一、顶点与几何

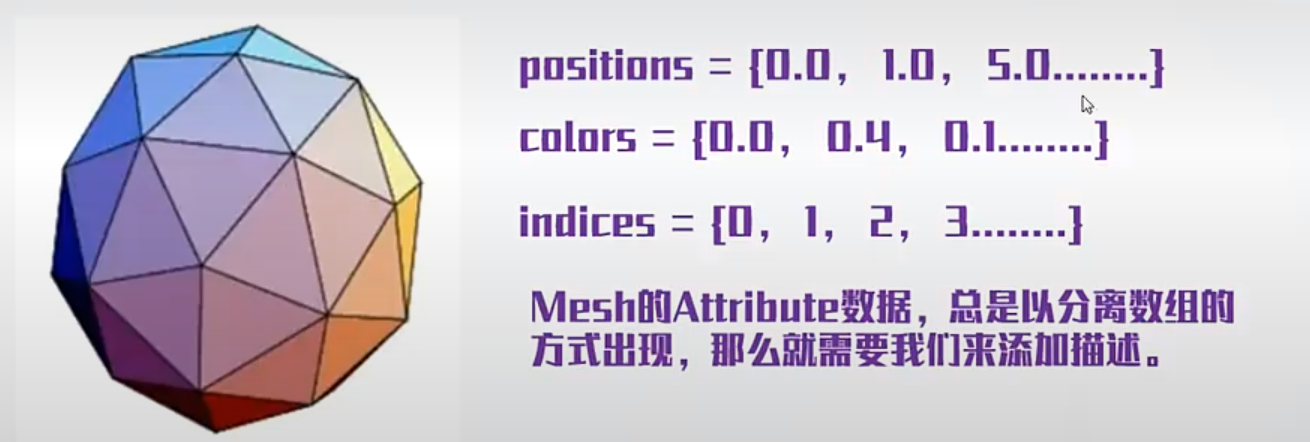
Attribute:顶点上的所有属性（位置、颜色）

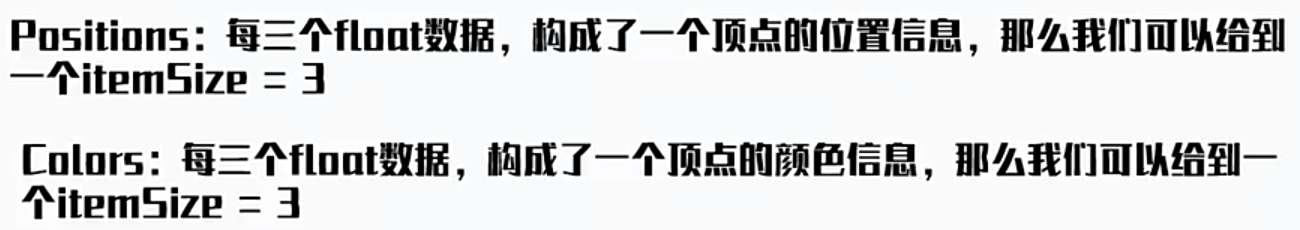
1.1如何描述三角形内部像素呢？

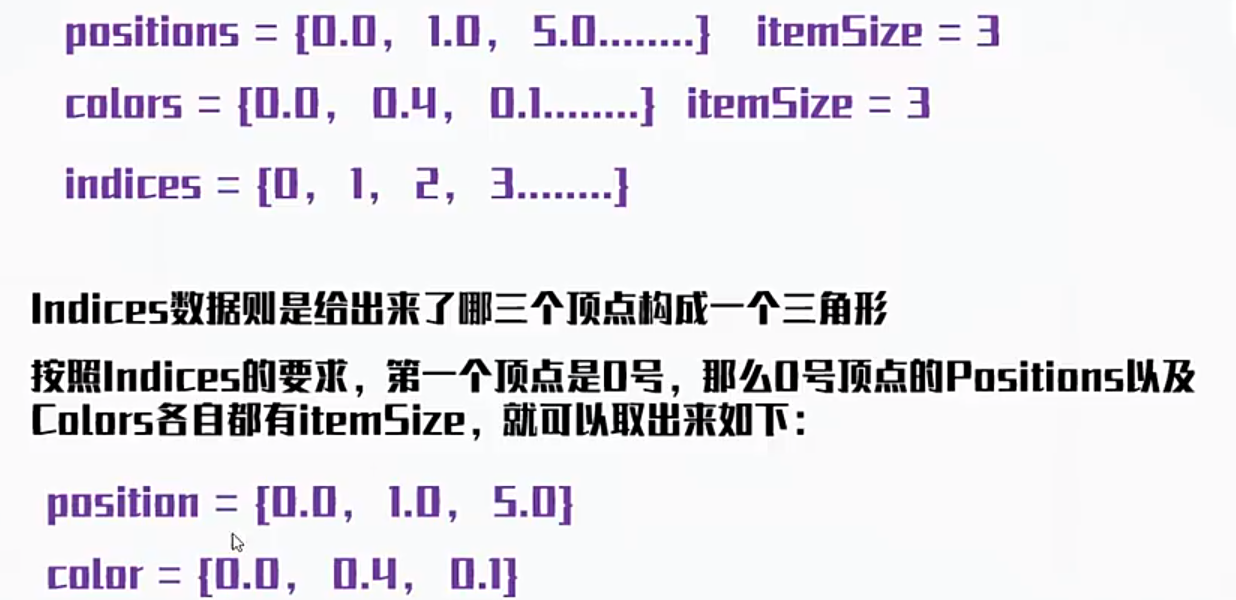
插值，通过GPU对三角形光栅化之后，得到的每一个像素点，都会有自己的重心坐标，重心坐标就是它的插值比例

Mesh:多个三角形构成的平面，立方体、球体、各类模型，都可以称之为一个Mesh。由多个三角形拼合而成，各个Attribute分离存放，各自是一个数组，并且配备描述信息

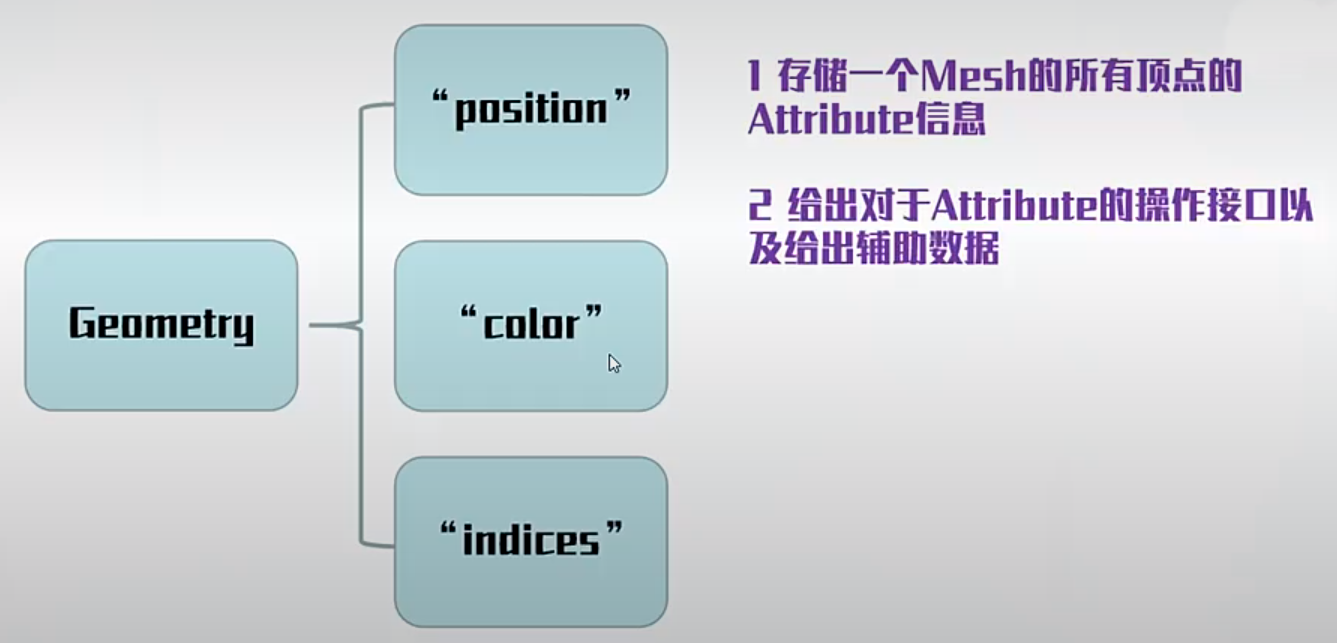
MeshAttribute描述方式：





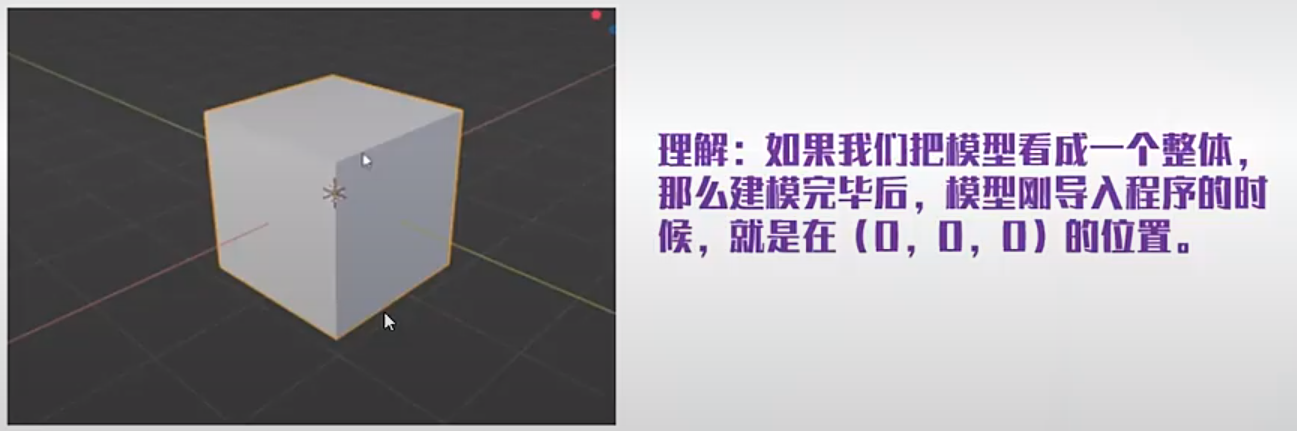


Geometry(几何)：描述一个Mesh的所有Attribute属性，并且提供各类附加函数以及数据功能



1. 矩阵变换

模型坐标系：一般用笛卡尔坐标系。分为左手坐标系和右手坐标系（只用X轴是反的，OPENGL是右手坐标系）。按照实际工程来说，在建模时模型会有一个坐标系，所有的顶点都在这个坐标系下，都有自己的位置（xyz）



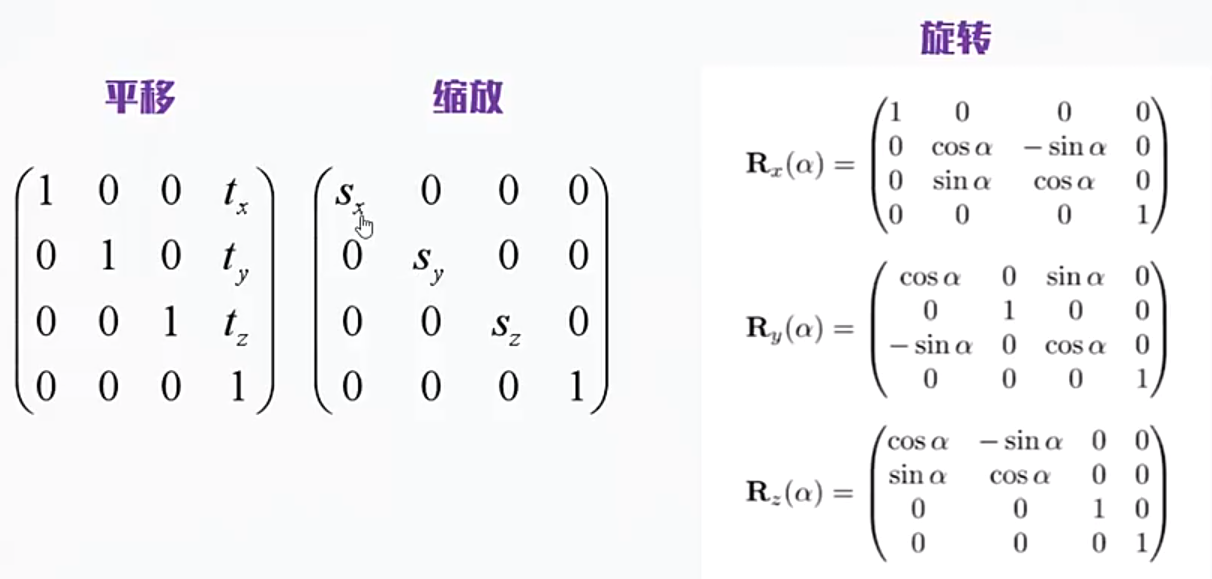
世界坐标系：我们把一个模型导入到我们的场景世界中，那么场景当中就必然存在一个坐标系，这个就是世界坐标系。



坐标系的相对性：我们一开始拿到的模型坐标，都是在模型坐标系当中的坐标，如

果将整个模型坐标系旋转后移动，那么Me5h顶点的模型坐标系下的坐标不变，但是世界坐标系下的坐标就会变化。

MatrixTransform：所有顶点的位置，都可以通过矩阵在世界坐标系进行移动、旋转、缩放。



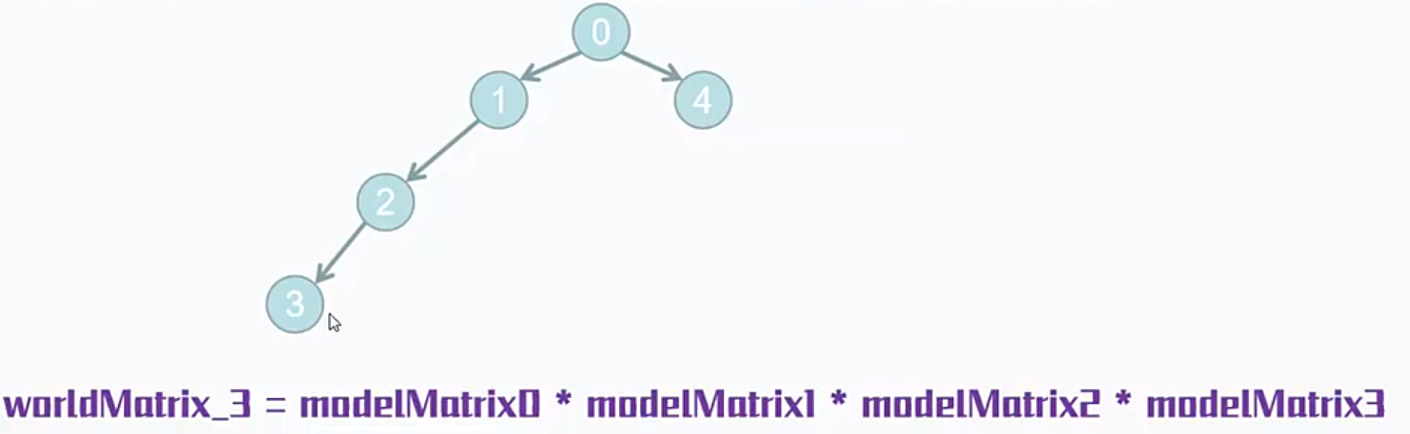
ModelMatrix：对模型坐标系施加的一系列平移、旋转、缩放等矩阵，相乘后综合起来，就是我们所说的模型变换矩阵。ModelMatrix：将物体的顶点坐标，由模型坐标系变换到自己父节点的坐标系内，如果没有父节点，则相当于变换到世界坐标系中。



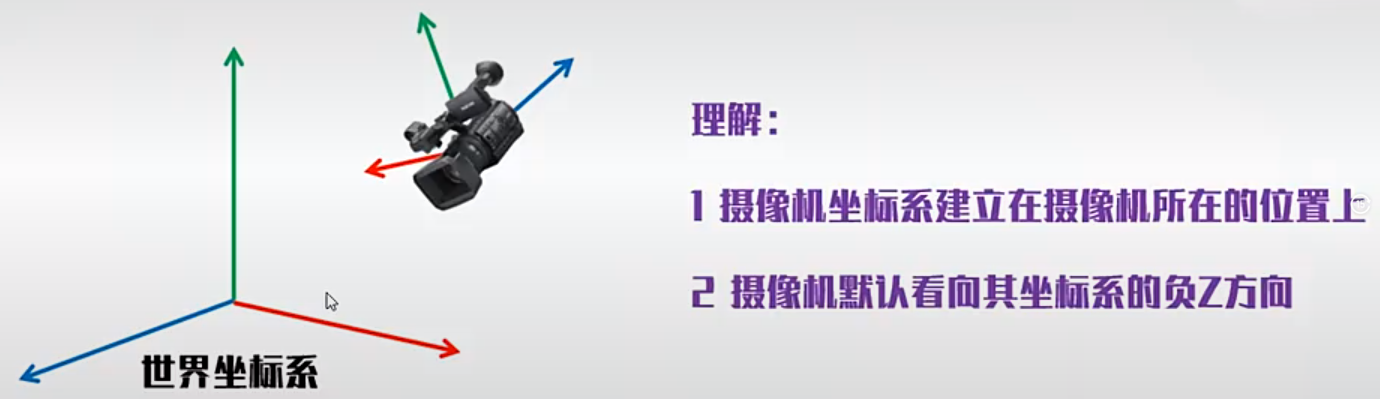
嵌套层级：如果一个物体被“绑定”在另一个物体上，那么其模型矩阵，是将本物体顶点坐标，变换到父节点所在的坐标系内。



WarldMatrix：将物体以及其所有上层节点的ModelMatrix顺序相乘得到的矩阵，用于把物体的顶点从模型坐标系直接变换到场景世界坐标系中。



摄像机矩阵：在将物体经过一系列变换，其顶点变为世界坐标系下的坐标后。需要将其转换到摄像机坐标系下，这样的坐标系被称为ViewMatrix

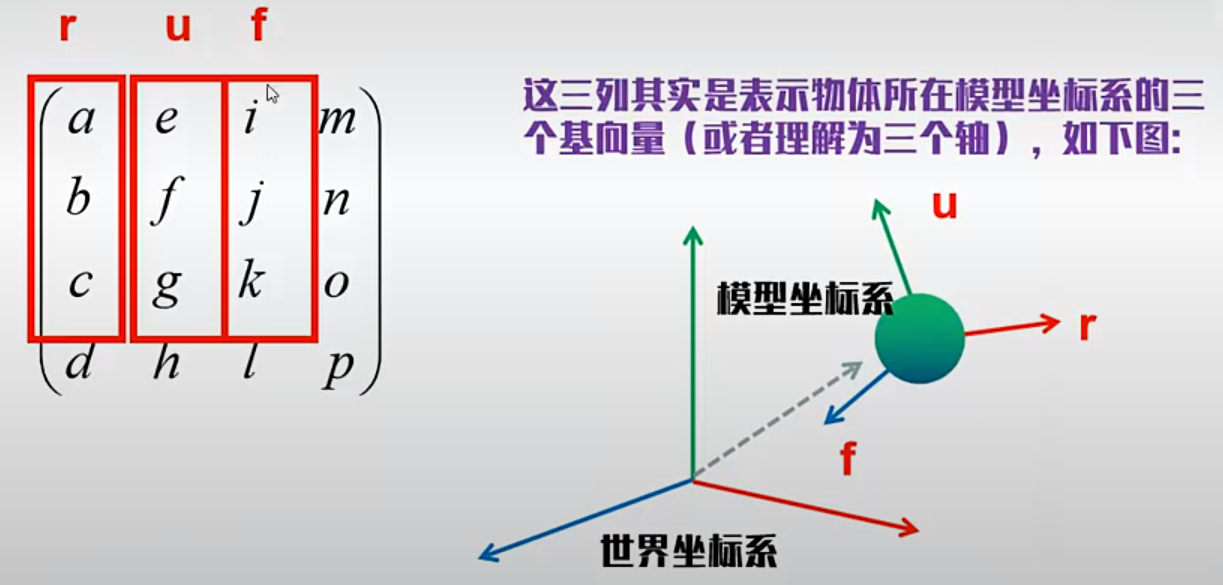


我们把Camera看作一个口bject,那么它就会有自己的ModelMatrix以及WorldMatrix。

WorldMatrix意义：将物体从本身的模型坐标系，转化到世界坐标系当中。WorldMatrix的Inverse，肯定就是将某个坐标，从世界坐标系转换到摄像机的模型坐标系之中了，就是我们需要的ViewMatrix

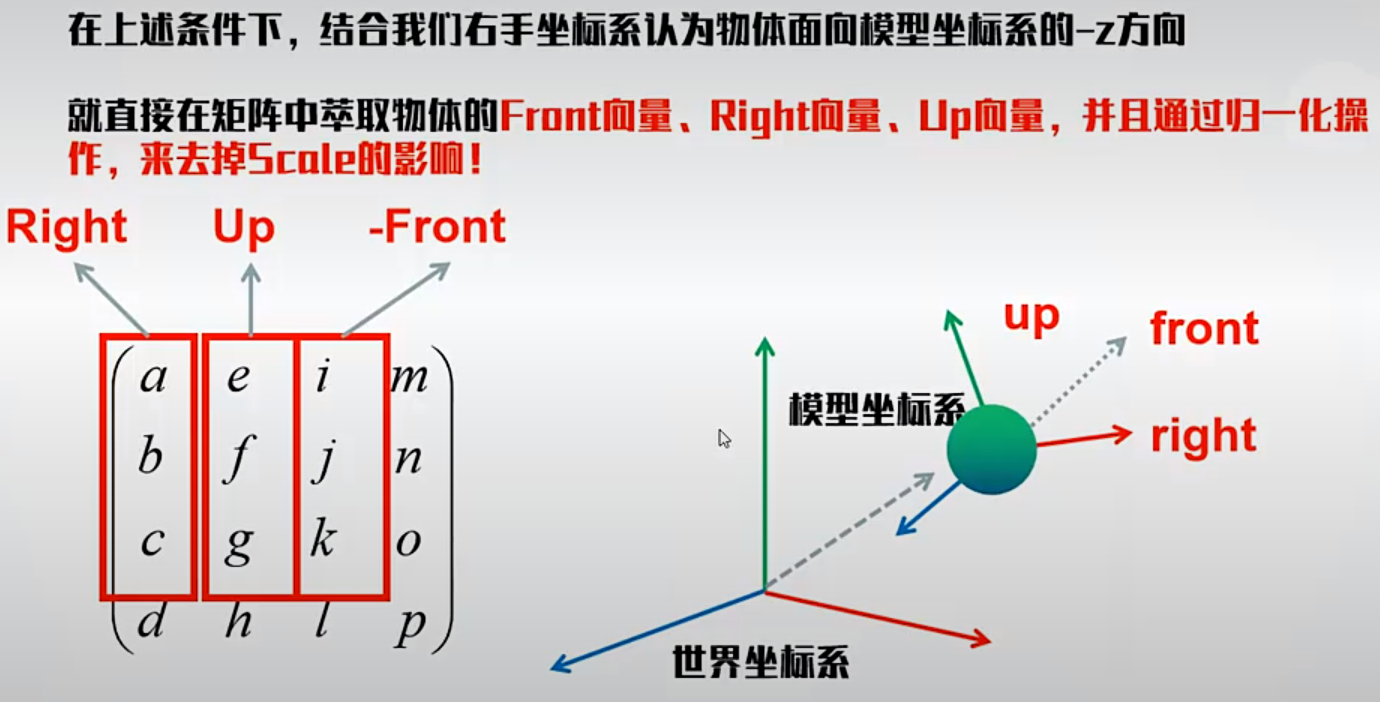
拿出一个Object的WorldMatrix研究，找到所有的含义





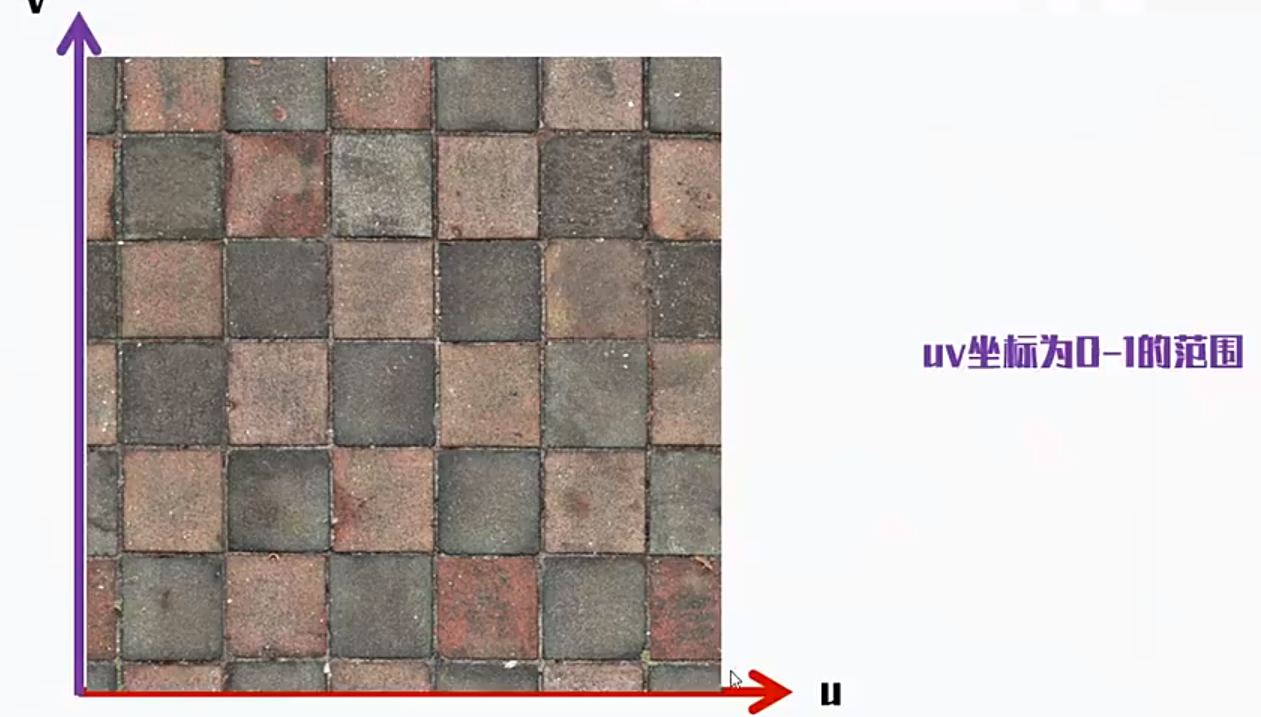
如果ruf的长度井不是l，即非归一化

则其长度，就代表了5caleX，ScaleY，Scalez

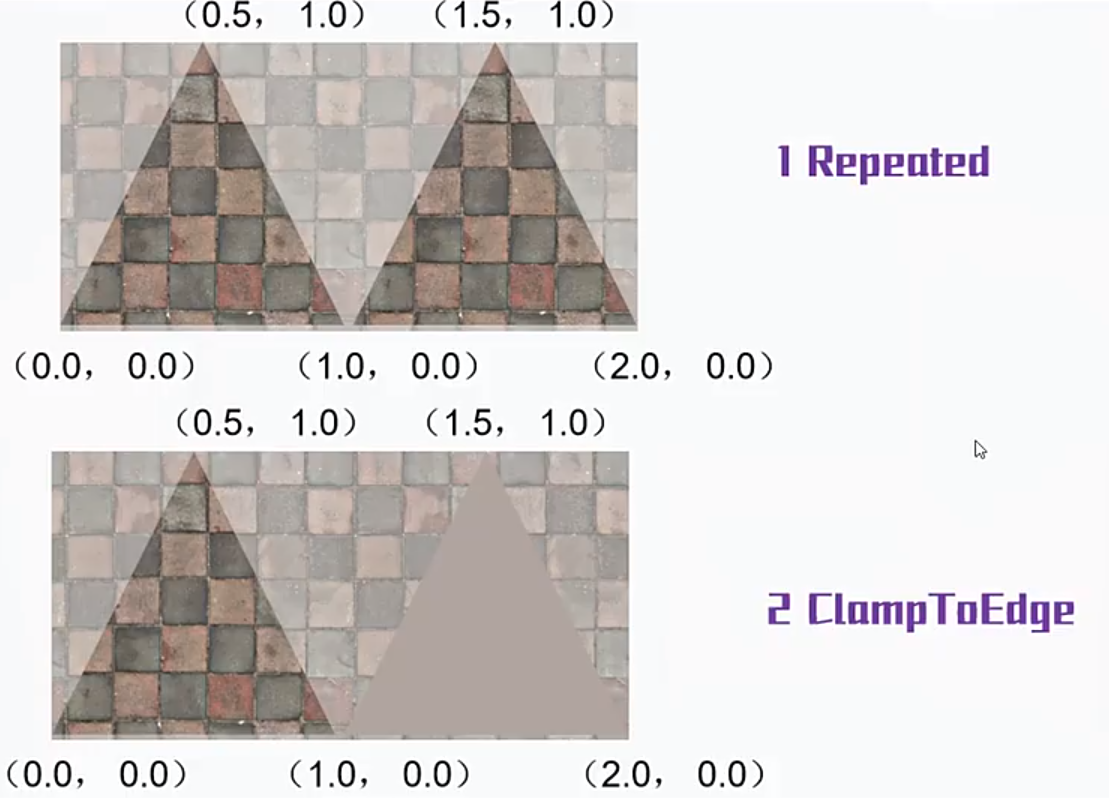


1. 纹理

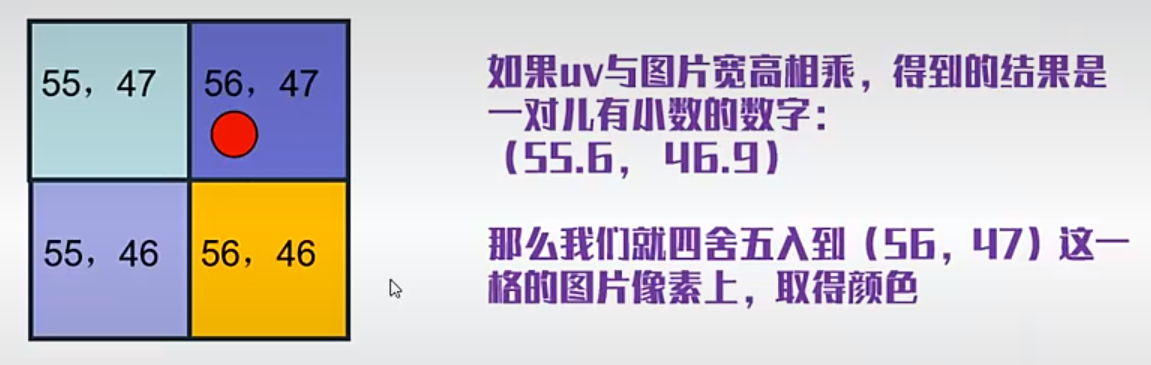
纹理坐标：附着在顶点上的Attribute，用于标定本顶点位于纹理贴图上的位置。纹理坐标表示的是采样比例；纹理坐标乘以图片的宽高，来决定取得哪一个像素的颜色；三角形内部片元通过插值构成



纹理的wrap



纹理的Filter：



但是这么做是有问题的



原因：拉近之后，纹理贴图的精度不足以支撑如此细节的采样。即采样频率过高

解决方法：双线性插值

结论：

在跟贴图距离拉近的时候，我们为了防止图片分辨率不足（样本不足）的问题，

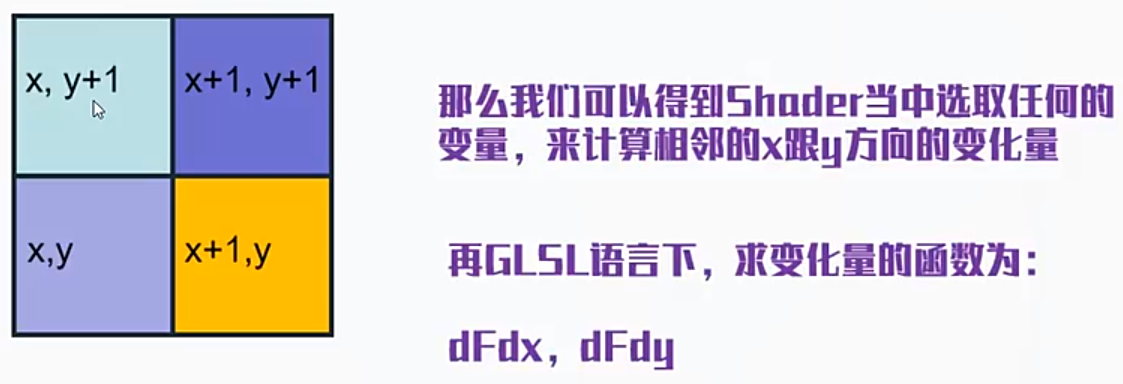
采用LinearFilter

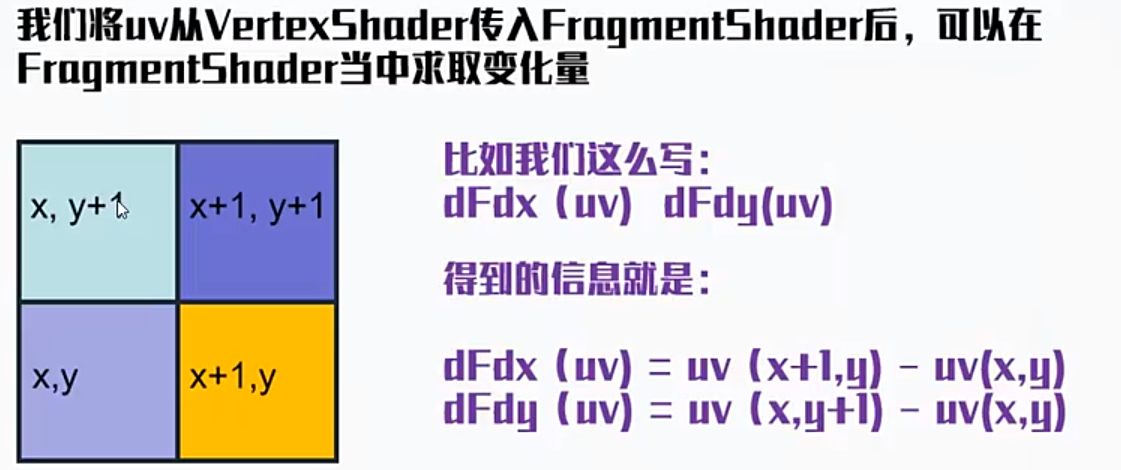
己在跟贴图距离拉远的时候，我们为了防止图片采样频率不足，会变糊，则采用更

加锐和的NearestFilter

3.1GPU是如何判断距离贴图实在拉近，还是在拉远呢？

Fragment处理的原理：GPU并不是按照每一个像素单独进行计算处理，而是每四个像素为一组，进行处理。





标准变化率的计算：

我们既然已经得到了当下相邻像素的uv变化率，那么就需要跟一

个标准变化率对比，这个变化率可以这么计算：

dx = 1.0/width

dy = 1.0/height

结论：

dFdx(uv) dFdy(uv)分别与（dx,dy)对比,如果其绝对值更大,那么就是进入了MinFilter设定的滤波方式；如果其绝对值更小，那么就进入了MagFilter设定的滤波方式。

纹理的其他作用：

纹理除了作为贴图，表示物体本身的色彩之外，还拥有各种功能：

1一个空白的纹理图片，可以作为一个RenderTarget的Attachment

2纹理的RG日通道可以存储法线的信息，从而作为法线贴图，描述模型细节

3纹理可以用其R通道，传入数值，表示物体每个片元着色的时候之镜面反射强度

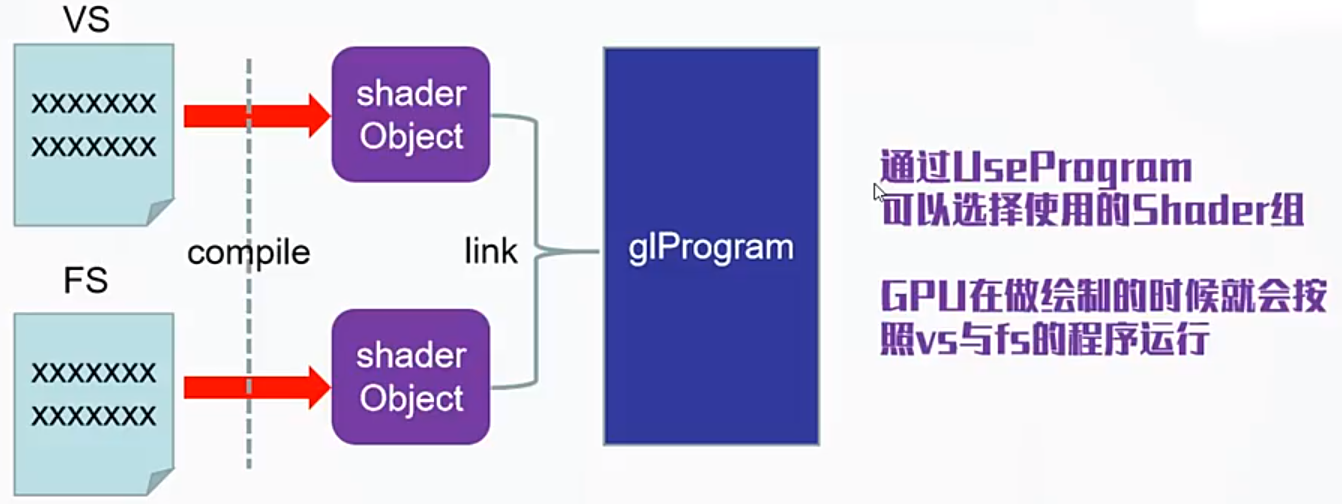
4一个物体可以贴多个纹理，互相之间RG日相乘，创造独特的效果

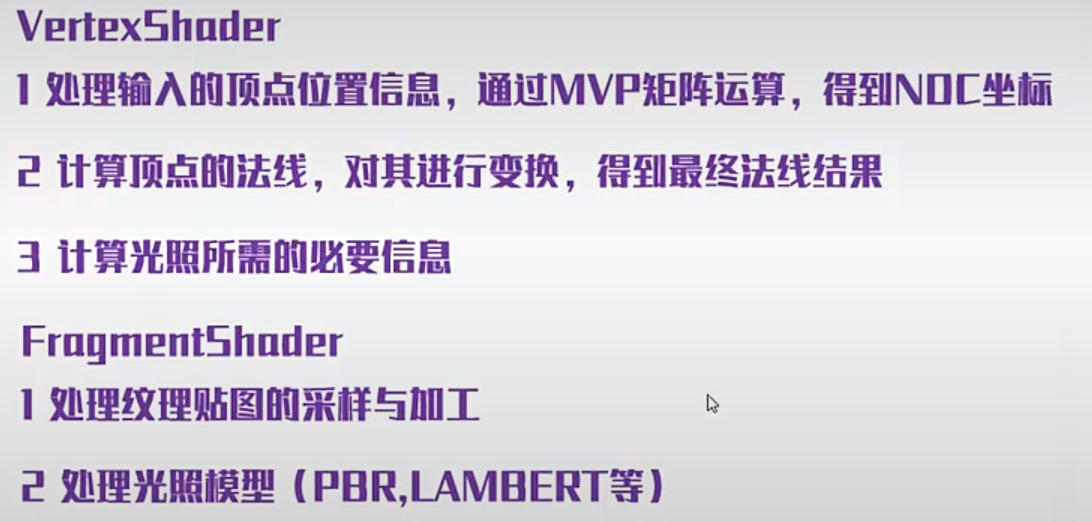
5纹理可以作为阻影贴图，传入FragmentShader，制造阻影效果

6PBR下，纹理拥有更为复杂的用途。

1. Shader

现代GPU提供了一系列的可编程定制功能，来定义顶点数据处理、片元数据处理等功能，用于做这些处理的程序，即为shader;shader是一套代码组成的字符串，那么与c++相同，都需要编译与链接，最终形成可执行文件

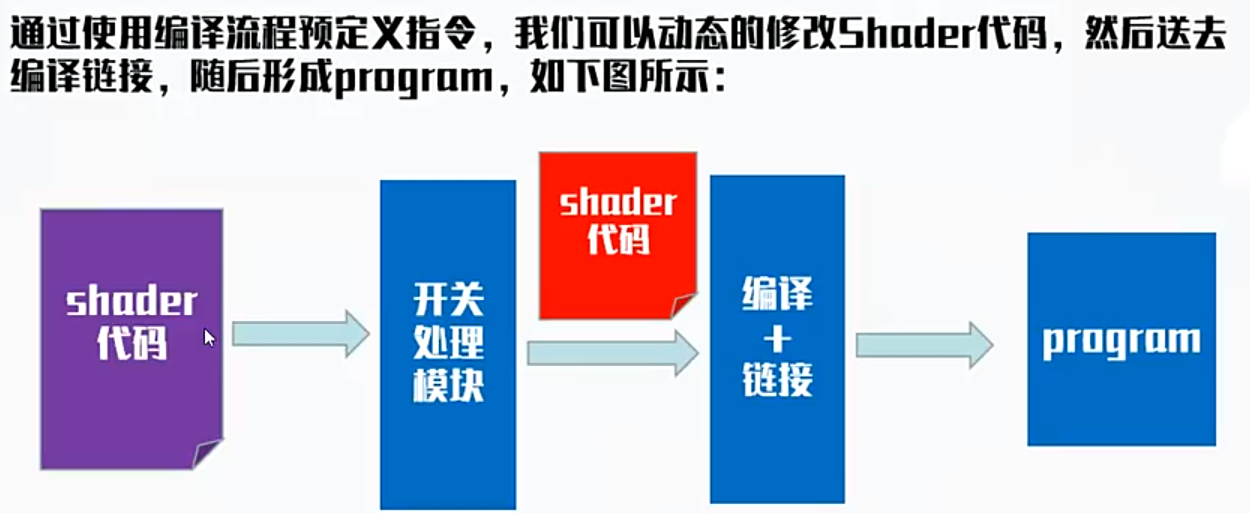




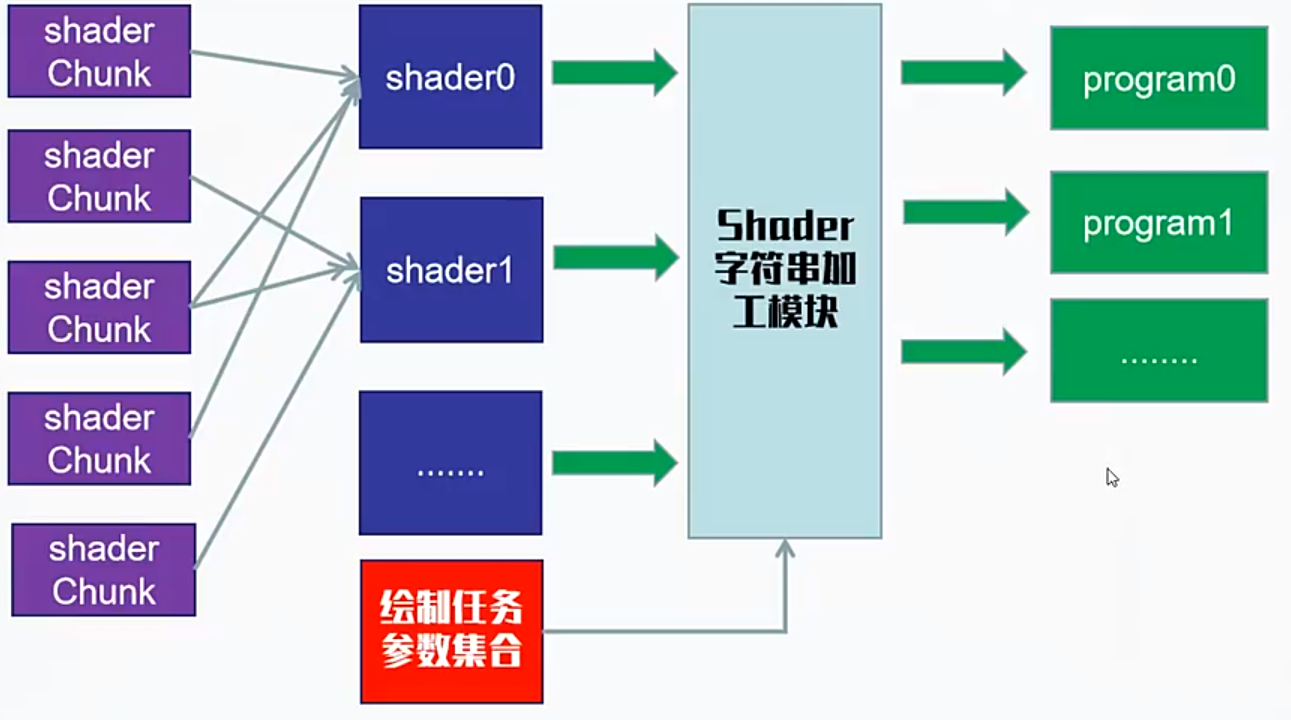
4.1Shader如何实现的？/丰富的功能，如果都写在同一个程序文档里面，就会复杂，且服用低效。

可以根据Shader代码的功能，将其进行拆分成ShaderChunk。随后可以进行按需组合

shader控制开关：Shader当中拥有很重要的功能，就是控制代码编译的预处理指令，类似C++的#ifdef #if 等。



Shader模块概览：



1. 材质

材质：一个物体本身的着色属性参数集合，诸如：是否参与光照计算、是否拥有贴图、是否拥有深度、是否透明、是否启用表面剔除等等。

材质决定使用怎样的shader来进行物体的绘制

材质属于一个单独的class,里面可以包含贴图的智能指针

管线当中的透明渲染方式的设置，也属于材质类

材质当中也会进行光栅化阶段的参数设置

材质的种类，直接决定了使用哪一个shader组

深度设置：用于计算当前最接近摄像机的深度值

Uniform是在CPU端和GPU端传输变量的参数类型

Uniform与Attribute区别：Attribute是随着顶点进入渲染管线，每个VS处理的顶点数据都不同，Uniform是对所有本次口Oraw操作的VS/FS都相同

