河北师范大学软件学院
Software College of Hebei Normal University

Android源生AR应用开发

第三讲 着色器与坐标映射

智能设备教研室

目录

着色器

- **着色器**（Shader）其实就是一段执行在GPU上的程序，此程序使用OpenGL ES SL语言来编写。它是一个描述顶点或像素特性的简单程序。在OpenGL ES中常用的Shader有两种：
  - 顶点着色器（Vertex Shader）
  - 片元着色器（Fragment Shader）

着色器 – 顶点着色器

❖**顶点着色器**（Vertex Shader）

对于发送给GPU的每一个Vertex(顶点)，都要执行一次Vertex Shader。其功能是把每个顶点在虚拟空间中的三维坐标变换为可以在屏幕上显示的二维坐标，并带有用于z-buffer的深度信息。Vertex Shader可以操作的属性有：位置、颜色、纹理坐标，但是不能创建新的顶点。

着色器 – 顶点着色器

- Vertex Shader主要完成以下工作：
  - 基于点操作的矩阵乘法位置变换
  - 根据光照公式计算每点的color值
  - 生成或者转换纹理坐标

着色器 – 顶点着色器

- 在顶点着色器阶段至少应输出位置信息的内建变量：

  gl_Position。

- gl_Position：变换后的顶点的位置，用于后面的固定的裁剪等操作。所有的顶点着色器都必须写这个值。

着色器 – 顶点着色器

❖顶点着色器可用的内置变量如下表：

名称	类型	描述
gl_Color	vec4	输入属性-表示顶点的主颜色
gl_SecondaryColor	vec4	输入属性-表示顶点的辅助颜色
gl_Normal	vec3	输入属性-表示顶点的法线值
gl_Vertex	vec4	输入属性-表示物体空间的顶点位置
gl_MultiTexCoordn	vec4	输入属性-表示顶点的第n个纹理的坐标
gl_FogCoord	float	输入属性-表示顶点的雾坐标
gl_Position	vec4	输出属性-变换后的顶点的位置，用于后面的固定的裁剪等操作。所有的顶点着色器都必须写这个值。

着色器 – 顶点着色器

名称	类型	描述
gl_ClipVertex	vec4	输出坐标，用于用户裁剪平面的裁剪
gl_PointSize	float	点的大小
gl_FrontColor	vec4	正面的主颜色的varying输出
gl_BackColor	vec4	背面主颜色的varying输出
gl_FrontSecondaryColor	vec4	正面的辅助颜色的varying输出
gl_BackSecondaryColor	vec4	背面的辅助颜色的varying输出
gl_TexCoord[]	vec4	纹理坐标的数组varying输出
gl_FogFragCoord	float	雾坐标的varying输出

着色器 – 片元着色器

❖**片元着色器**（Fragment Shader）

片元着色器计算每个像素的颜色和其它属性。通过应用光照值、凹凸贴图，阴影，镜面高光，半透明等处理来计算像素的颜色并输出。它也可改变像素的深度(z-buffer)或在多个渲染目标被激活的状态下输出多种颜色。一个Fragment Shader不能产生复杂的效果，因为它只在一个像素上进行操作，而不知道场景的几何形状。。

着色器 – 片元着色器

- 片元着色器可用的内置变量如下表：

名称	类型	描述
gl_Color	vec4	包含主颜色的插值只读输入
gl_SecondaryColor	vec4	包含辅助颜色的插值只读输入
gl_TexCoord[]	vec4	包含纹理坐标数组的插值只读输入
gl_FogFragCoord	float	包含雾坐标的插值只读输入
gl_FragCoord	vec4	只读输入，窗口的x,y,z和1/w
gl_FrontFacing	bool	只读输入，如果是窗口正面图元的一部分，则这个值为true

着色器 – 片元着色器

名称	类型	描述
gl_PointCoord	vec2	点精灵的二维空间坐标范围在(0.0, 0.0)到(1.0, 1.0)之间，仅用于点图元和点精灵开启的情况下。
gl_FragData[]	vec4	使用glDrawBuffers输出的数据数组。不能与gl_FragColor结合使用。
gl_FragColor	vec4	输出的颜色用于随后的像素操作
gl_FragDepth	float	输出的深度用于随后的像素操作，如果这个值没有被写，则使用固定功能管线的深度值代替。

着色器

- 变量修饰符

  在 GLSL 中，实际有三种标签可以赋值给我们的变量：

  - Uniforms
  - Attributes
  - Varyings

着色器

- **Uniforms** 是一种外界和你的着色器交流的方式。

  Uniforms 是为在一个渲染循环里不变的输入值设计的。如果你正在应用茶色滤镜，并且你已经指定了滤镜的强度，那么这些就是在渲染过程中不需要改变的事情，你可以把它作为 Uniform 输入。 Uniform 在顶点着色器和片段着色器里都可以被访问到。

着色器

- **Attributes** 仅仅可以在顶点着色器中被访问。Attribute 是在随着每一个顶点不同而会发生变动的输入值，例如顶点的位置和纹理坐标等。顶点着色器利用这些变量来计算位置，以它们为基础计算一些值，然后把这些值以 Varyings 的方式传到片段着色器。

着色器

- **Varying** 在顶点着色器和片段着色器都会出现。Varying 是用来在顶点着色器和片段着色器传递信息的，并且在顶点着色器和片段着色器中必须有匹配的名字。数值在顶点着色器被写入到 Varying ，然后在片段着色器被读出。被写入 Varying 中的值，在片段着色器中会被以插值的形式插入到两个顶点直接的各个像素中去。

着色器

- 着色器的一般使用流程如下：
  1. 创建着色器：glCreateShader
  2. 指定着色器源代码字符串：glShaderSource
  3. 编译着色器：glCompileShader
  4. 创建着色器可执行程序：glCreateProgram
  5. 向可执行程序中添加着色器：glAttachShader
  6. 链接可执行程序：glLinkProgram

着色器

```
int glCreateShader(int type);
```

创建一个指定类型的着色器

type：

- GLES20.GL_VERTEX_SHADER：顶点着色器
- GLES20.GL_FRAGMENT_SHADER：偏远着色器

着色器

```
void glShaderSource(int shader,
                    String str);
```

将源码添加到着色器

shader：着色器索引

str：着色器源码字符串

着色器

```
void glCompileShader(int shader);
```

编译着色器

shader：着色器索引

着色器

```
int glCreateProgram();
```

创建一个OpenGL ES渲染程序

着色器

```
void glAttachShader(int program,
                    int shader);
```

将编译好的着色器添加到渲染程序

program：着色程序索引

shader：着色器索引

着色器

```
void glLinkProgram(int program);
```

链接渲染程序

program：着色程序索引

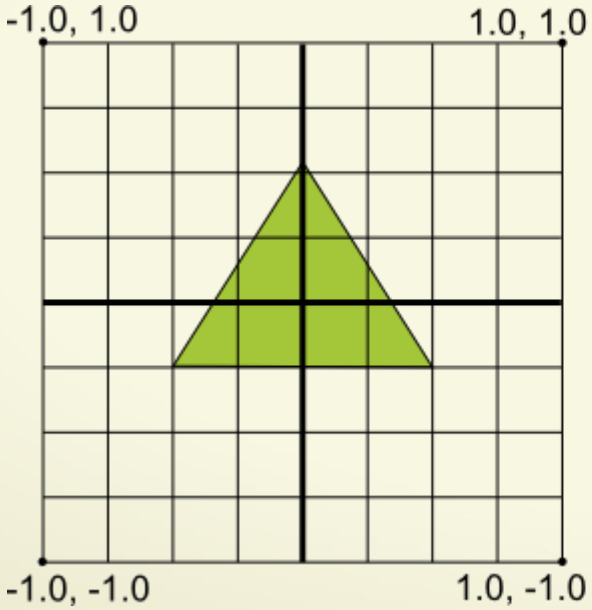
目录

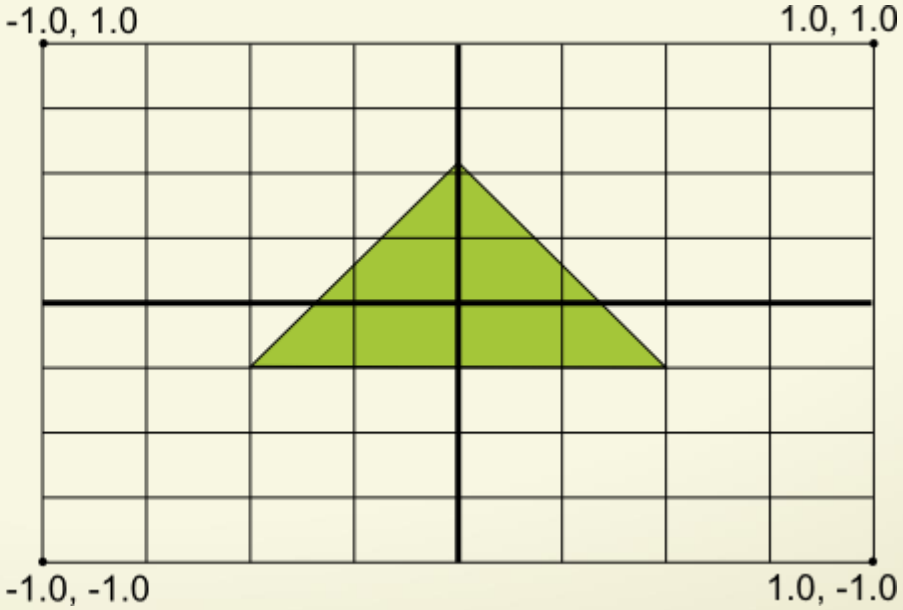
坐标映射

- 在Android设备上显示图形的一个基本问题是它们的屏幕在大小和形状上可以变化。OpenGL ES假设一个正方形，统一的坐标系，并且默认情况下，愉快地将这些坐标绘制到通常的非方形屏幕上，就好像它是完全正方形的。这些坐标如何实际映射到实际的典型设备屏幕上，为了解决这个问题，可以应用OpenGL**投影模式**和**相机视图**来转换坐标，用来保证图形在任何显示器上具有正确的比例。

坐标映射

- 下图显示了左侧OpenGL框架所采用的统一坐标系，以及实际映射到右侧横向方向的典型设备屏幕的情况。





坐标映射 – 相机视图

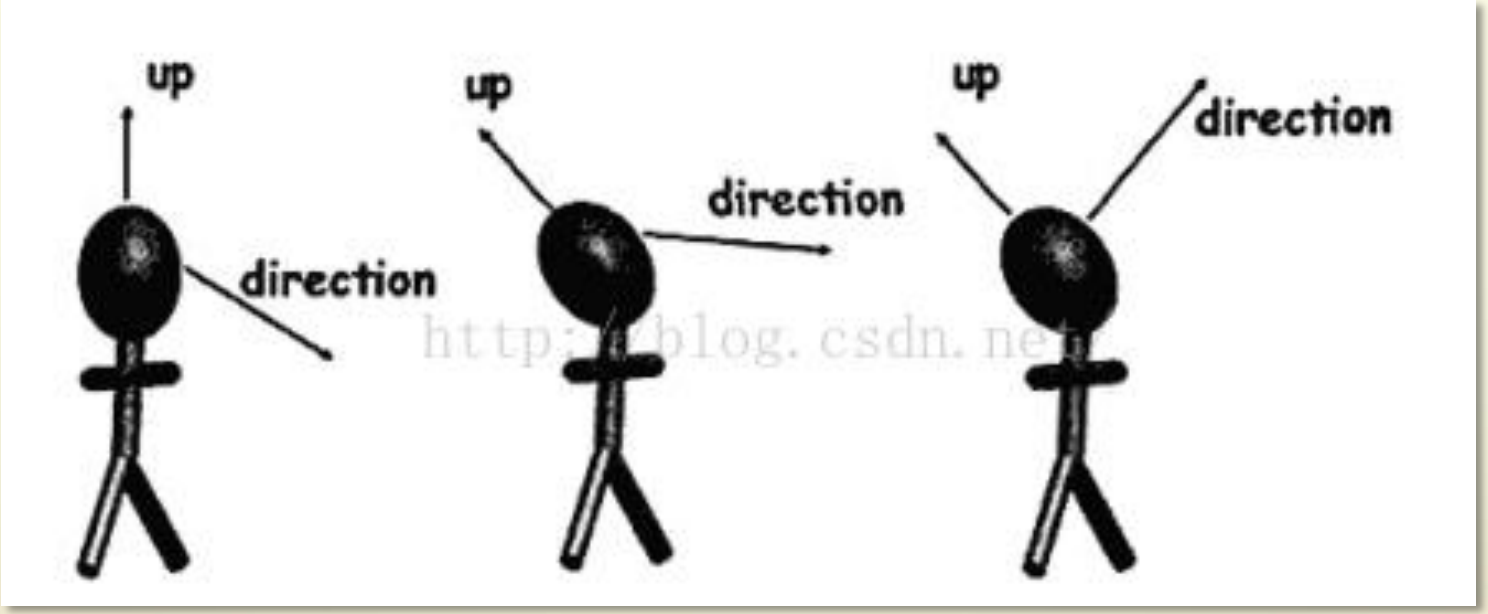
- 想想你手机的摄像头，它的位置不同，朝向不同，对同一个事物拍摄得到的画面肯定是不同的，OpenGL ES中的摄像头和我们日常生活中的摄像头是一样的道理。

坐标映射 – 相机视图

- 在Opengl中摄像头包含三部分的信息：
  1. 摄像头的位置 ，在三维空间中用 X Y Z表示
  2. 摄像头的镜头的指向，这里即观察的物体的坐标，一般选取物体的Center坐标（通过摄像头的位置与观察的物体的坐标可以确定一个向量，这个向量就可以决定观察的方向）
  3. 摄像头的UP方向，摄像机顶端的指向

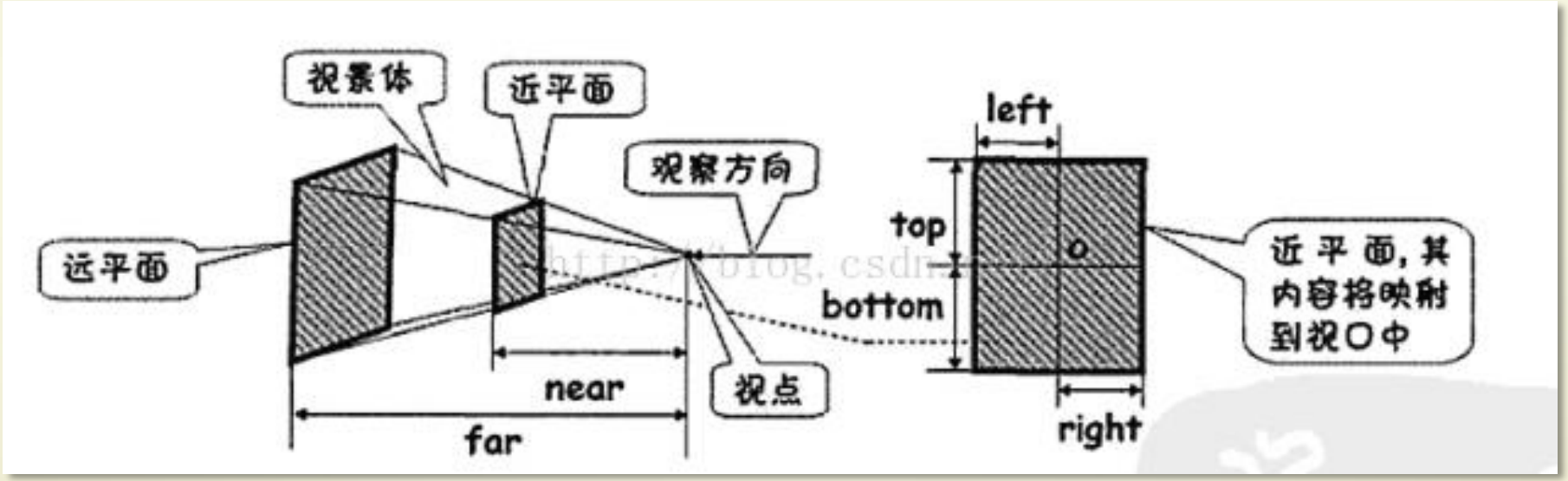
坐标映射 – 相机视图

- 下面的人眼观察物体的图示更容易帮助我们理解，可以看出摄像机的位置，朝向，UP方向有很多不同的组合，对于不同的组合观察同一物体会得到不同的结果。



坐标映射 – 投影模式

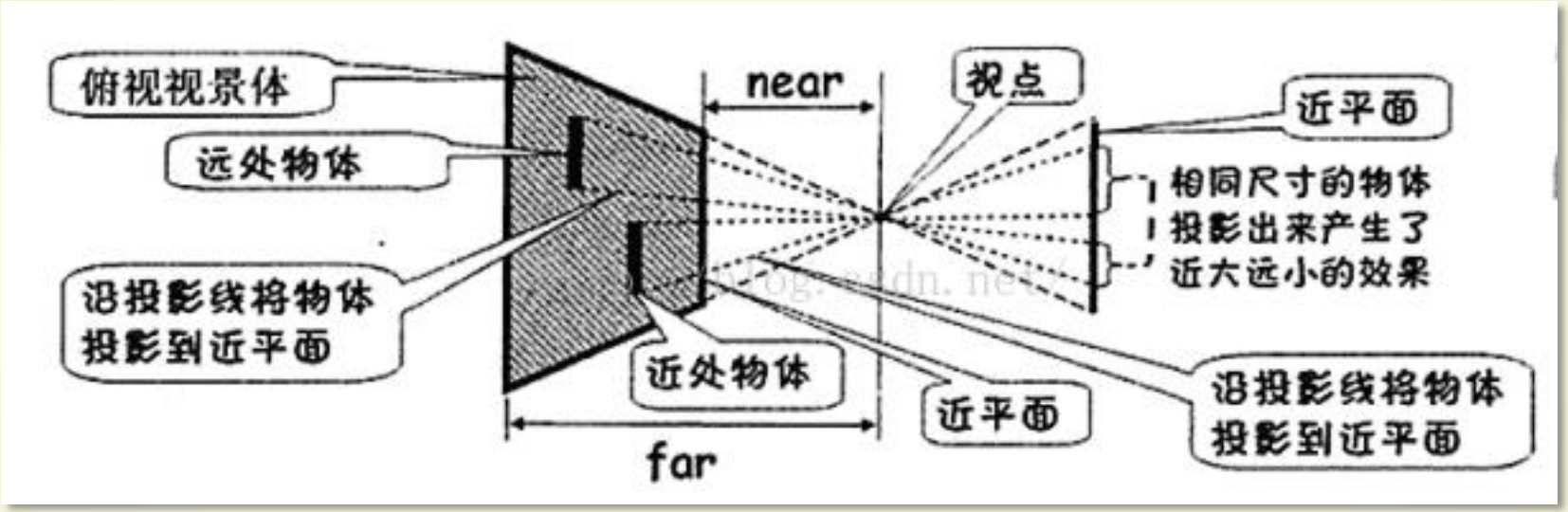
- 生活中观察物体，会有近大远小的效果，透视投影即为了产生这种效果，和美术中的透视是一个概念。



坐标映射 – 投影模式

- 其中，视点指摄像机的位置，近平面指距离视点较近的垂直于观察方向的平面，视景体又叫做视锥体为椎台形区域。
- 透视投影的投影线互不平行，相较于视点，因此对于同样尺寸的物体，在近处投影出来大，在远处投影出来小，由此产生近大远小的效果。

坐标映射 – 投影模式


俯视视景体
远处物体
沿投影线将物体
投影到近平面
near
far
近处物体
视点
近平面
近平面
相同尺寸的物体
投影出来产生了
近大远小的效果
沿投影线将物体
投影到近平面


THANK YOU