2022 WebGL高级课程

WebGL Advanced Course

WebGL模型变换

讲解人：冰老师

讲解时间：20221106

冰鉴韵

西藏 阿里

扫一扫上面的二维码图案，加我微信

WebGL交流群

目录

Lorem Ipsum is simply dummy text of the printing and typesetting industry

01

前言

前言

想起自己上半年，捏的一个陶器，每个陶器都有一个自己的坐标，我可以任意把这个泥团捏成想要的样子，变大、变小、旋转等等。这就是我们这节课要介绍的模型矩阵。另外大家猜猜哪个是我的杰作。（得意）

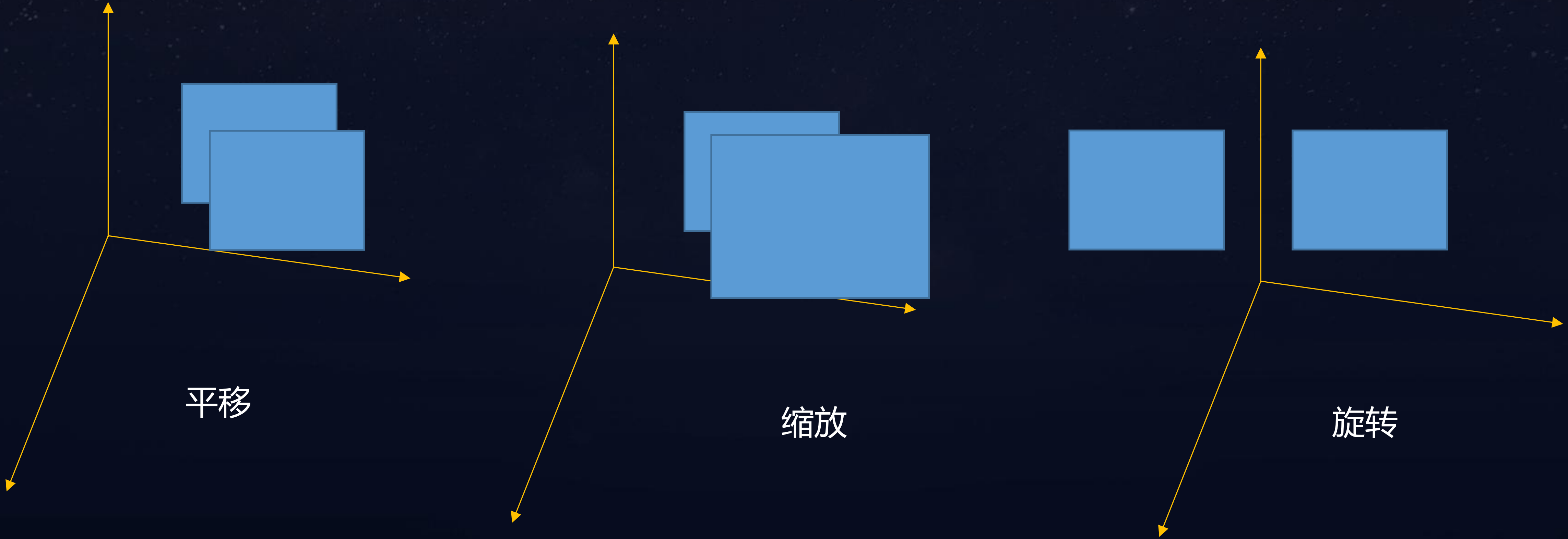
Lorem Ipsum is simply dummy text of the printing and typesetting industry

02

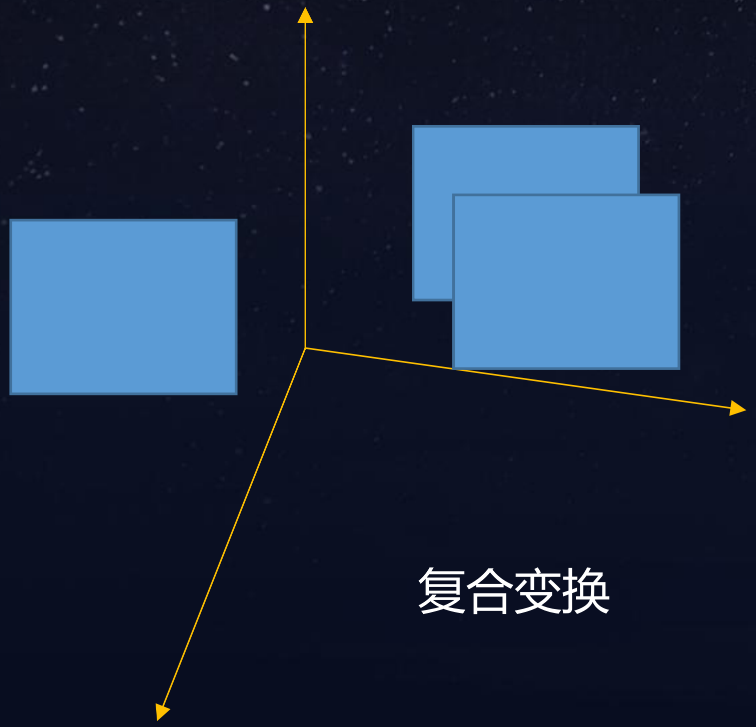
模型变换

定义

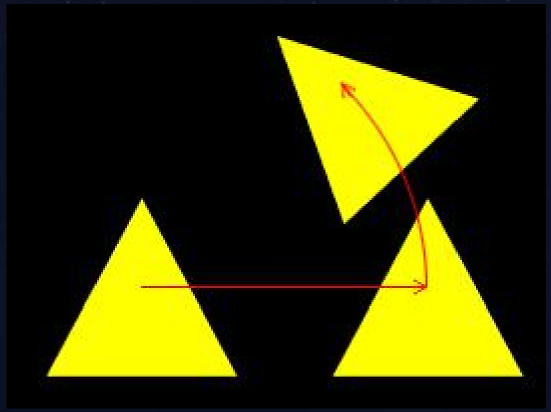
模型变换：是从模型坐标系到世界坐标系的转换。
简单通俗来讲，3D建模之初，模型有自己的坐标系，以及相应的点坐标。就是将场景中的模型摆好，这个过程就叫模型变换。



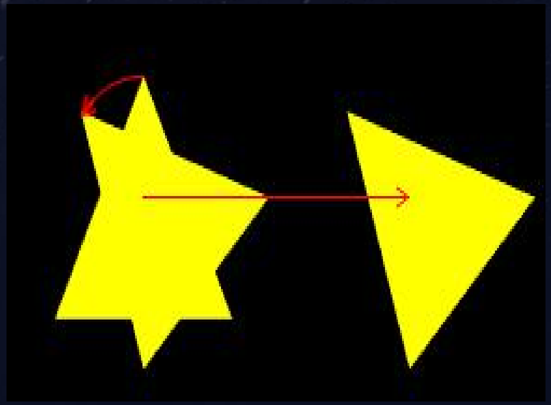
定义



复合变换



先平移后旋转



先旋转后平移

模型变换不同顺序有不同的结果，其实就是因为矩阵相乘不满足交换律，但满足结合律，所以对应同一个复合变换，可以先得出其中的基础变换的矩阵乘积，再与输入向量相乘。

Lorem Ipsum is simply dummy text of the printing and typesetting industry

03

公式推导

齐次坐标

什么是齐次坐标?

很多人应该都知道欧式几何，我们学的初中、高中、大学、研究生几乎都是欧式几何为主。
欧式几何里面最重要的一个定理就是:
两条平行线永远不会相交。
但实际上在日常应用中我们也有很多非欧几何的场景。比如透视空间（我们回头会在投影坐标系讲到），最开始齐次坐标是用来解决这个问题的。

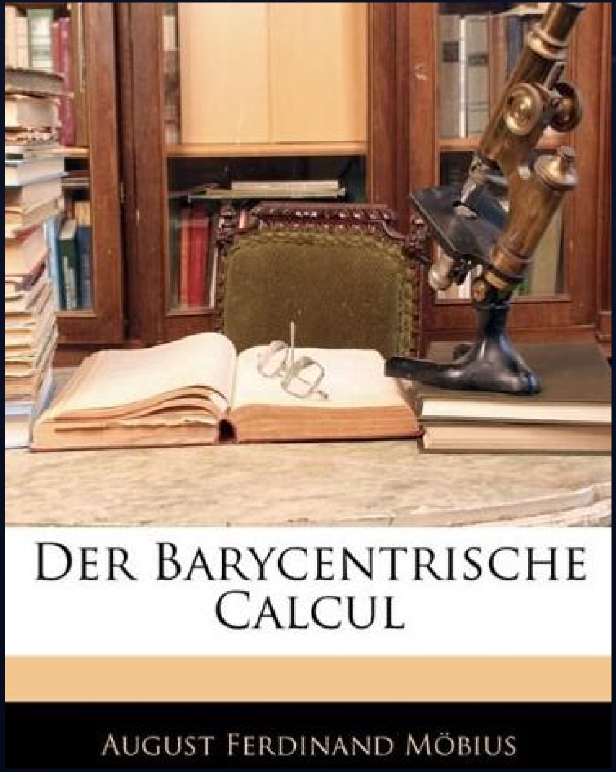
齐次坐标

什么是齐次坐标?

齐次坐标是奥古斯特 · 费迪南德 · 莫比乌斯于1827年在其著作《Der barycentrische Calcul》首次提出的，使其在投影空间中进行图形和几何计算成为可能。

举个例子：
笛卡尔坐标系下一个点(X,Y)在齐次坐标里面变成了（x,y,w），变形为（x/w,y/w,1）
例如，笛卡尔坐标系下（1，2）的齐次坐标可以表示为（1，2，1），如果点（1，2）移动到无限远处，在笛卡尔坐标下它变为(∞,∞)，然后它的齐次坐标表示为（1，2，0），因为(1/0, 2/0) = (∞,∞)，我们可以不用”∞“来表示一个无穷远处的点了，可以用更为量化的比例关系来表示无穷远出的几何关系。
顾名思义：所谓的“齐次”可以理解成同比例关系。

总结来说：齐次坐标用N+1维度表示N



齐次坐标

为什么要在三维图形学面广泛用到了齐次坐标

使用齐次坐标有什么优势?

比如把(1, 4, 7)如果写成（1,4,7,0），它就是个向量；如果是(1,4,7,1)，它就是个点。下面是如何在普通坐标(Ordinary Coordinate)和齐次坐标(Homogeneous Coordinate)之间进行转换：这里可以看出，区别就是▲与Y。点的重点在点，向量的重点在方向。

(1) 从普通坐标转换成齐次坐标时

如果(x,y,z)是个点，则变为(x,y,z,1);

如果(x,y,z)是个向量，则变为(x,y,z,0)

○◑㊀从齐次坐标转换成普通坐标时

如果是(x,y,z,1)，则知道它是个点，变成(x,y,z);

如果是(x,y,z,0)，则知道它是个向量，仍然变成(x,y,z)

Y、能够表示无穷远　**◑、能够表示点在直线或平面**　**Y、能够表示两条直线的交点**　**▭、能够区分一个向量和一个点**

如何判断点在直线l上？

ax+by+c =0 用向量表示为l=(a,b,c)T

点p=(x, y)在直线 l 上的充分必要条件是 ax + by + c = 0，如果使用齐次坐标的话，点p的齐次坐标就是p‘=(x, y, 1)

等价于l p‘ = 0

因此，点p在直线l上的充分必要条件就是直线l 与p的齐次坐标p‘的内积为零。

如何判断点在平面A上？

方程 ax + by + cz + d = 0用向量表示为是s= (a,b,c,d)T

三维空间的一个平面A可以用方程 ax + by + cz + d = 0 来表示，

三维空间的一个点P=(x, y, z) 的齐次坐标 P'=(x, y, z, 1)

类似的，点P在空间平面A上可用两个向量的内积表示：

s P’=0

首先，根据前面叉乘的定义，l x m 的结果向量（记为 p = l x m） 与 l 和 m都垂直，根据点乘的定义，垂直的向量之间的点积为0，因此可以得到：

$$l^T*(l\times m)=l^T*p=0$$
$$m^T*(l\times m)=m^T*p=0$$

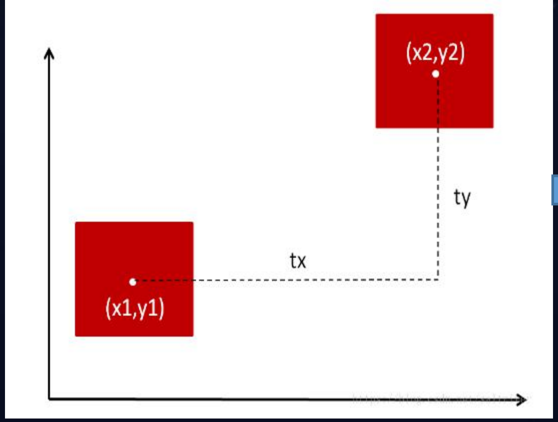
因此，根据前面点在直线上的结论，可以看到p既在直线l 上又在直线m上，所以 p = l x m 是两条直线的交点。此处 p 是齐次坐标。

齐次坐标

为什么要在三维图形学面广泛用到了齐次坐标

使用齐次坐标有什么优势?

一、高效的表示图形变换



这对于图像中的每一个点都是成立的。写成矩阵的形式如下:

$$x2 = x1 + tx$$
$$y2 = x2 + ty$$

非齐次

$$\begin{bmatrix} x2 \\ y2 \end{bmatrix} = \begin{bmatrix} x1 \\ y1 \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

齐次

$$\begin{bmatrix} x2 \\ y2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ 1 \end{bmatrix}$$

齐次坐标表示时可以将加法转换为乘法。

平移

$$x' = x + Tx$$
$$y' = y + Ty$$
$$z' = z + Tz$$

方法一借助人力资源

变换前：

```
var data=new Float32Array([
   1.0, 0.0, 0.0,//三角形顶点1坐标
   0.0, 1.0, 0.0,//三角形顶点2坐标
   0.0, 0.0, 1.0//三角形顶点3坐标]);
```

变换后：

```
var data=new Float32Array([
   3.0, 0.0, 0.0,//三角形顶点1坐标
   2.0, 1.0, 0.0,//三角形顶点2坐标
   2.0, 0.0, 1.0//三角形顶点3坐标]);
```

方法二 借助CPU

```
for(var i = 0;i<9;i += 3 ){
data[i] += 2.0;}
```

方法三：借助GPU

```
// 在顶点着色器中逐顶点沿着x轴平移+2.0
gl_Position =vec4(apos.x+2.0,apos.y,apos.z,1);
```

方法四：平移矩阵法（手动推导）

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

平移

No.

Date

$$x' = x \times 1 + y \times 0 + z \times 0 + T_x$$
$$y' = x \times 0 + y \times 1 + z \times 0 + T_y$$
$$z' = x \times 0 + y \times 0 + z \times 1 + T_z$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

旋转

$$x' = x\cos\beta - y\sin\beta$$
$$y' = x\sin\beta + y\cos\beta$$
$$z' = z$$

绕z轴旋转90度

方法一借助人力资源

变换前：

```
var data=new Float32Array([
   1.0, 0.0, 0.0,//三角形顶点1坐标
   0.0, 1.0, 0.0,//三角形顶点2坐标
   0.0, 0.0, 1.0//三角形顶点3坐标]);
```

变换后：

```
var data=new Float32Array([
    0.0, 1.0, 0.0,//三角形顶点1坐标
   -1.0, 0.0, 0.0,//三角形顶点2坐标
    0.0, 0.0, 1.0//三角形顶点3坐标]);
```

方法二 借助CPU

```
let arr = [ 0.0, 0.0, 1.0,    0.0, 1.0, 0.0,    1.0, 0.0, 0.0]
let newArr = [];
let b = Math.PI/2;
for (let i = 0; i < arr.length; i += 3) {
   newArr[i] = arr[i]*Math.cos(b)-arr[i+1]*Math.sin(b);
   newArr[i+1] = arr[i]*Math.sin(b)+arr[i+1]*Math.cos(b);
   newArr[i+2] =arr[i+2];}
```

方法三：借助GPU

```
// 在顶点着色器中逐顶点绕z轴旋转90度
gl_Position =vec4(apos.x*cosb-apos.y*sinb,apos.x*sinb+apos.y*cosb,apos.z,1);
```

方法四：旋转矩阵法（手动推导）

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & -\sin\beta & 0 & 0 \\ \sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

旋转

$$x' = x\cos\beta - y\sin\beta + z\times 0 + D$$

$$y' = x\sin\beta + y\cos\beta + z\times 0 + F$$

$$z' = z + 0\times x + 0\times y + W$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & -\sin\beta & 0 & 0 \\ \sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

缩放

$$
\begin{aligned}
x' &= x * 5 \\
y' &= y * 5 \\
z' &= z * 5
\end{aligned}
$$

Xyz分别扩大5倍

方法一借助人力资源

变换前：

```
var data=new Float32Array([
   1.0, 0.0, 0.0,//三角形顶点1坐标
   0.0, 1.0, 0.0,//三角形顶点2坐标
   0.0, 0.0, 1.0//三角形顶点3坐标]);
```

变换后：

```
var data=new Float32Array([
    5.0, 0.0, 0.0,//三角形顶点1坐标
    0.0, 5.0, 0.0,//三角形顶点2坐标
    0.0, 0.0, 5.0//三角形顶点3坐标]);
```

方法二 借助CPU

```
let arr = [0.0, 0.0, 1.0,   0.0, 1.0, 0.0,   1.0, 0.0, 0.0 ]
      let newArr = [];
      let b = 5;
      for (let i = 0; i < arr.length; i += 3) {
         newArr[i] = arr[i]*b;
         newArr[i+1] = arr[i+1]*b;
         newArr[i+2] =arr[i+2]*b;}
```

方法三：借助GPU

```
// 在顶点着色器中xyz分别扩大5倍
gl_Position =vec4(apos.x*2*,apos.y*2,apos.z*2,1);
```

方法四：缩放矩阵法（手动推导）

$$
\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

旋转

$x' = x \times 5$

$y' = y \times 5$

$z' = z \times 5$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

谢谢观看

冰鉴韵

西藏 阿里

扫一扫上面的二维码图案，加我微信

WebGL交流群