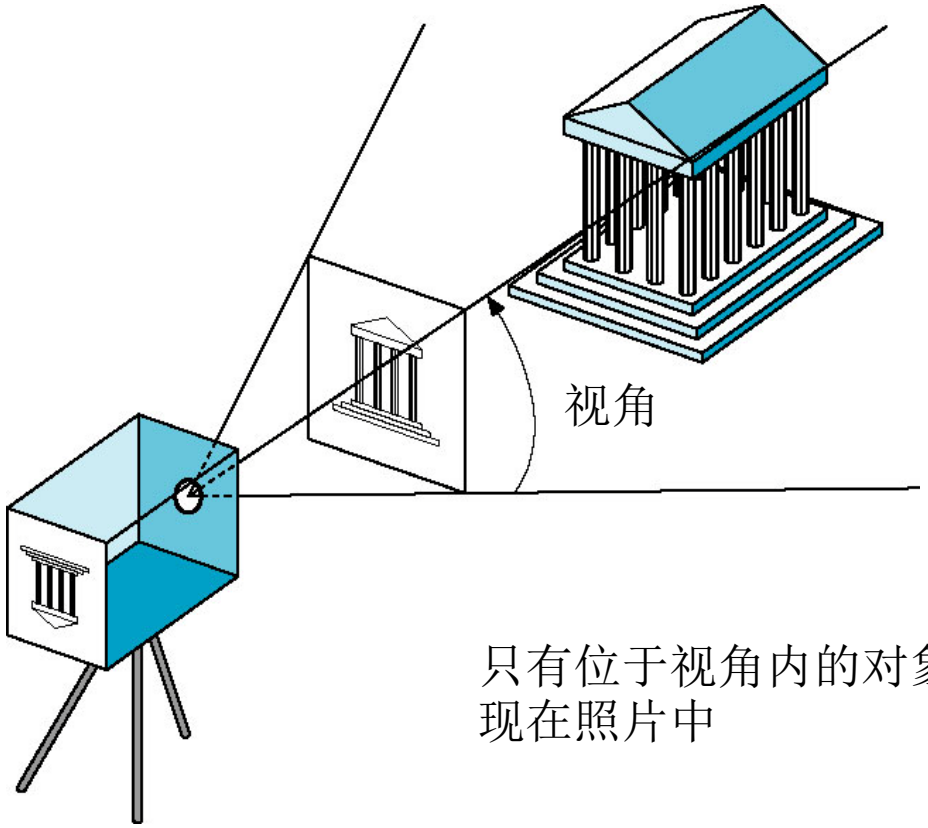
透视变换

- 透视除法是非线性的，导致非均匀缩短
  - 离COP远的对象投影后尺寸缩短得比离COP近的对象大
- 透视变换是保直线的，但不是仿射变换
- 透视变换是不可逆的，因为沿一条投影直线上的所有点投影后的结果相同

投影流水线

- 在模型－视图矩阵后应用4x4的投影矩阵实现简单的投影，但在最后需要进行透视除法
- 透视除法可以成为流水线的一部分

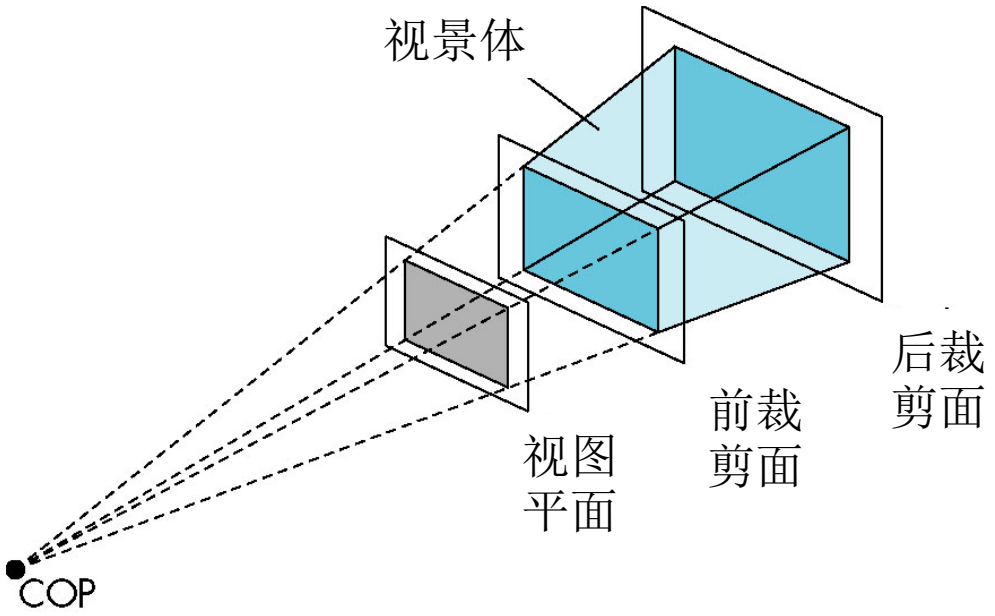
视角（angle of view）



只有位于视角内的对象才会出现在照片中

视景体

- 若胶卷是矩形的，那么由视角张成一个半无穷的棱台，这称为视景体
  - 其顶点就是COP
  - 但实际的视景体通常有前后裁剪面
  - 不在视景体内的物体被裁剪掉

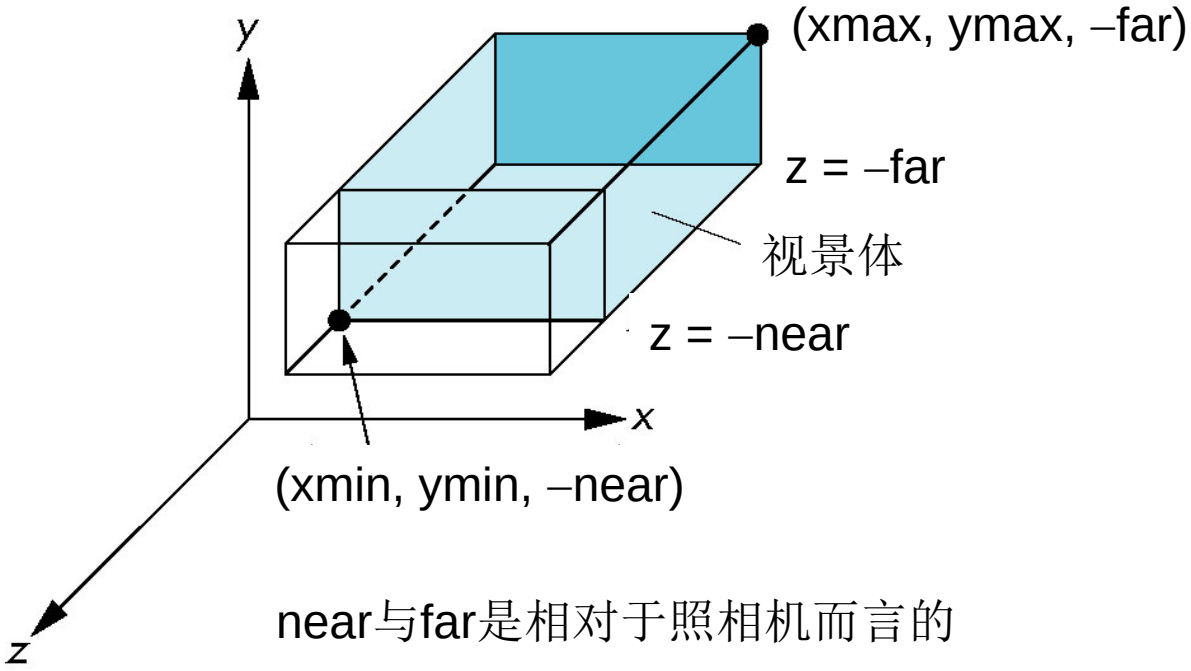


四棱体（frustum）

- 在绝大多数API中通过定义投影确定裁剪参数
- 通过指定前后裁剪面以及视角可以确定一个四棱体作为视景体
- 在这个指定中有一个参数是固定的，即COP在原点
- 可以定义四棱体的六个面差不多具有任意定向和位置，但这样很难确定视图，而且很少需要这种弹性

OpenGL的正交视图

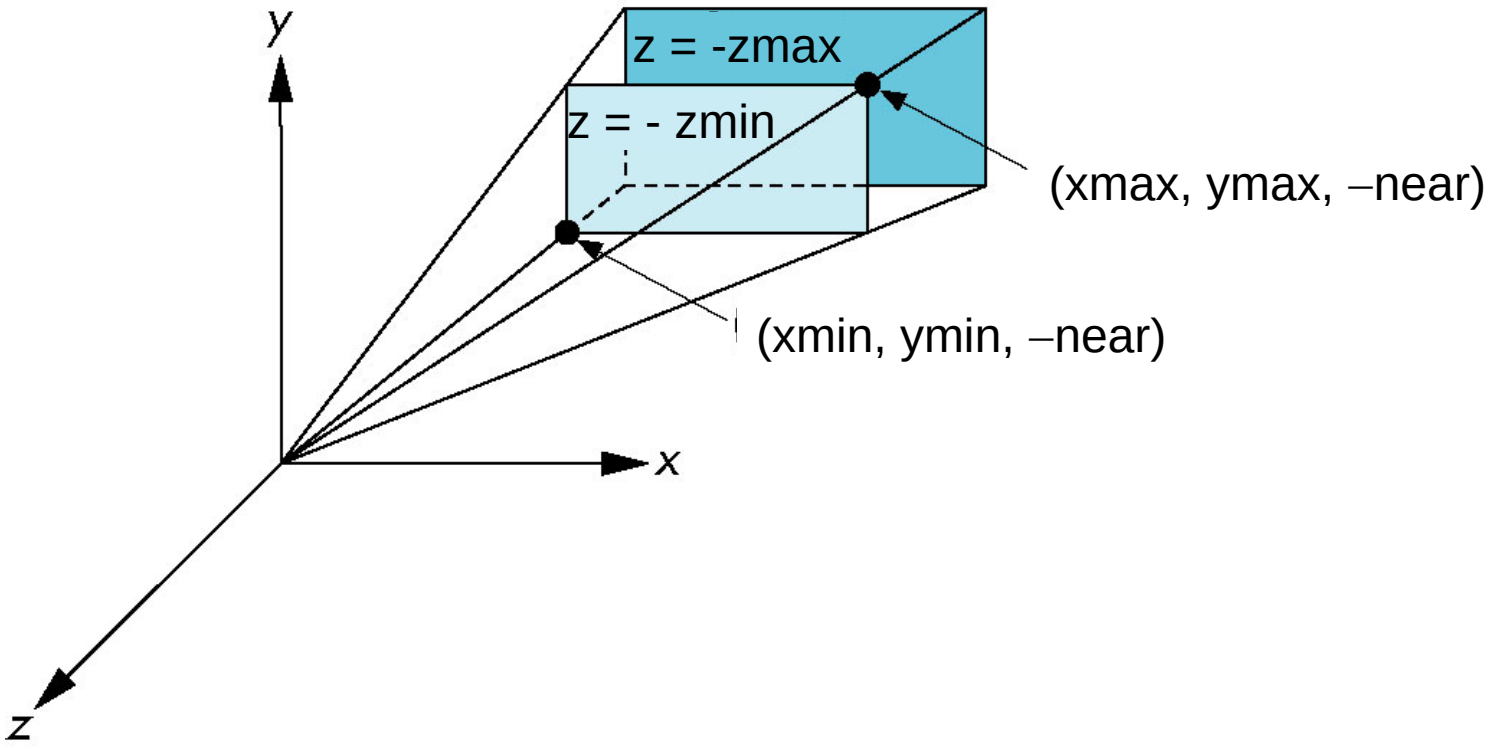
```
glOrtho(xmin, xmax, ymin, ymax, near, far)
glOrtho(left, right, bottom, top, near, far)
```



near与far是相对于照相机而言的

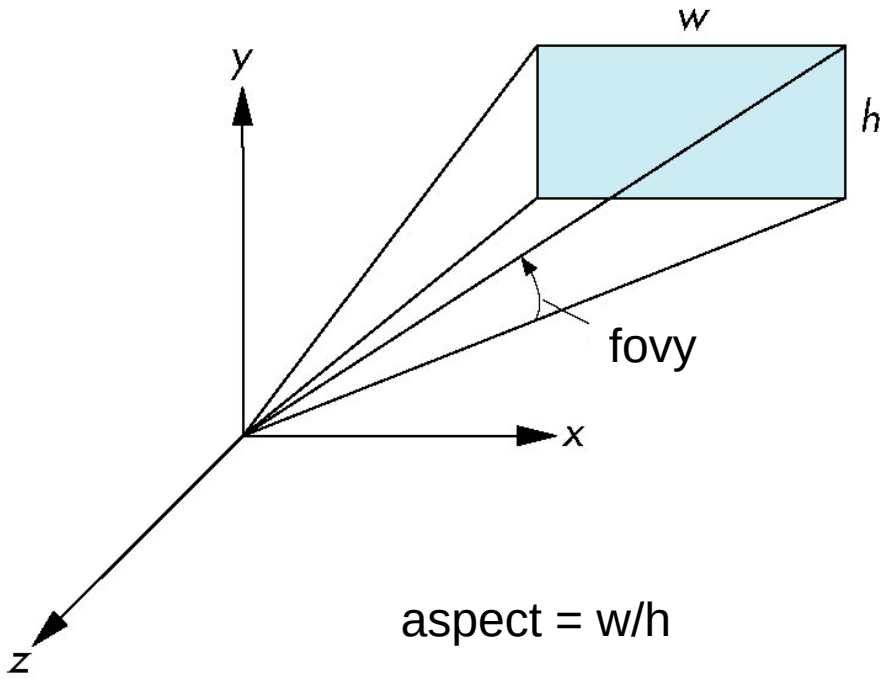
OpenGL的透视

```
glFrustum(xmin, xmax, ymin, ymax, near, far)
```



视野的应用

- 应用glFrustum有时很难得到所期望的结果
- gluPerspective(fovy, aspect, near, far)可以提供更好的界面



Thanks for your attention!