# 3D图形学基础

## 1前言

本文讲述一些普及知识

## 2什么是图形学

Computer Graphics简称CG。研究和图形有关的一切，显示存储处理。通常VR或是游戏应用到的主要是一个分支：实时渲染（RealtimeRendering）。

渲染：用软件从模型生成到图像。模型是对三维物体和虚拟场景的描述。主要包括几何、纹理、光照、阴影、视点等信息。在图形流水线中，渲染是最后一项重要的步骤。

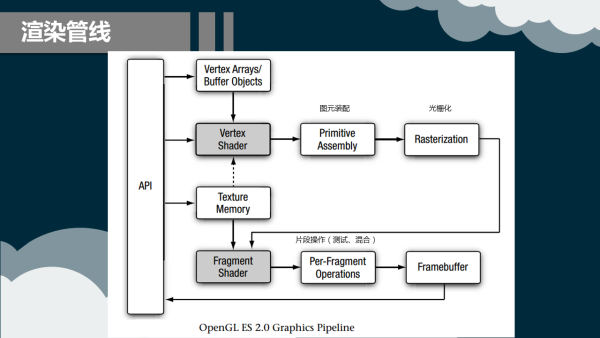
## 3渲染管线

### 3.1 什么是渲染管线？

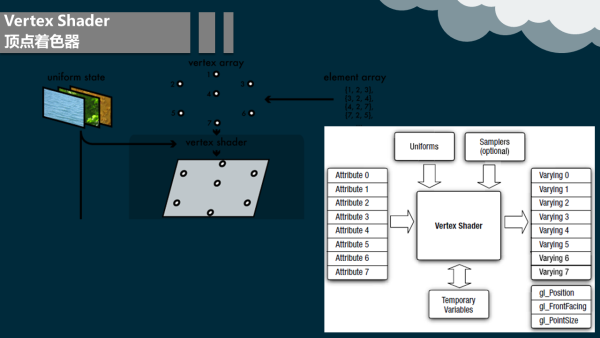
用硬件电路给图形配上颜色。是GPU内部处理图形信号的处理单元。

### 3.2渲染管线的流程

#### 3.2.1总流程图



#### 3.2.2顶点着色器

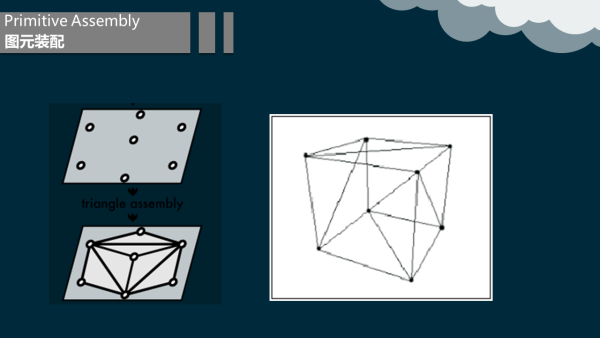


渲染对象的形状和位置，还要模拟外观，包括对象的材质，光源对对象产生的特效。

模拟材质和光源的方法有很多种，包括从最简单的颜色到精细的物理特性描述。决定光和材质特效的操作称为着色（shader）。这类计算运行在几何的模型顶点数组上，另一些运行在逐像素的光栅化阶段。

顶点着色的结果会被送进光栅去插值。

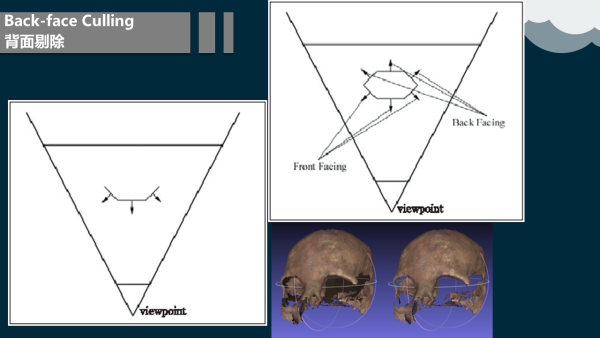
#### 3.2.3图元装配



图元装配：经过变换的顶点按照顺序被送到下一个。首先把顶点序列的几何图元的分类信息装配成几何图元。这将产生一系列的三角，线段和点。

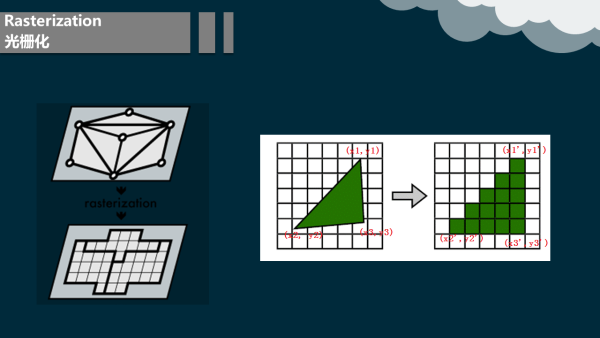
这些图元需要经过裁剪，映射到一个可视平截体和任何有效地应用程序指定的裁剪平面。

#### 3.2.4背面剔除



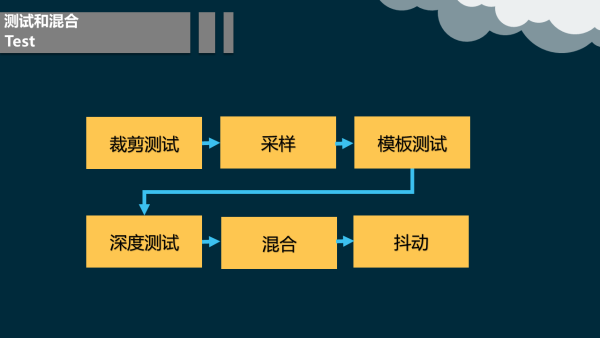
渲染的基本单位是三角面皮。固有正面背面。只渲染正面会提高很多效率。

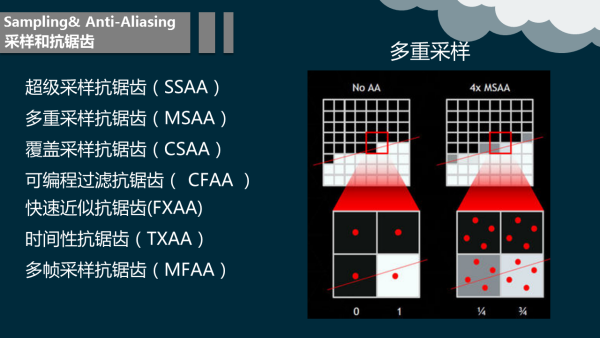
#### 3.2.5光栅化



得到变换和投影后的顶点及与之相关联的着色数据，计算并设置像素的颜色。

#### 3.2.6测试与混合





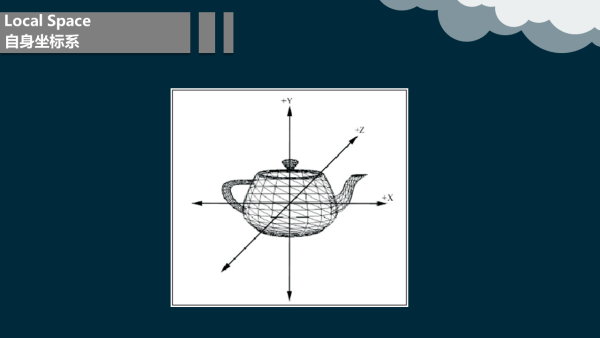
深度测试，一般把深度称为Z值。

#### 3.2.7混合

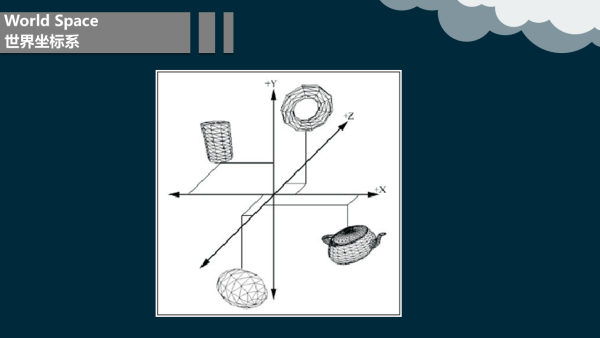


## 4坐标变换

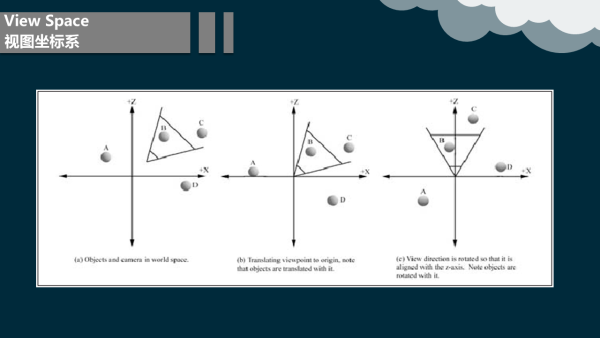
### 4.1自身坐标系



### 4.2世界坐标系

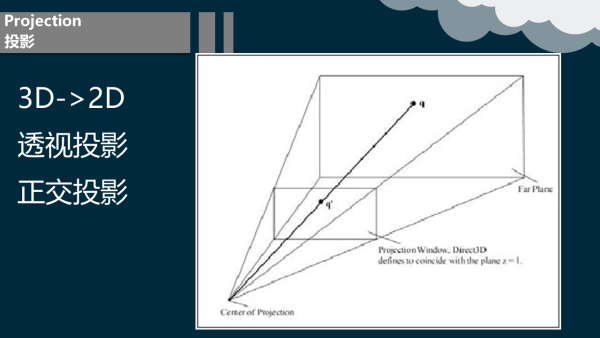


### 4.3视图坐标系



世界坐标系中的几何图与照相机是相对于世界坐标系而定义的，如图2.10所示。然而在世界坐标系中当照相机是任意放置和定向时，投影和其它一些操作会变得困难或低效。为了使事情变得更简单，我们将照相机平移变换到世界坐标系的源点并把它的方向旋转至朝向Z轴的正方向，当然，世界坐标系中的所有物体都将随着照相机的变换而做相同的变换。这个变换就叫做视图坐标系变换（view space transformation）。

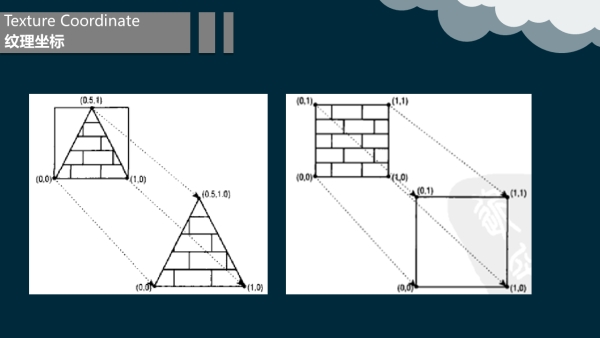
### 4.4投影



## 5色彩和纹理

一个纹理实际上就是一个位图。从这个意义上来讲，当纹理一词被用于计算机图形学时，它就有了一个明确的定义。从语义学角度来讲，纹理一词既是指一个物体上颜色的模式，又是指物体表面是粗糙的还是光滑的。

### 5.1纹理坐标



纹理实际上是一个二维数组，它的元素是一些颜色值。单个的颜色值被称为纹理元素（texture elements）或纹理像素（texel）。每一个纹理像素在纹理中都有一个唯一的地址。这个地址可以被认为是一个列（column）和行（row）的值，它们分别由U和V来表示。纹理像素地址会被映射到对象的坐标系中。再被平移到屏幕坐标系或像素位置上。

### 5.2纹理过滤

对于图元在二维屏幕上图象的每个像素来说，都必须从纹理中获得一个颜色值。我们把这一过程称为纹理过滤（texture filtering）。渲染一个图元时，会将三维图元映射到二维屏幕上。如果图元有纹理，就必须用纹理来产生图元的二维渲染图象上每个像素的颜色。对于图元在二维屏幕上图象的每个像素来说，都必须从纹理中获得一个颜色值。我们把这一过程称为纹理过滤（texture filtering）

### 6粒子系统

