计算机图形学小白入门

——————从0开始实现OpenGL

透视修正算法

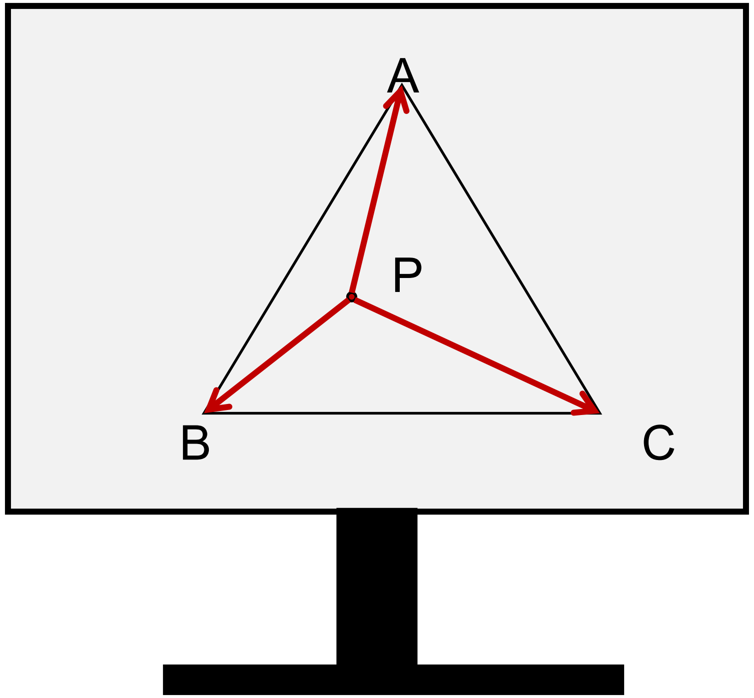
授课：赵新政
资深三维工程师

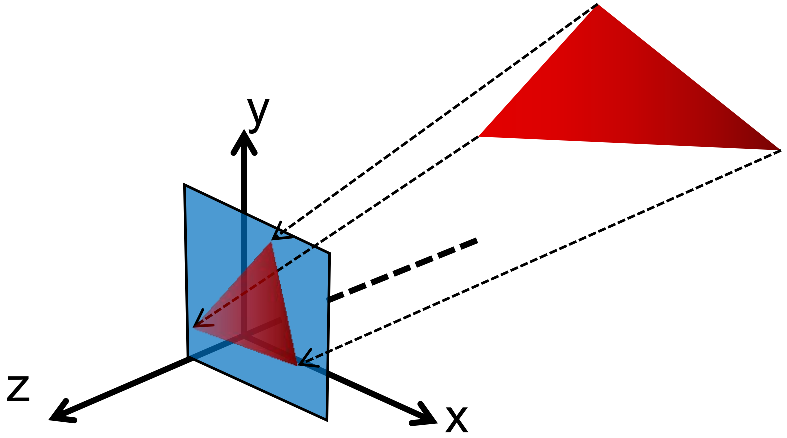
专注3D图形学技术
教育品牌

插值算法回顾

- **透视投影下**，目前代码中，都是变换到屏幕空间坐标系后，按照屏幕空间形成的三角形对某点进行重心坐标插值：

$$Color_P = \alpha.Color_A + \beta.Color_B + \gamma.Color_C$$

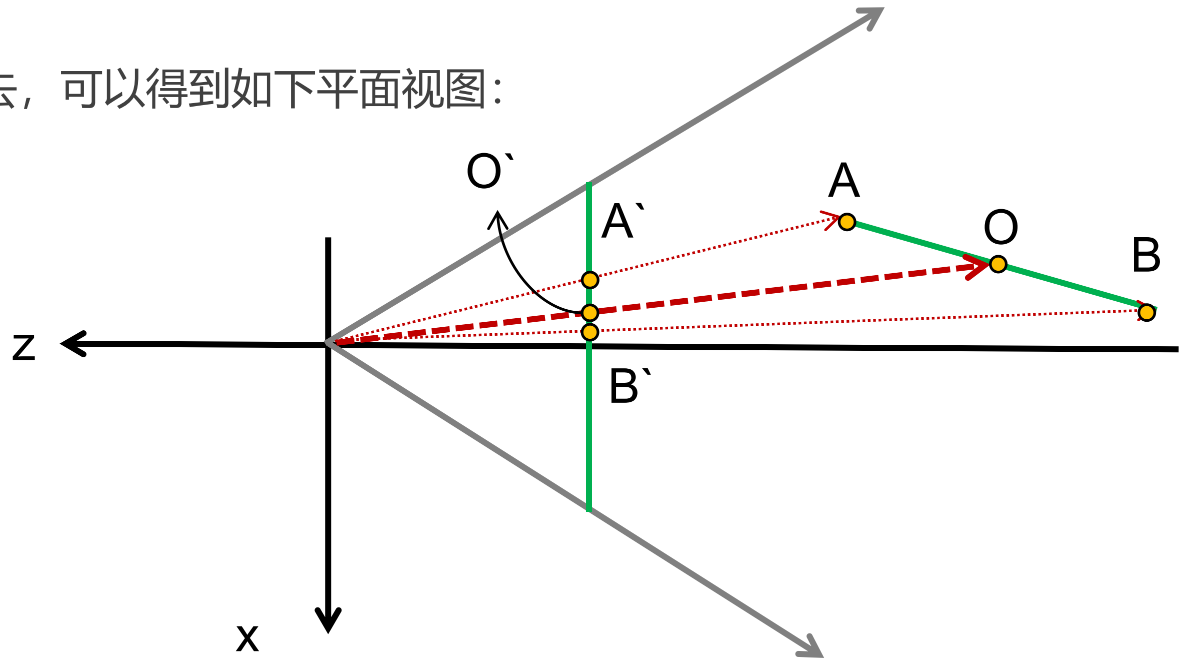




- 在三维空间中，三角形是立体/有深度的图形，需要使用**空间中的三角形重心坐标**，而不是**屏幕上二维的重心坐标**

问题分析

- 逆着y轴看下去，可以得到如下平面视图：

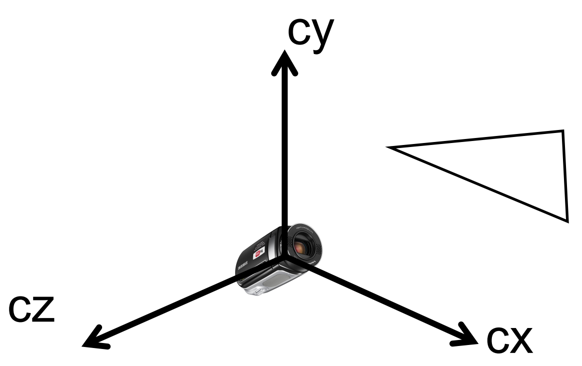


- 在屏幕空间，A`O`:B`O` = 7：3；
- 在三维空间，AO:BO=5:5
- **透视修正**：我们能方便计算**屏幕空间**三角形**像素的重心坐标**，需要找到一种方法计算**三维空间对应点的重心坐标**

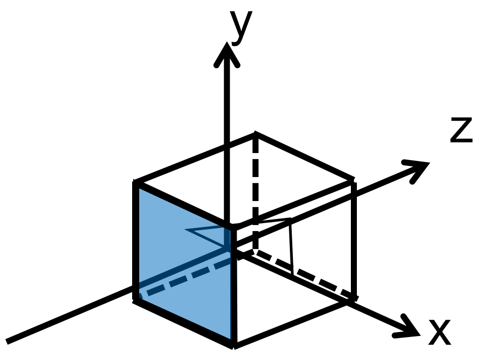
透视修正

问题分析

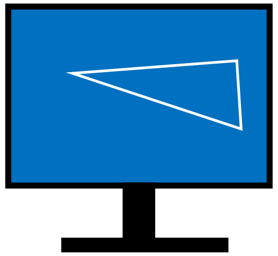
- 顶点到屏幕，会经历如下空间：



视图空间（摄像机坐标系）-V



NDC坐标空间-N

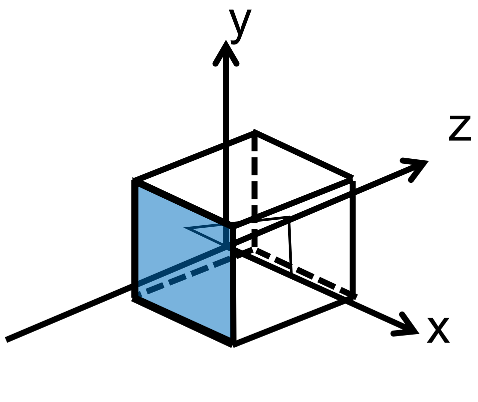


屏幕空间-S

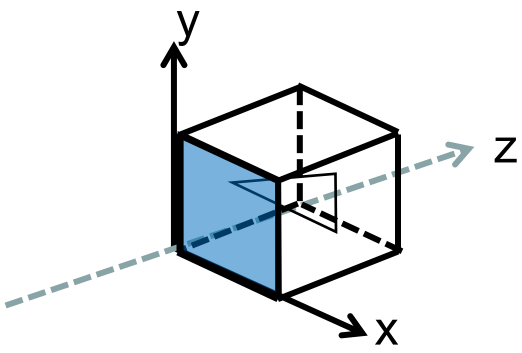
- 问题：
  - N空间到S空间，对应点的重心坐标是否发生变化?

问题分析

- N空间到S空间，会发生两个变化：
  - 顶点坐标从-1到1，变化为0-1，这个过程，这个过程NDC内对应顶点重心坐标不会发生改变



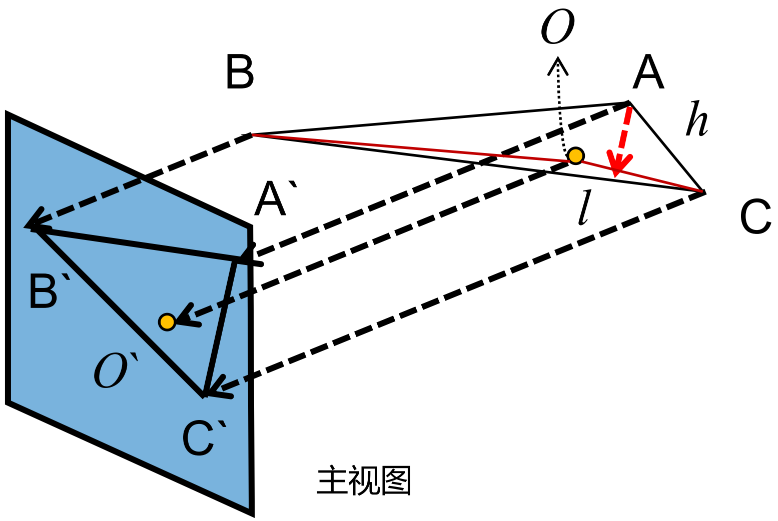
NDC坐标空间-N



0-1坐标空间

问题分析

- N空间到S空间，第二个变化：
  - 顶点坐标从为0-1变化为屏幕上具体像素点，并且由于width与height不同，产生了x/y拉伸
  - 可以理解为，**先投影到了1*1的幕布**，然后横纵**统一缩放width/height倍**



主视图

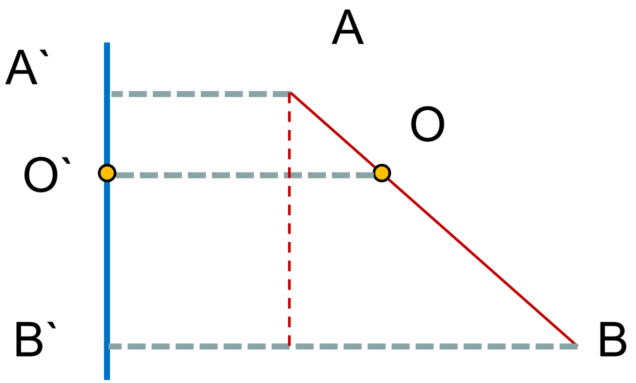
提问

O与O`点的重心坐标是否发生了变化?

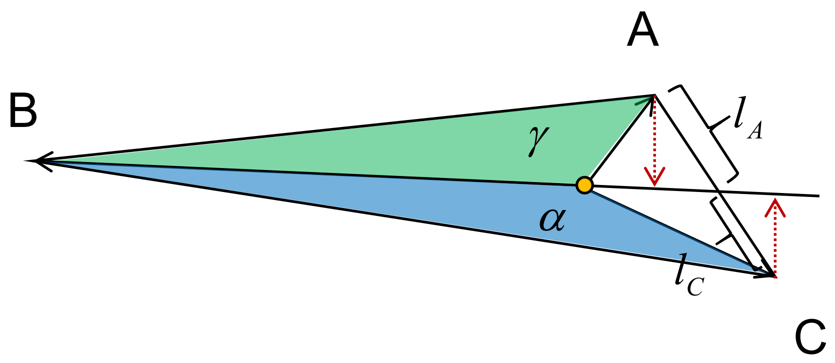
只要1*1幕布上重心坐标不变，则统一放大width/height倍后，重心坐标不变

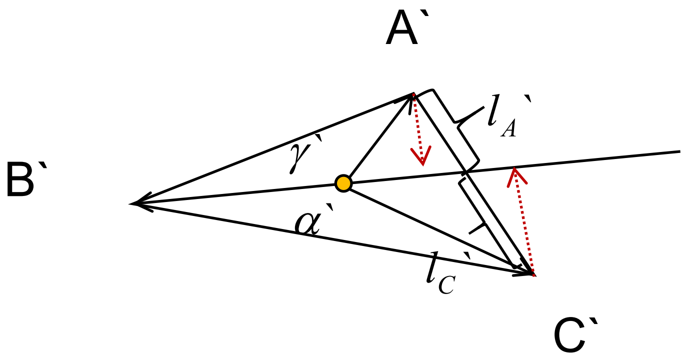
问题分析

- **观察直线投影**



- **AO:BO = A`O`:B`O`**
- 说明O点在投影与空间中插值比例不变





- 考察其中A的权重以及C的权重

$\alpha:\gamma = h_C : h_A = l_C : l_A$

- 考察其中A`的权重以及C`的权重

$\alpha`:\gamma` = h_C` : h_A` = l_C` : l_A` = l_C : l_A$

初步结论

- 存在O点在三角形内；
- 在**NDC**下计算O点**三维空间重心坐标**；
- 在**屏幕空间**计算O点**二维空间重心坐标**；
- 二者相等

只需要修正视图空间下O点的重心坐标与NDC空间下O点坐标的关系即可

问题明确

- 在**NDC空间**，已知三角形三个点为**A`/B`/C`**，并且已知其中点**O`**在本三角形重心坐标为: $(\alpha` \quad \beta` \quad \gamma`)$
- 即通过点O`在**屏幕空间**三角形内重心坐标计算而来，与之相等: $(\alpha` \quad \beta` \quad \gamma`)$

- 点**O`**在视图空间（摄像机坐标系）对应的三维点为**O**
- **A`/B`/C`**在视图空间（摄像机坐标系）对应的三维点为**A/B/C**
- 求三维空间真正的插值重心坐标: $(\alpha \quad \beta \quad \gamma)$

透视修正推导

- 考察从**A/B/C**到**A\`/B\`/C\`**的变换过程，需要一个投影矩阵P；
- P矩阵是4*4投影矩阵，可以分解为：3*3矩阵M，3维列向量t，以及最后一行数字（abcd）
- abcd这几个数字只会对w产生影响

$$A_c = \begin{pmatrix} M & & \vec{t} \\ a & b & c & d \end{pmatrix} . \begin{pmatrix} \vec{A} \\ 1 \end{pmatrix} \xrightarrow{\text{相乘得到结果，表示为：}} A_c = \begin{pmatrix} M.\vec{A}+\vec{t} \\ w_A \end{pmatrix} \xrightarrow{\text{透视除法}} A` = \left(\frac{M.\vec{A}+\vec{t}}{w_A} \right)$$

$$A` = \left( \frac{M.\vec{A}+\vec{t}}{w_A} \right) \qquad B` = \left(\frac{M.\vec{B}+\vec{t}}{w_B} \right) \qquad C` = \left(\frac{M.\vec{C}+\vec{t}}{w_C} \right)$$

透视修正推导

- 假设在NDC下有一个点O`，按照重心坐标定义，可以得到其坐标如下：

$$O`=\alpha`.\vec{A`}+\beta`.\vec{B`}+\gamma`.\vec{C`}$$

- 展开为：

$$O`=\alpha`.\left(\frac{M.\vec{A}+\vec{t}}{w_A}\right)+\beta`.\left(\frac{M.\vec{B}+\vec{t}}{w_B}\right)+\gamma`.\left(\frac{M.\vec{C}+\vec{t}}{w_C}\right)$$

- O在视图空间，可以表示为A/B/C点经过重心坐标插值的结果：

$$O=\alpha.\vec{A}+\beta.\vec{B}+\gamma.\vec{C}$$

- O`也是由O经过投影P变换以及透视除法变换而来，可以得到：

$$O`=\left(\frac{M.\vec{O}+\vec{t}}{w_o}\right)=\left(\frac{M.(\alpha.\vec{A}+\beta.\vec{B}+\gamma.\vec{C})+\vec{t}}{w_o}\right)$$

透视修正推导

- 上述结果联系在一起可得：

$$O` = \alpha` . \left(\frac{M.\vec{A} + \vec{t}}{w_A} \right) + \beta` . \left( \frac{M.\vec{B} + \vec{t}}{w_B} \right) + \gamma` . \left(\frac{M.\vec{C} + \vec{t}}{w_C} \right) = \frac{M.(\alpha.\vec{A} + \beta.\vec{B} + \gamma.\vec{C}) + \vec{t}}{w_o}$$

- 由投影矩阵得到的如上公式，适用于**任何的A/B/C取值**，所以左右系数必须非零且相等：

$$\frac{\alpha`}{w_A} = \frac{\alpha}{w_o} \qquad \alpha = \frac{\alpha`}{w_A}.w_o$$

$$\frac{\beta`}{w_B} = \frac{\beta}{w_o} \quad \Rightarrow \quad \beta = \frac{\beta`}{w_B}.w_o$$

$$\frac{\gamma`}{w_C} = \frac{\gamma}{w_o} \qquad \gamma = \frac{\gamma`}{w_C}.w_o$$

此时，我们已经求出来了两组系数之间的关系，距离求得真正的 $(\alpha \quad \beta \quad \gamma)$ 还有一个未知数 w_o

透视修正推导

- 由于已知重心坐标相加为1，可得：

$$\alpha+\beta+\gamma=1$$

$$\frac{\alpha`}{w_A}.w_o+\frac{\beta`}{w_B}.w_o+\frac{\gamma`}{w_C}.w_o=1$$

$$\frac{1}{w_o}=\frac{\alpha`}{w_A}+\frac{\beta`}{w_B}+\frac{\gamma`}{w_C}$$

- 从而我们可以得到如下结论：

$$\frac{1}{w_o}=\frac{\alpha`}{w_A}+\frac{\beta`}{w_B}+\frac{\gamma`}{w_C}$$

$$\alpha=\frac{\alpha`}{w_A}.w_o$$

$$\beta=\frac{\beta`}{w_B}.w_o$$

$$\gamma=\frac{\gamma`}{w_C}.w_o$$

$$Color_O=(\frac{\alpha`}{w_A}.Color_A+\frac{\beta`}{w_B}.Color_B+\frac{\gamma`}{w_C}.Color_C).w_O$$

透视修正应用（图例）

$$Color_O = (\frac{\alpha`}{w_A}.Color_A + \frac{\beta`}{w_B}.Color_B + \frac{\gamma`}{w_C}.Color_C).w_O$$

A-clip

B-clip

C-clip

perspectiveDivision

- p.oneOverW=1/p.w
- p.color *= oneOverW
- p.uv*= oneOverW

$$Color_A = \frac{Color_A}{w_A}$$

$$Color_B = \frac{Color_B}{w_B}$$

$$Color_C = \frac{Color_C}{w_C}$$

raster正常插值每个像素

$$p.oneOverW = \alpha`.A.oneOverW + \beta`.B.oneOverW + \gamma`.C.oneOverW$$

$$p.color = \alpha`.A.color + \beta`.B.color + \gamma`.C.color$$

$$p.uv = \alpha`.A.uv + \beta`.B.uv + \gamma`.C.uv$$

$$Color_O = \frac{\alpha`}{w_A}.Color_A + \frac{\beta`}{w_B}.Color_B + \frac{\gamma`}{w_C}.Color_C$$

perspectiveRecover
针对每个像素

- p.color /= p.oneOverW
- p.uv /= p.oneOverW

$$Color_O = (Color_O).w_O$$

透视修正-讨论深度Depth插值

- 对于顶点的z值，经过投影矩阵以及透视除法后，到达NDC坐标下，经过屏幕空间变换，到达0-1的坐标范围内，称为**Depth**
- 问题：
  - 此时Depth作为一个被插值的属性，可否直接使用屏幕空间的重心坐标插值呢？
- 考察透视除法z变化的细节：

$$p_c = \begin{pmatrix} \frac{1}{aspect.\tan(fovyHalf)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(fovyHalf)} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$z = -\frac{f+n}{f-n}.z + \frac{-2fn}{f-n}$$

$$z_{ndc} = \frac{f+n}{f-n} + \frac{2fn}{f-n}.\frac{1}{z}$$

- z_{ndc} 当中的常数项可以忽略，查之后不会发生改变
- z_{ndc} 当中的第二项，**已经变成了z的倒数**，可以直接在屏幕空间使用重心坐标进行插值