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

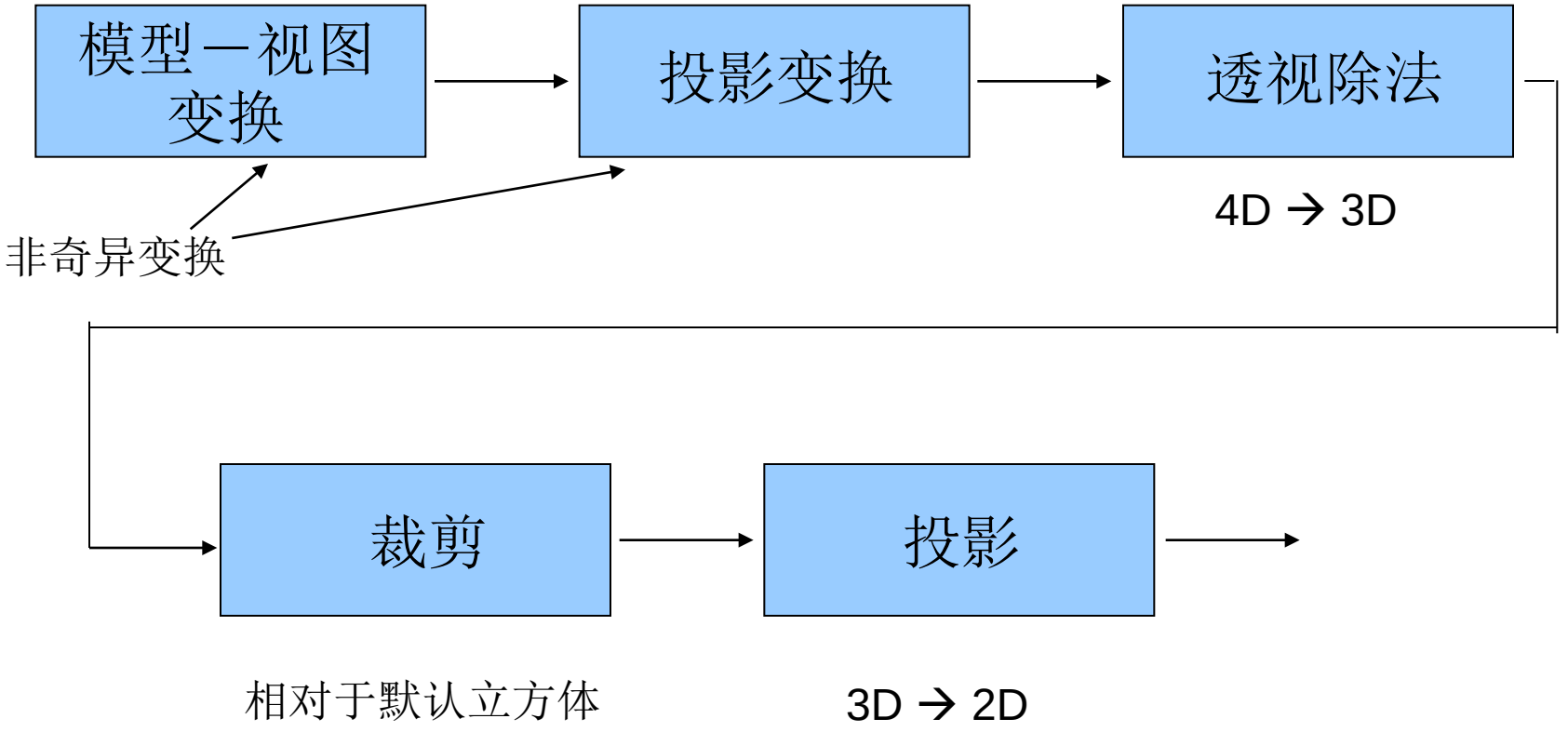
http://math.ustc.edu.cn/

第三节　投影矩阵

规范化

- 不想为每种类型的投影设计不同的投影矩阵，所以把所有的投影转化为具有默认视景体的正交投影
- 这种策略可以使我们在流水线中应用标准变换，并进行有效的裁剪

流水线



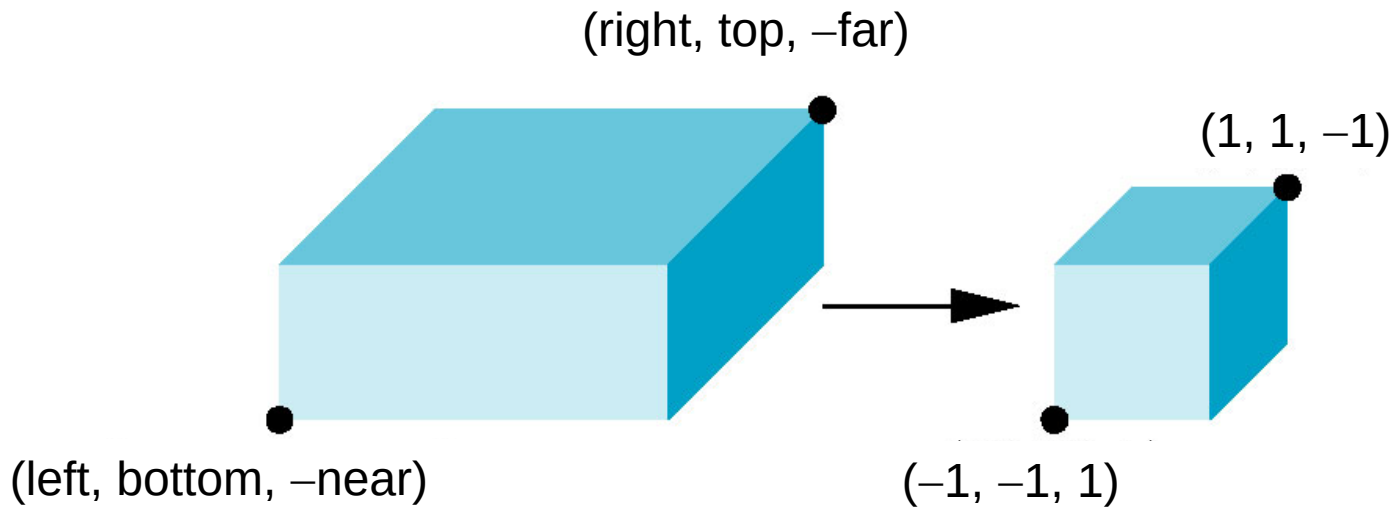
注释

- 在模型－视图变换和投影变换的过程中，我们是一直在四维齐次坐标系中的
  - 这些变换都是非奇异的
  - 默认值为单位阵（正交视图）
- 规范化使得不管投影的类型是什么，都是相对于默认的简单立方体进行裁剪
- 投影直到最后时刻才进行
  - 从而可以尽可能的保留深度信息，这对隐藏面消除是非常重要的

正交规范化

- 规范化 ⇒ 求出把指定裁剪体转化为默认裁剪体的变换

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



正交规范化矩阵

- 两步
  - 把中心移到原点，对应的变换为
    T(–(left+right)/2, –(bottom+top)/2, (near+far)/2))
  - 进行放缩从而使视景体的边长为2
    S(2/(left – right), 2/(top – bottom), 2/(near-far))

$$\mathbf{P} = \mathbf{ST} = \begin{bmatrix} \frac{2}{\text{right}-\text{left}} & 0 & 0 & -\frac{\text{right}+\text{left}}{\text{right}-\text{left}} \\ 0 & \frac{2}{\text{top}-\text{bottom}} & 0 & -\frac{\text{top}+\text{bottom}}{\text{top}-\text{bottom}} \\ 0 & 0 & \frac{2}{\text{near}-\text{far}} & \frac{\text{far}+\text{near}}{\text{far}-\text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

最后的投影

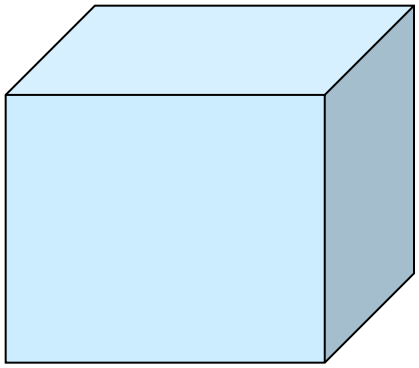
- 令z = 0
- 这等价于如下的齐次坐标变换

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

- 从而在4D中一般的正交投影为$P = M_{orth}ST$

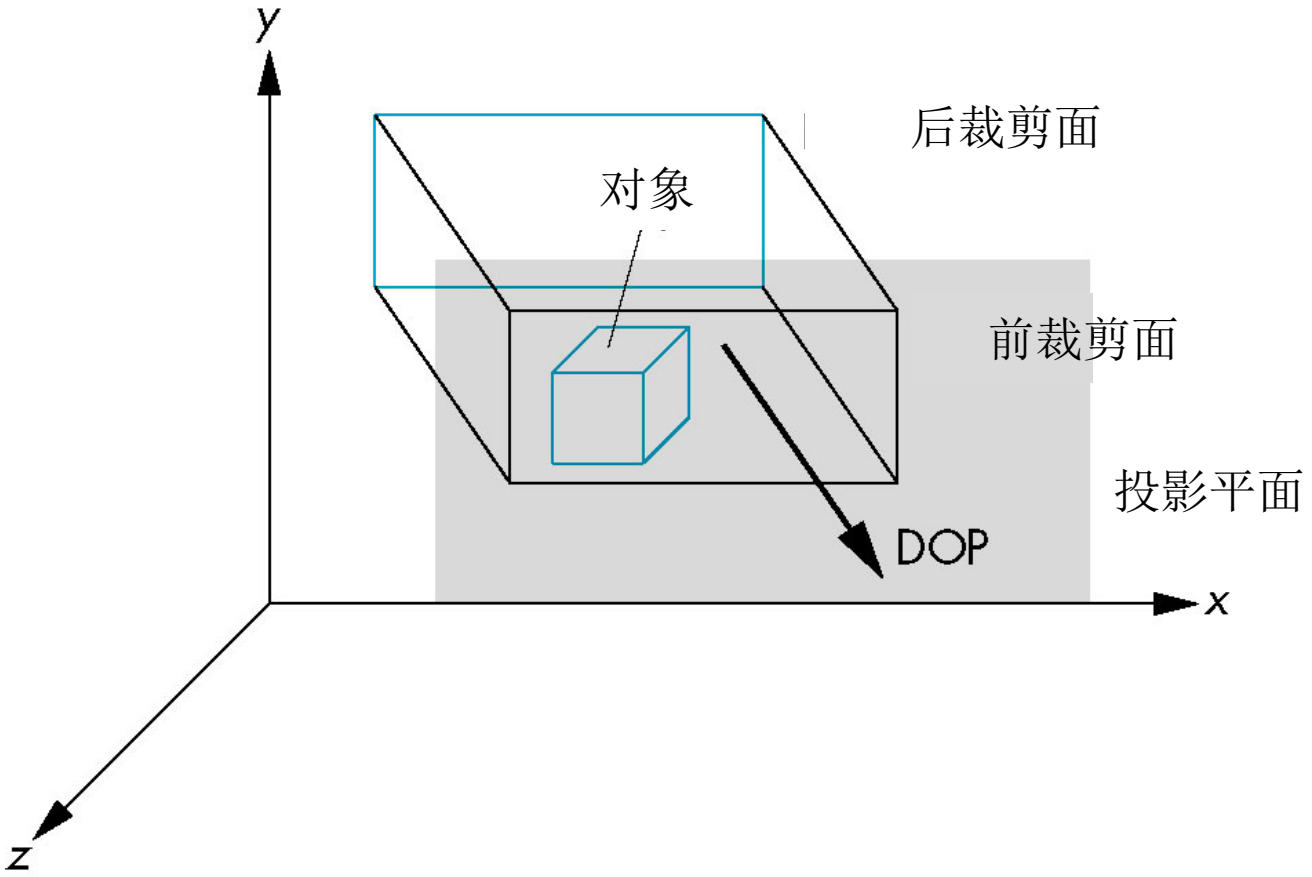
倾斜投影

- OpenGL的投影函数不支持一般的平行投影，例如立方体的如下图示

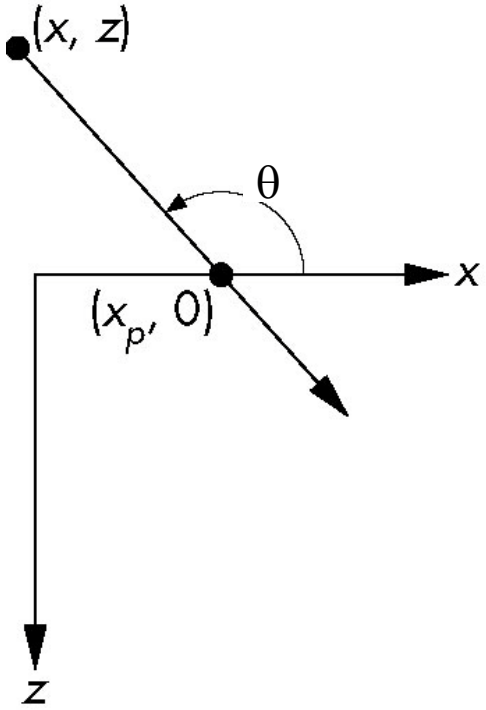


- 此时立方体好像发生了错切，然后再进行正交投影
- 倾斜投影 = 错切 + 正交投影

一般的错切

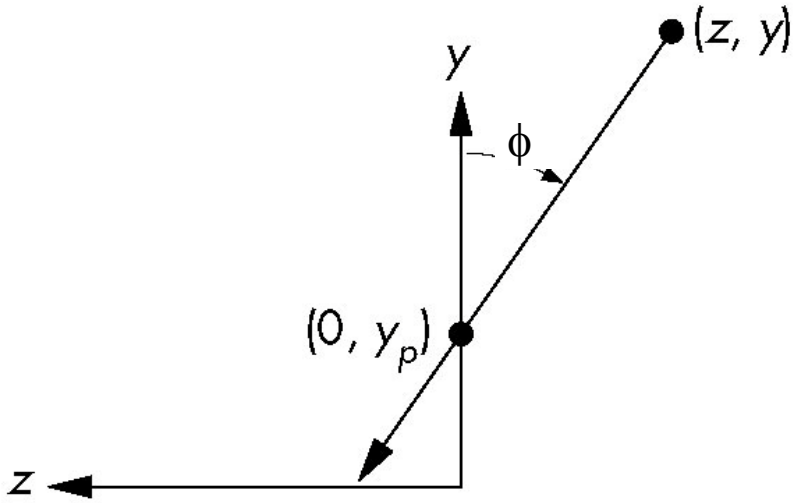


顶视图和侧视图



xp = x – z cot θ

(a)



yp = y – z cot ϕ

(b)

错切矩阵

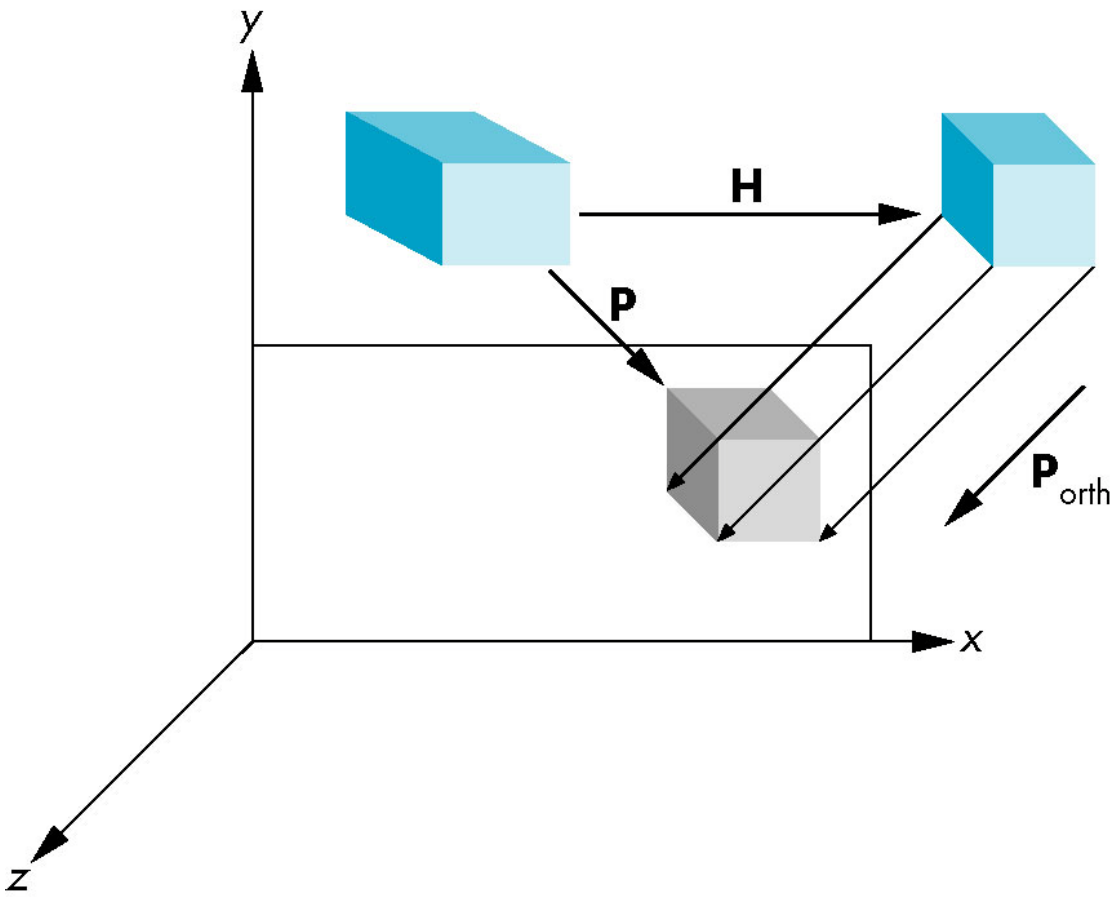
- *xy* 错切（z值不变）

$$\mathbf{H}(\theta,\phi)=\begin{bmatrix}1 & 0 & -\cot\theta & 0\\ 0 & 1 & -\cot\phi & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

投影矩阵 $P = M_{orth}\, H(\theta, \phi)$

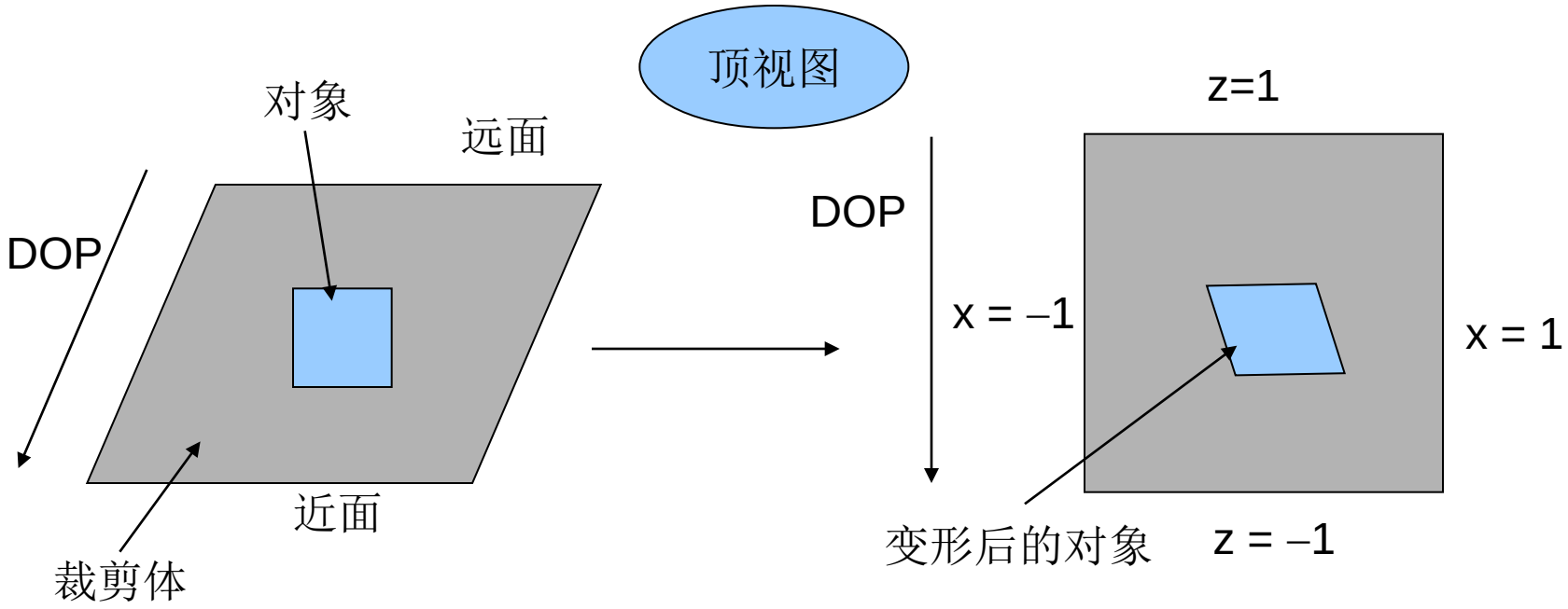
一般情形：$P = M_{orth}\, STH(\theta, \phi)$

等价性



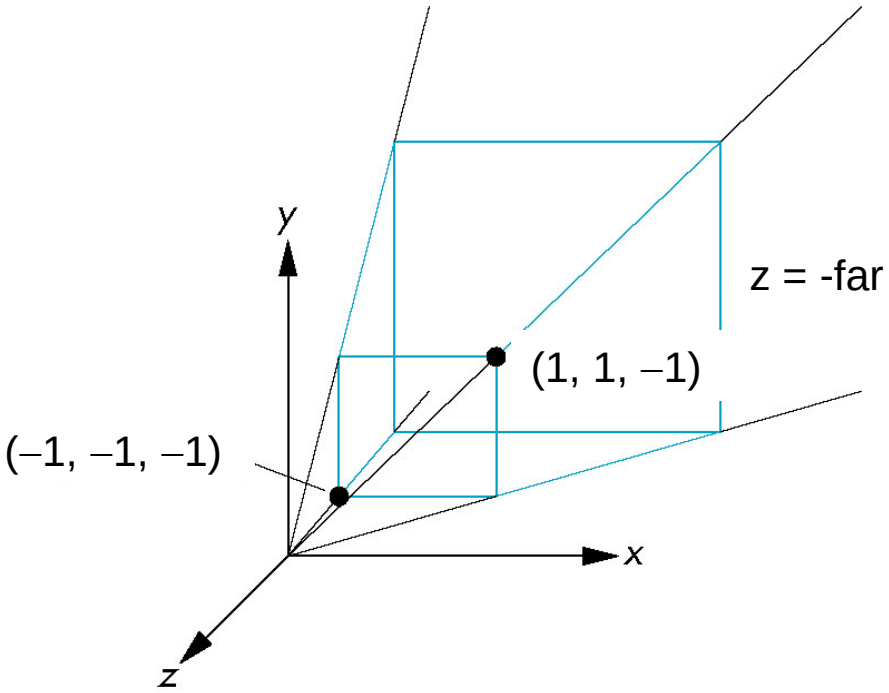
对裁剪体的影响

- 投影矩阵P = STH把原来的裁剪体变换为默认的裁剪体



简单透视

- 考虑简单透视：COP在原点，近裁剪面在z = -1, 由平面 x = ±z, y = ±z确定的有90度的视野



透视矩阵

- 齐次坐标下的简单投影矩阵为

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

注意这个矩阵与远裁剪面无关

推广

- 在透视除法后，点(x,y,z,1)变到了

  x’ = -x/z, y’ = -y/z, z’ = -(α+β/z)

  无论α,β的值是什么，在正交透影后就得到所期望的点。此时矩阵N非奇异

$$\mathbf{N}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & \alpha & \beta\\ 0 & 0 & -1 & 0\end{bmatrix}$$

α与β的选取

- 如果取

$\alpha = (near + far)/(far - near)$

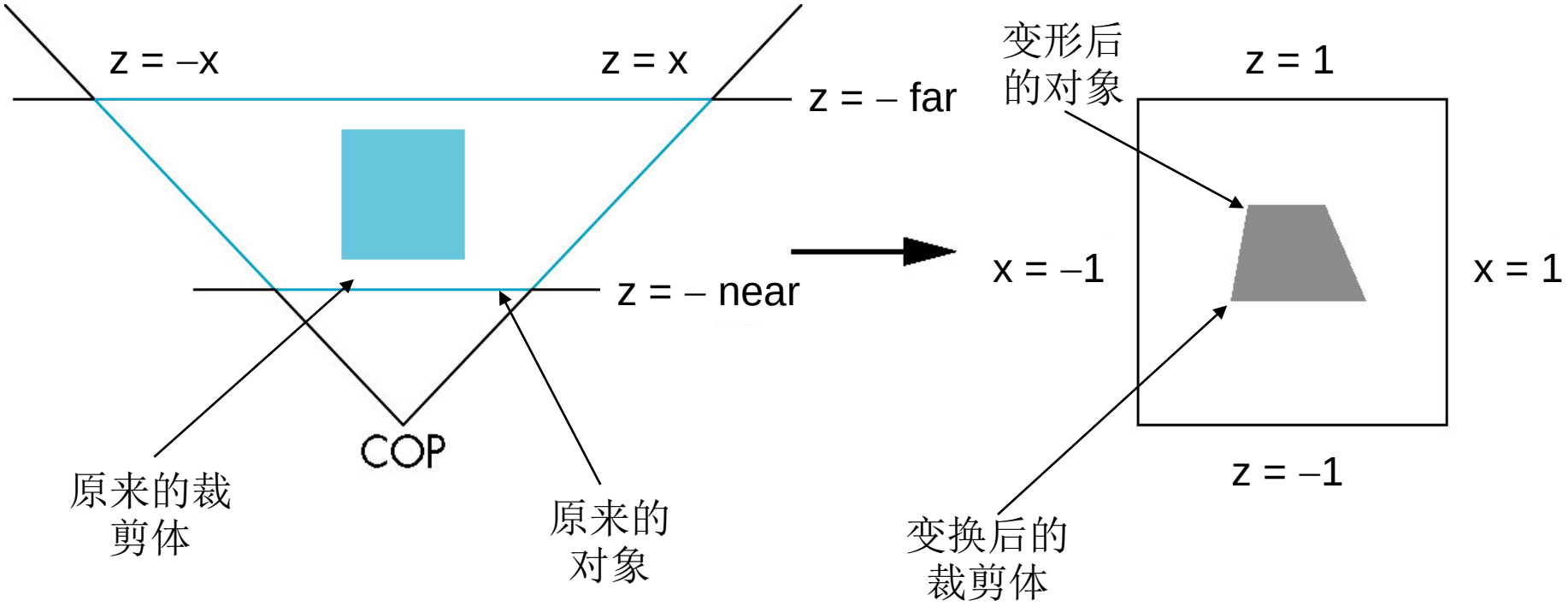
$\beta = 2\ near*far/(near - far)$

那么近平面映射到$z = -1$

远平面映射到$z = 1$

各侧边映射到 $x = \pm 1, y = \pm 1$

规范变换

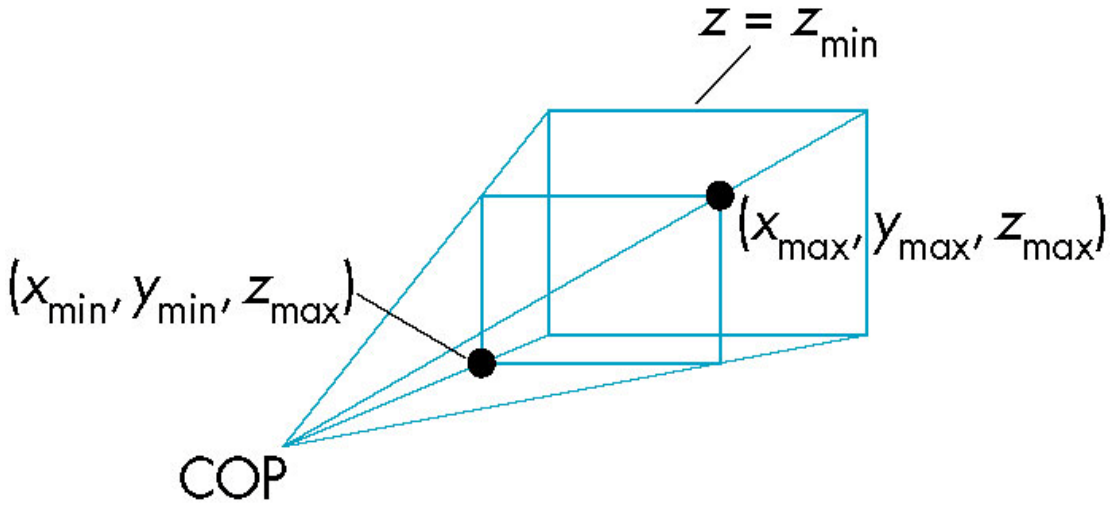


规范化与隐藏面消除

- 虽然这里选择的透视矩阵形式上看起来有点儿任意，但这种选择保证如果在原来的裁剪体内z1>z2，那么变换后的点满足z1'>z2'
- 因此如果首先应用规范变化，隐藏面消除算法有效
- 然而，公式z' = -(α+β/z)意味着由于规范化导致距离发生了改变，这可能导致数值问题，特别是当近距离非常小的时候更是如此

OpenGL的透视

- glFrustum可以定义非对称视景体，但gluPerspective不能做到这一点



OpenGL透视矩阵

- 在glFrustum中的规范化需要进行一个初始剪切变换，从而形成一个视景棱台，接着进行放缩变换，得到规范后的透视视景体。最后，透视矩阵导致只需要最后的正交变换：

$$P = NSH$$

投影矩阵

$$\mathbf{P} = \mathbf{NSH} = \begin{bmatrix} \frac{2z_{\min}}{x_{\max} - x_{\min}} & 0 & \frac{x_{\max} + x_{\min}}{x_{\max} - x_{\min}} & 0 \\ 0 & \frac{2z_{\min}}{y_{\max} - y_{\min}} & \frac{y_{\max} + y_{\min}}{y_{\max} - y_{\min}} & 0 \\ 0 & 0 & -\frac{\text{far} + \text{near}}{\text{far} - \text{near}} & -\frac{2\text{far} * \text{near}}{\text{far} - \text{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

为何采取这种方法？

- 规范化使得只需要一个流水线体系就可以进行透视投影和正交投影
- 尽可能位于四维齐次空间中，以便保持隐藏面消除和明暗处理所需要的三维信息
- 简化了裁剪的操作

Thanks for your attention!