Logic_视觉班课堂笔记 [CC001]

- **日期**: 2019年5月10日星期五
- **授课**: CC老师
- **主题**: OpenGL 主题

课程内容

- **OpenGL 初探**
  - 图形API简介
  - 图形API目的解决什么问题
  - OpenGL 专业名词解析
  - OpenGL 下坐标系解析
  - 图片/图形从文件渲染到屏幕过程解析
  - 案例01: 固定管线下使用OpenGL 渲染三角形
  - 案例02: 固定管线下使用OpenGL 渲染正方形,并能通过键盘移动该图形
  - 案例03: 固定管线下使用OpenGL 绘制图形

课后作业

请在个人博客上更新一篇博文,选题如下

1. 快速了解图形API
2. OpenGL下坐标系统解析
3. 理解图片从文件渲染屏幕的过程
4. 快速了解OpenGL 下专业名词

> 要求:
> 1. 将课程内容加上自己的理解
> 2. 建议更新的博客地址发送到讨论群.互相学习

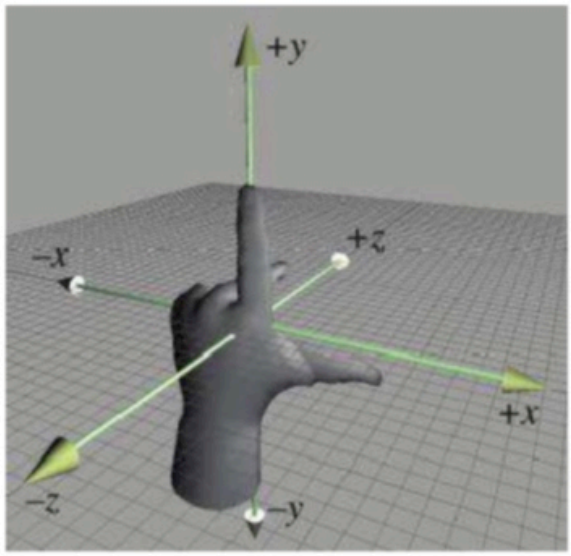
iOS开发者需要学习OpenGL/OpenGL ES

- Metal -> OpenGL ES

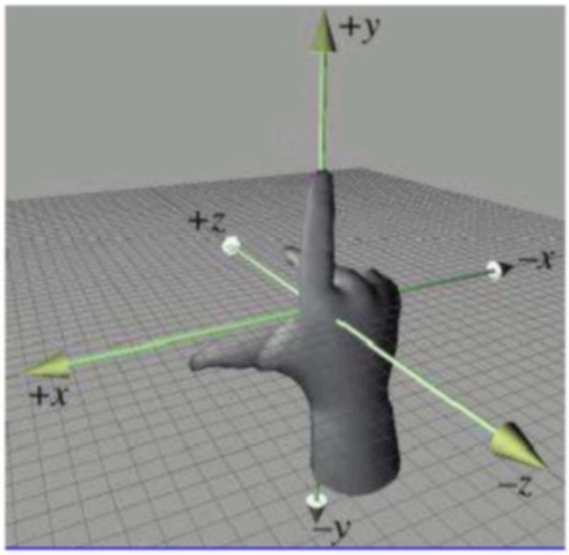
```
glClearColor(1.0,1.0,1.0,1.0);
```

```
glEable(GL_DEPTH_TEST);
glEable(GL_BLEND)
```

顶点数据是由 CPU / GPU 来处理?
GPU -> 内存 顶点数组(内存中)
顶点缓存区: 区域(不在内存! -> 显卡显存中.)
片元着色器
片段着色器
像素着色器
片元函数
GPUImage



左手坐标系



右手坐标系

[-1,1] 标准化设备坐标系(NDC)

//补充完整文字笔记

- 局部空间(物体空间坐标系)
- 世界空间
- 观察空间
- 裁剪空间
- 屏幕空间

MVP 矩阵:
Model,view,projection

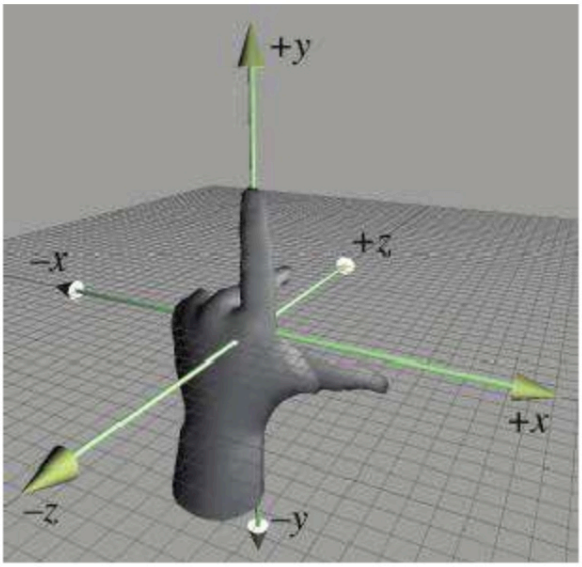
- 世界坐标系

- 物体坐标系
- 摄像机坐标系
- 惯性坐标系

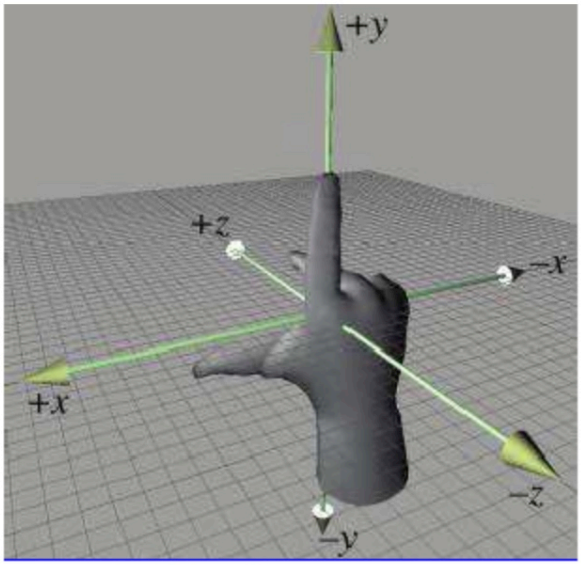
物体/世界/照相机空间-> 右手系
规范化设备坐标: 左手系.

x,y,z => 0,1,2

左手坐标系和右手坐标系



左手坐标系



右手坐标系

注意 `OpenGL` 中坐标系 `OpenGL` 中的物体、世界、照相机坐标系都属于右手坐标系，而规范化设备坐标系使用左手坐标系。笼统地说 `OpenGL` 使用右手坐标系是不合适的

坐标系

OpenGL 希望每次顶点着色后，我们的可见顶点都为标准化设备坐标（**Normalized Device Coordinate,NDC**）。**也就是说每个顶点的z,y,z都应该在–1到1之间，超出这个范围的顶点将是不可见的。**

通常情况下我们会自己设定一个坐标范围，之后再在顶点着色器中将这些坐标变换为表转化设备坐标。然后这些标化设备坐标传入光栅器（Rasterizer）,将它们变换为屏幕上的二维坐标和像素。

将坐标变换为标准化设备坐标，接着再转化为屏幕坐标的过程通常是分步进行的，也就是类似于流水线那样子。在流水线中，物体的顶点在最终转化为屏幕坐

标之前还会被变换到多个坐标系统(Coordinate System)。将物体的坐标变换到几个过渡坐标系(Intermediate Coordinate System)的优点在于，在这些特定的坐标系统中，一些操作或运算更加方便和容易，这一点很快就会变得很明显。对我们来说比较重要的总共有5个不同的坐标系统

- **局部空间(Local Space，或者称为物体空间(Object Space))**
- **世界空间(World Space)**
- **观察空间(View Space，或者称为视觉空间(Eye Space))**
- **裁剪空间(Clip Space)**
- **屏幕空间(Screen Space)**

就是一个顶点在最终被转化为片段之前需要经历的所有不同状态.为了将坐标从一个坐标系变换到另一个坐标系，我们需要用到几个变换矩阵，最重要的几个分别是**模型(Model)、观察(View)、投影(Projection)三个矩阵。物体顶点的起始坐标再局部空间（Local Space）,这里称它为局部坐标（Local Coordinate），它在之后会变成世界坐标（world Coordinate）,观测坐标（View Coordinate）,裁剪坐标（Clip Coordinate）,并最后以屏幕坐标（Screen Corrdinate）的形式结束**

在3D图形学中常用的坐标系:

- **世界坐标系**
- **物体坐标系**
- **摄像机坐标系**
- **惯性坐标系**

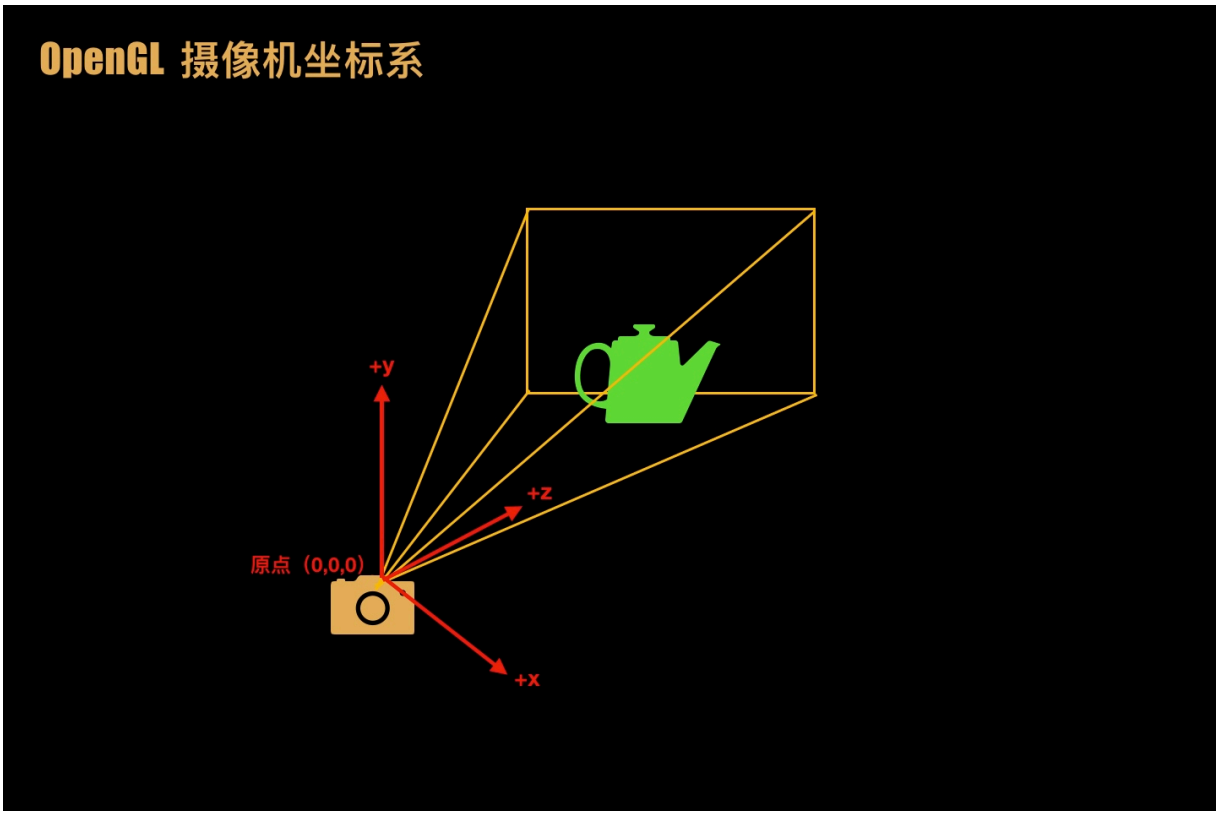
世界坐标系: 世界坐标系是系统的绝对坐标系,在没有建立用户坐标系之前画面上所有的点的坐标都可以在该坐标系的原点来确定各自的位置.世界坐标系始终是固定不变的

物体坐标系: 每个物体都有他们独立的坐标系.当物理移动或者改变方向时.该物体相关联的坐标系将随之移动或改变方向。

物体坐标系是以物体本身而言，比如，我先向你发指令，“向前走一步”,是向您的物体坐标体系指令。我并不知道你会往哪个绝对的方向移动。比如说，当你开车时，有人会说向左转，有人说向东。但是，向左转是物体坐标系的概念，而向东则是世界坐标系中的。

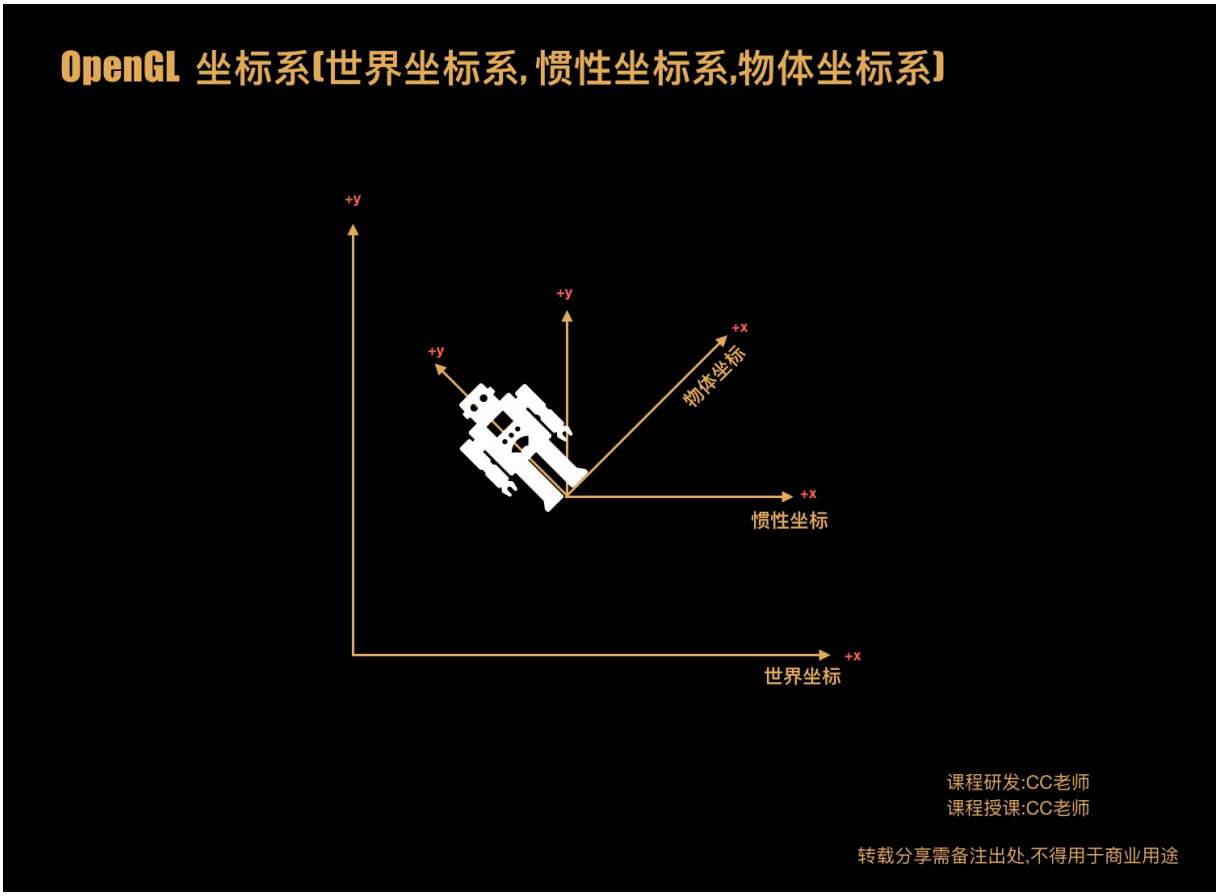
在某种情况下，我们可以理解物体坐标系为模型坐标系。因为模型顶点的坐标都是在模型坐标系中描述的。

摄像机（照相机）坐标系:在坐标系的范畴里，摄像机坐标系和照相机坐标系都是一样的意义。照相机坐标系是和观察者密切相关的坐标系。照相机坐标系和屏幕坐标系相似，差别在于照相机坐标系处于3D空间中，而屏幕坐标系在2D平面里。



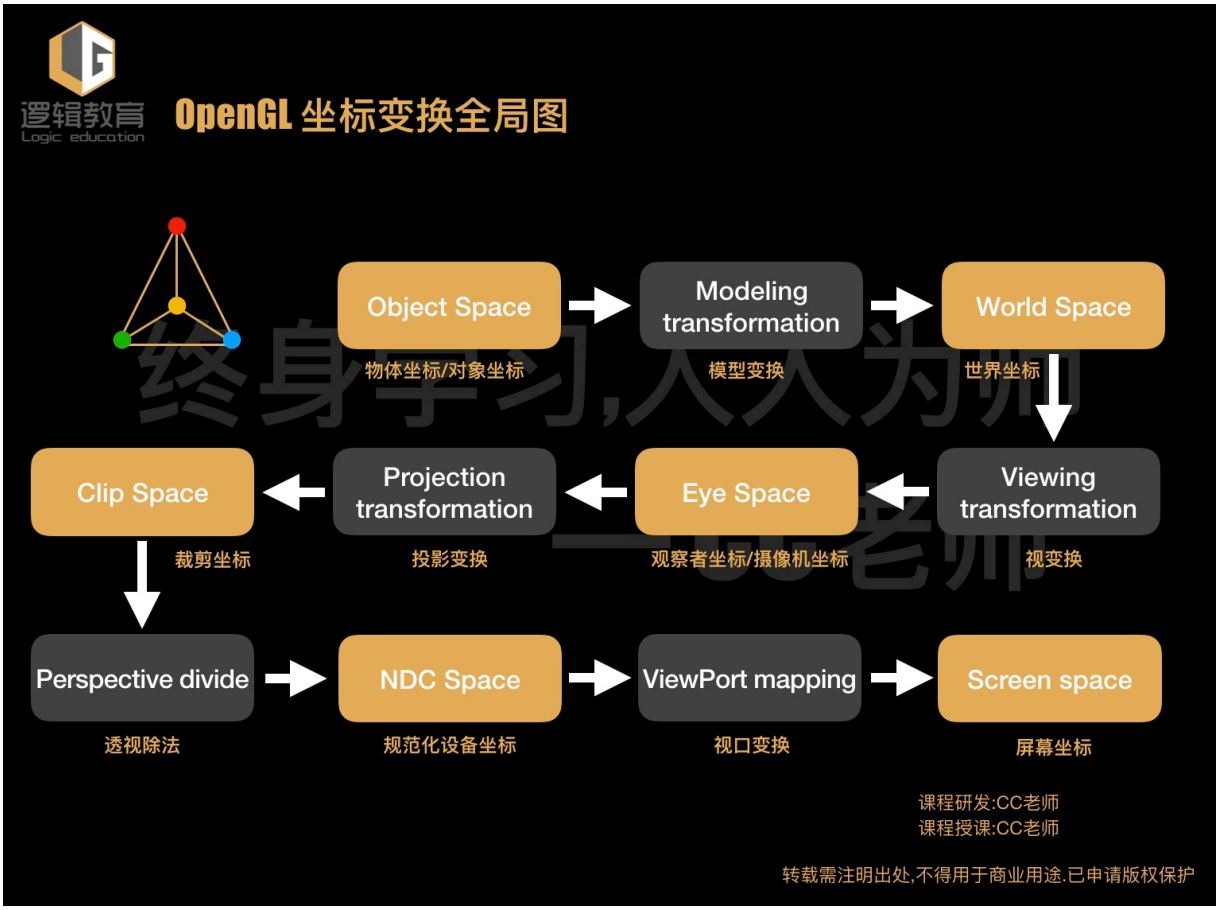
惯性坐标系: 指的是世界坐标系到物体坐标系的"半途". 惯性坐标系的原点和物体坐标原点重合,但惯性坐标系的轴平行于世界坐标系的轴.

为什么要引入惯性坐标系?因为物体坐标系转换到惯性坐标系只需要旋转,从惯性坐标系转换到世界坐标系只需要平移.

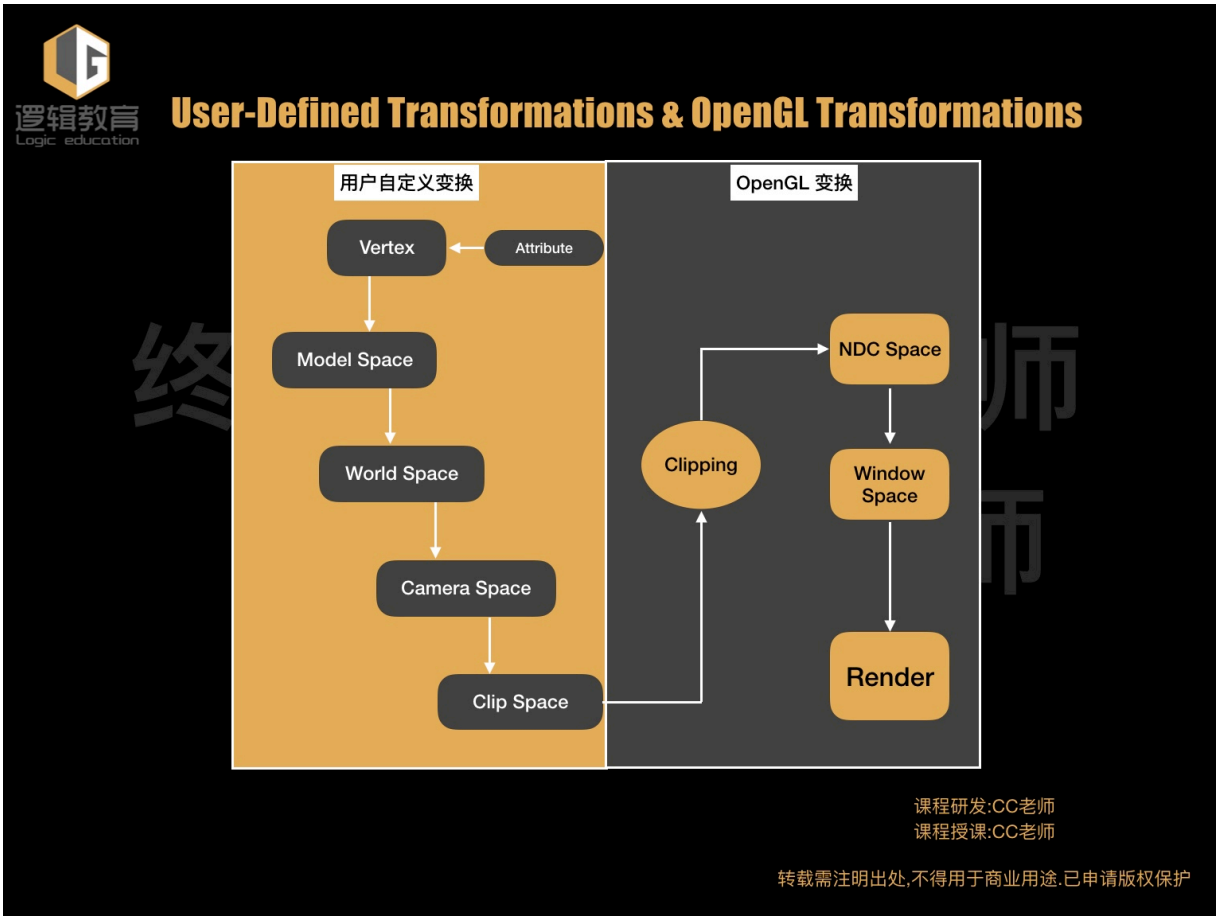


坐标变换的全局图

OpenGL最终的渲染设备是2D的，我们需要将3D表示的场景转换为最终的2D形式，前面使用模型变换和视变换将物体坐标转换到照相机坐标系后，需要进行投影变换，将坐标从相机—》裁剪坐标系，经过透视除法后，变换到规范化设备坐标系(NDC)，最后进行视口变换后，3D坐标才变换到屏幕上的2D坐标，这个过程如下图所示



在上面的图中，**注意，OpenGL只定义了裁剪坐标系、规范化设备坐标系和屏幕坐标系，而局部坐标系(模型坐标系)、世界坐标系和照相机坐标系都是为了方便用户设计而自定义的坐标系，它们的关系如下图所示**



- 图中左边的过程包括**模型变换、视变换，投影变换，这些变换可以由用户根据需要自行指定，这些内容在顶点着色器中完成**；
- 图中右边的两个步骤，**包括透视除法、视口变换，这两个步骤是OpenGL自动执行的，在顶点着色器处理后的阶段完成。**

将坐标系统组合在一起

我们为上述的每一个步骤都创建了一个变换矩阵：**模型矩阵**、**观察矩阵**和**投影矩阵**。一个顶点坐标将会根据以下过程被变换到裁剪坐标：

$$V_{clip} = M_{pro} \cdot M_{view} \cdot M_{model} \cdot V_{local}$$

这一系列的矩阵变换需要从右往左读。最后的顶点应该被赋值到顶点着色器中的gl_Position，OpenGL将会自动进行透视除法和裁剪。

OpenGL 然后对裁剪坐标执行透视除法从而将它们变换到标准化设备坐标。**OpenGL** 会使用 `glViewPort` 内部的参数来将标准化设备坐标映射到屏幕坐标，每个坐标都关联了一个屏幕上的点。这个过程称为视口变换

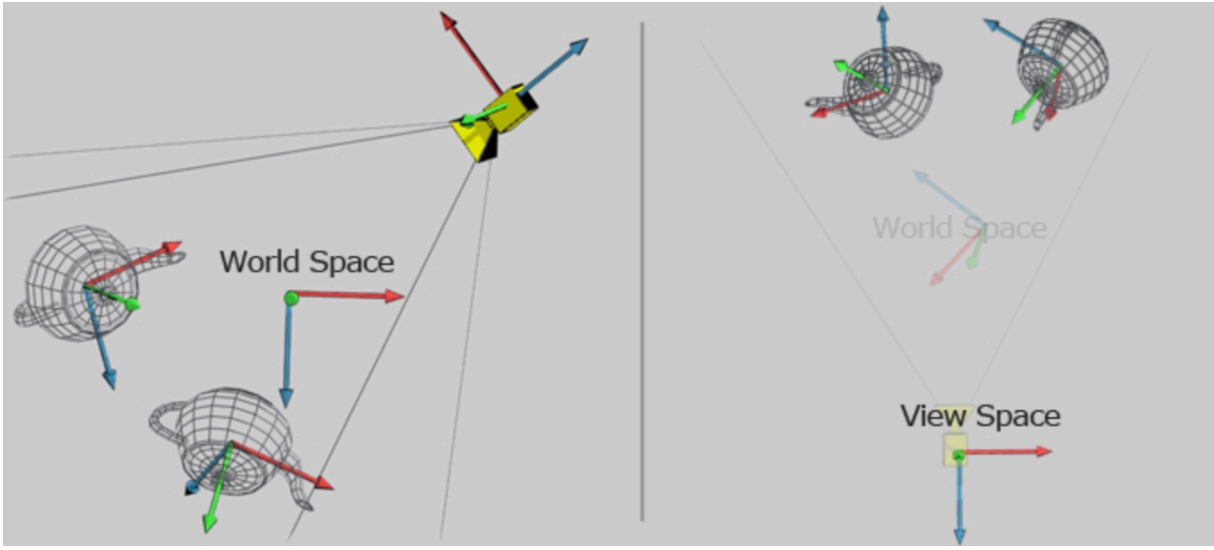
模型变换

局部坐标系(模型坐标系)是为了方便构造模型而设立的坐标系，建立模型时我们无需关心最终对象显示在屏幕哪个位置.

模型变换的主要目的是通过变换使得用顶点属性定义或者3d建模软件构造的模型，能够按照需要，通过缩小、平移等操作放置到场景中合适的位置。通过模型变换后，物体放置在一个全局的世界坐标系中，世界坐标系是所有物体交互的一个公共坐标系

视变换

视变换是为了方便观察场景中物体而设立的坐标系，在这个坐标系中相机是个假想的概念，是为了便于计算而引入的。相机坐标系中的坐标，就是从相机的角度来解释世界坐标系中位置



OpenGL 中相机始终位于原点，指向 -Z 轴，而以相反的方式来调整场景中物体，从而达到相同的观察效果。例如要观察 -Z 轴方向的一个立方体的右侧面，可以有两种方式：

1. 立方体不动，让相机绕着+y轴，旋转+90度，此时相机镜头朝向立方体的右侧面，实现目的。完成这一旋转的矩阵记作$R_y(\frac{\pi}{2})$
2. 相机不动，让立方体绕着+y轴，旋转-90度，此时也能实现同样的目的。注意这时相机没有转动。完成这一旋转的矩阵记作$R_y(-\frac{\pi}{2})$