Android OpenGL ES开发

主讲人：杨丰盛

课程目标

◆Android书籍

➢Android应用开发书籍推荐

➢技术支持（交流平台）

◆课程计划

➢课程概述

➢功能演示

➢课程需求分析

HZ Books
华章培训

课程计划

Android OpenGL ES开发基础		
2:Opengl ES概述	13:离子系统	24:TGA文件
3:基本图形绘制	14:变形	25:多重视口
4:3D图形绘制及透视	15:多级纹理-二次几何体	26:轨迹球
5:光效	16:曲面映射	27:射线拾取
6:材质	17:多重纹理	28:地形
7:纹理及纹理映射	18:反射-蒙板缓存	29:天空盒
8:隧道实例	19:图像字体	30:帧动画
9:雾气	20:反走样	31:骨骼动画
10:2D文字显示	21:缓存及片元测试	32:碰撞检测
11:飘动的旗帜	22:贝塞尔曲面	引擎实现（通过NDK来开发原生OpenGL ES程序）
12:蒙板	23:BLT函数	

课程准备

构建Android SDK应用程序开发环境

- Eclipse(3.4及其以上版本)
- Android SDK（1.5以上）
- ADT（0.9以上）

课程目标

OpenGL ES概述

- OpenGL与OpenGL ES概述
- OpenGL与OpenGL ES区别

OpenGL ES开发框架

- GLSurfaceView
- Renderer

OpenGL 概述

OpenGL

是由SGI公司开发的一套3D图形软件接口标准，由于具有体系结构简单合理、使用方便、与操作平台无关等优点，OpenGL迅速成为一种3D图形接口的工业标准，并陆续在各种平台上得以实现。作为一个性能优越的图形应用程序设计界面（API）而适合于广泛的计算环境，从个人计算机到工作站和超级计算机，OpenGL都能实现高性能的三维图形功能。由于许多在计算机界具有领导地位的计算机公司纷纷采用OpenGL作为三维图形应用程序设计界面，OpenGL应用程序具有广泛的移植性。因此，OpenGL已成为目前的三维图形开发标准，是从事三维图形开发工作的技术人员所必须掌握的开发工具。

(http://www.opengl.org/)

OpenGL ES概述

OpenGL ES

OpenGL ES是专为内嵌和移动设备设计的一个 2D/3D轻量图形库，它是基于OpenGL API设计的。OpenGL ES 1.0版基于OpenGL 1.3，而OpenGL ES 1.1则是基于OpenGL 1.5的。。

Android平台在sdk2.0之前支持OpenGL ES 1.1，而在2.0以后的版本则支持OpenGL ES 2.0。

OpenGL与OpenGL ES区别

之所以会推出OpenGL-ES版本，主要是应对嵌入式环境和应用的要求。

嵌入式设备一般工作于较恶劣的环境，包括：温度、湿度、振动、冲击、酸碱腐蚀等。例如：中国的酸雨气候就给很多室外电子设备带来了新的难题，中东地区的风沙也使得美军必须采用更先进的非IT技术来保护他们的电子设备。

需要人机界面的嵌入式应用，由于受环境受环境因素的影响，一般不能提供有缘电源，在有限的电能限制下工作，如何以更低的功耗完成人机交互界面，成为OpenGL必须要面对的问题，进而推出了OpenGL-ES标准。应该说在高效完成2D/3D界面的同时，达到了降低功耗的效果。

OpenGL与OpenGL ES区别

特别说明，在OpenGL发展到1.3版本时，OpenGL API不再采用纯软件的形势进行运算，开始与硬件图形芯片结合，出现了OpenGL硬加速的实现形式。例如：很多显示芯片厂商开始推出支持OpenGL硬加速的芯片，并与软件公司合作，实现 OpenGL硬加速。很多PC机的游戏会有加速软件，如实况足球，但目前这类基于PC加速软件还是通过软件形式进行优化，即优化了 3D渲染引擎。OpenGL硬加速的优点在于，使CPU从繁重的图形运算工作中解脱出来，将运算重点集中于非界面应用，即嵌入式操作系统中优先级较高，但与界面无关的应用。 GPU（图形处理器）与CPU（中央处理器）的分工合作，带来的就是高效率。

OpenGL与OpenGL ES区别

当然，事物均具有两面性， OpenGL-ES硬加速也有缺点，即增加了设备成本。这方面主要取决于其应用是否需要强劲的性能，即对人机交互界面的更高性能的追求。

OpenGL ES相对OpenGL删减了一切低效能的操作方式，有高性能的决不留低效能的：

- 没有double型数据类型，但加入了高性能的定点小数数据类型；
- 没有glBegin/glEnd/glVertex，只能用glDrawArrays等。
- 没有实时将非压缩图片数据转成压缩贴图的功能，程序必须直接提供压缩好的贴图；

OpenGL ES 1.x和OpenGL ES 2.x

OpenGL ES 1.x 为固定渲染管线（Fixed_Function）而设计。子版本包括：1.0，1.1。1.0从OpenGL 1.3裁减而来；1.1从OpenGL_1.5裁减而来。1.1向下兼容1.0。经研究，1.1因为更先进，而且相比1.0增加的特性也都很有用，所以基本上不用考虑1.0了。1.1和1.0的变化不算很大

OpenGL ES 2.x 为可编程渲染管线（Programmable）而设计。目前只有2.0这一个子版本，从OpenGL2.0裁减而来。和1.x的区别是可以支持vertex和pixel shader，因此能够实现更多的特效。另外2.0就不再支持1.x里面的固定管线功能了，也就是说2.x并不向下兼容1.x。

Android OpenGL ES开发框架

OpenGL ES开发包

android.opengl.*;

OpenGL ES视图

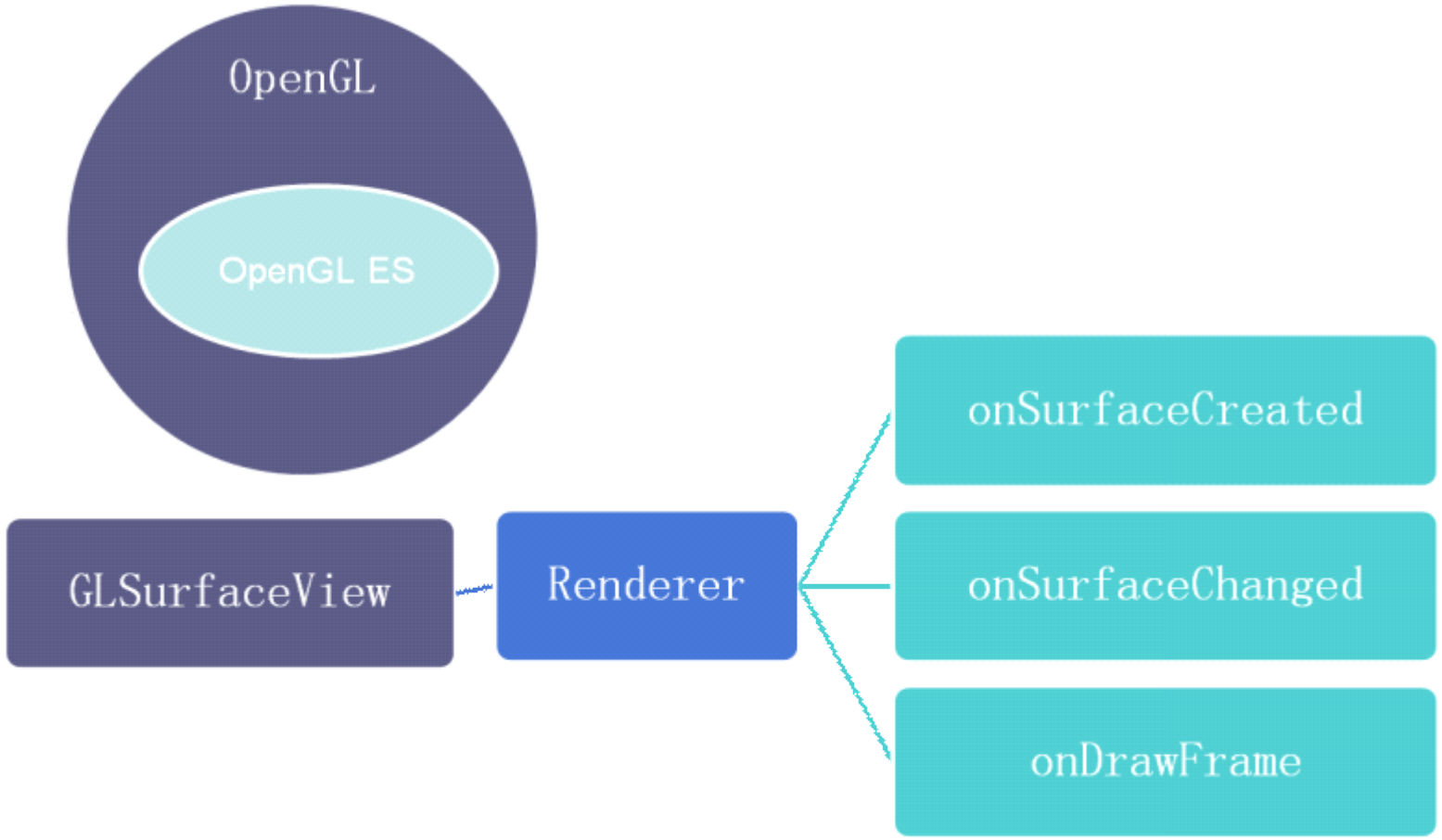
- GLSurfaceView　//OpenGL视图
- Renderer　　//渲染器

Android OpenGL ES开发框架

Renderer接口

- onSurfaceCreated(GL10 gl, EGLConfig config)
- onSurfaceChanged(GL10 gl, int width, int height)
- onDrawFrame(GL10 gl)

小结


OpenGL
OpenGL ES
GLSurfaceView
Renderer
onSurfaceCreated
onSurfaceChanged
onDrawFrame

作业

思考？

- 如何在OpenGL ES开发框架上绘制图形？并为图形添加颜色和进行变换操作？

课程目标

- **通过OpenGL来完成2D多边形的绘制。**

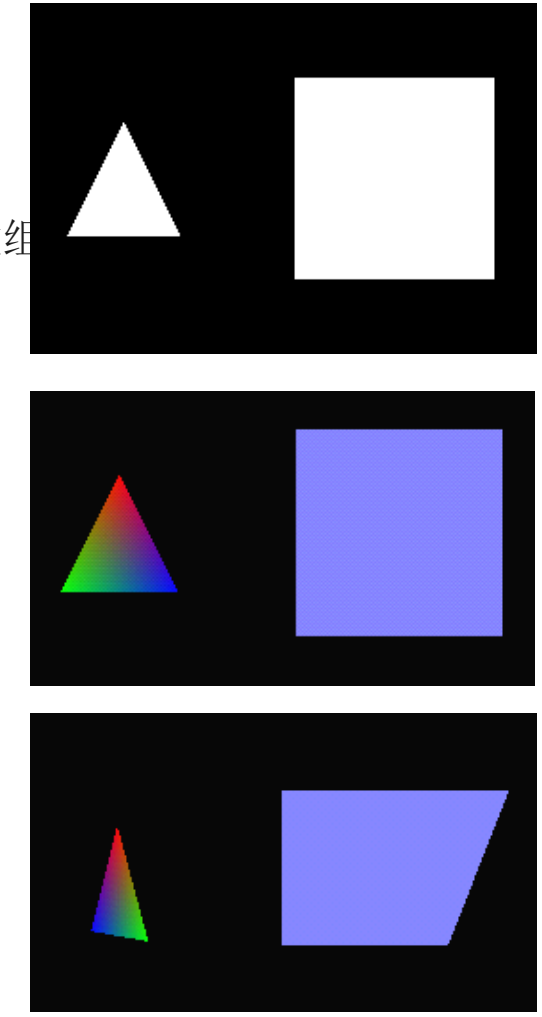
  坐标系、点、线、三角形、四边形、顶点数组

- **为多边形添加颜色。**

  颜色数组、着色模式

- **变换操作。**

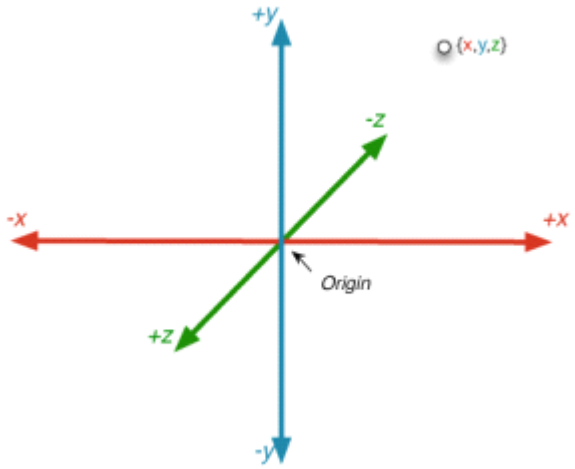
  平移、旋转、缩放



2D多边形

坐标系、点、顶点

3D图像的最小单位称为 点（point） 或者 顶点(vertex)。它们代表三维空间中的一个点并用来建造更复杂的物体。多边形就是由点构成，而物体是由多个多边形组成。尽管通常OpenGL支持多种多边形，但OpenGL ES只支持三边形（即三角形）。



2D多边形

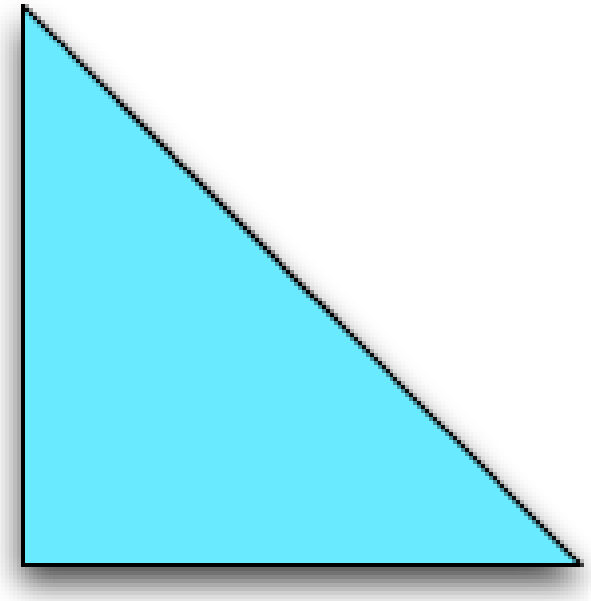
三角形

由不在同一直线上的三条线段首尾顺次连接所组成的封闭图形叫做*三角形*。

（0，1，0）//上顶点

（0，0，0）//直角处顶点

（1，0，0）//右边顶点



2D多边形

在OpenGL中绘制2D多边形常用的函数以及常量：

- glEnableClientState/glDisableClientState：状态开关
- glVertexPointer：设置顶点数据
- glDrawArrays：绘制函数
- *GL_VERTEX_ARRAY：顶点数组*
- GL_BYTE/GL_SHORT/GL_FIXED/GL_FLOAT：顶点数据的类型
- GL_LINES：线
- *GL_TRIANGLES：三角形*
- *GL_TRIANGLE_STRIP：三角形带*

颜色

OpenGL ES只支持RGBA颜色模式，即我们通过定义红，绿，蓝以及alpha元素来定义颜色， alpha值定义了颜色之后物体的透视（明）程度。

颜色数组

和顶点数组一样，由每一个顶点的颜色数据组成。

着色模式

在OpenGL ES中，我们可以为整体物体设计一个单一的颜色，称之为“单一着色”； 也可以用多种颜色混合渲染，而颜色之间过渡很平滑，称之为“平滑着色”。

颜色

常用函数及常量

- glColor4f：设置单一颜色
- glColorPointer：设置颜色数组
- *GL_COLOR_ARRAY：颜色数组（通过状态开关函数* glDisableClientState 来操作）
- *GL_COLOR_BUFFER_BIT：颜色缓存*

变换

OpenGL ES有三种不同类型的变换，它们分别是：

- 转移（Translate）：在3D空间中移动物体
- 旋转（Rotate）：绕X, Y, 或者 Z 轴进行旋转
- 缩放（Scale）：改变物体的大小

变换

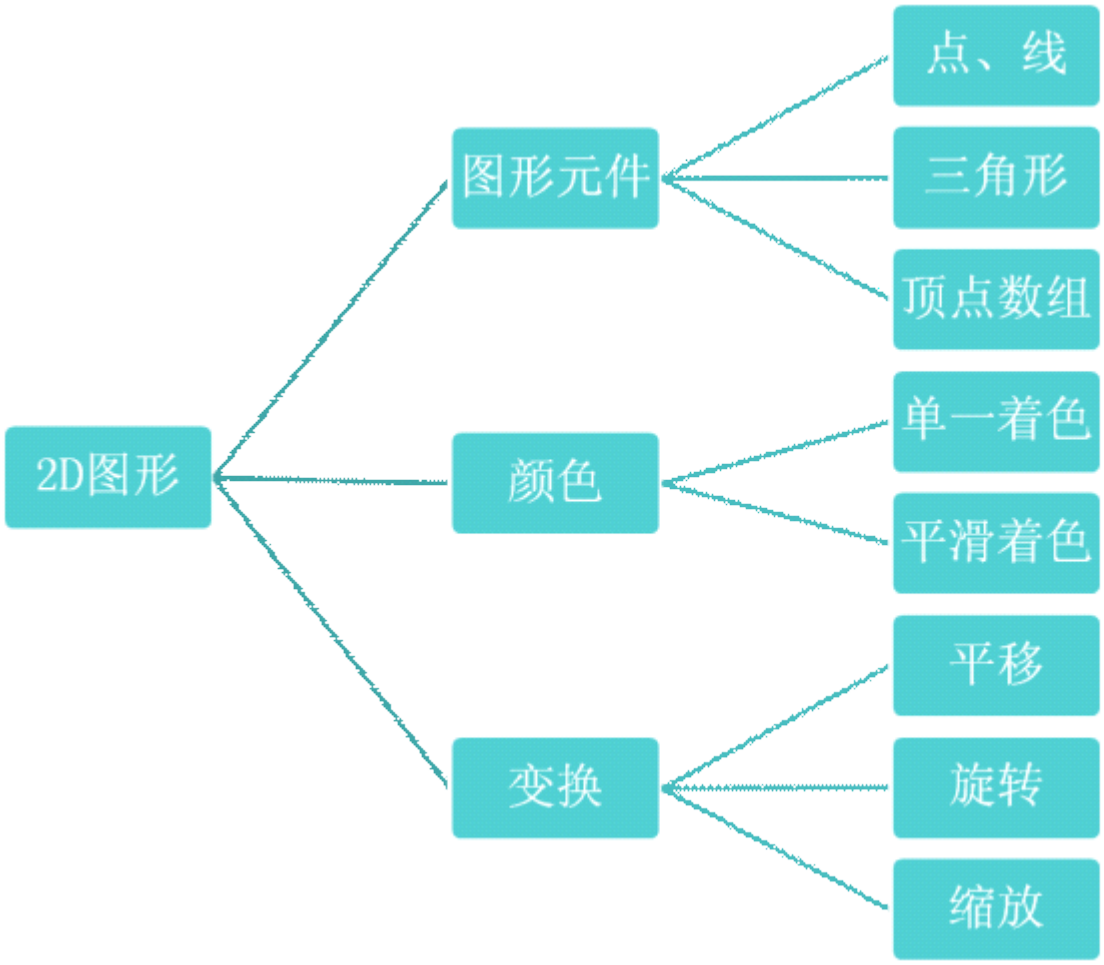
常用函数及常量

glTranslatef：平移

glRotatef：旋转

glScalef：缩放

小结



作业

思考？

- 如何绘制3D图形呢？比如：立方体。
- 在3D空间中，当多个相同物体在同一直线上时，前面的物体是否要遮挡后面的物体呢？如何才能让观察者看得更逼真呢？

课程目标

绘制函数：glDrawElements

- 索引数组
- 通过glDrawElements来绘制一系列的三角形

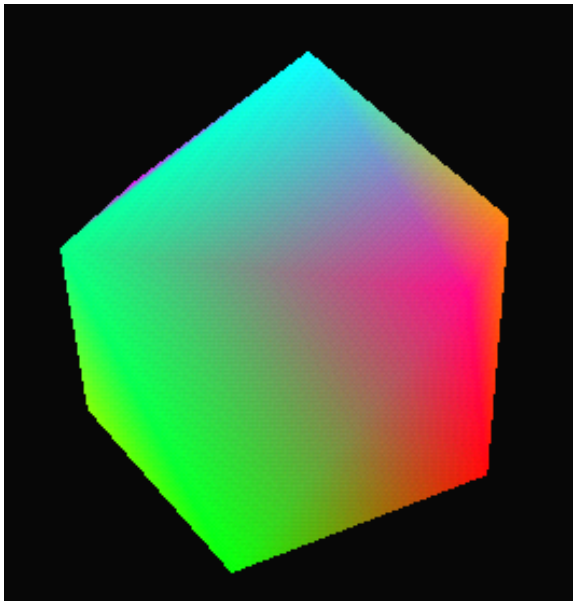
设置视口

- 透视
- 正交

3D图形绘制

索引数组

我们在绘制一个复杂的物体时，可以使用glDrawElements方法通过一组特殊的数据来绘制一系列的三角形，从而构成复杂的物体，那么这一组特殊的数据需要对应顶点数组中每一个数据的索引号，我们就把这组特殊的数据称之为“索引数组”。



3D图形的绘制

```
glDrawElements(int mode, int count,
                    int type, Buffer indices)
```

- mode:绘制图形的模式（三角形）
- count：顶点数目
- type：索引数组的数据类型
- indices：索引数组

设置视口

正交和透视

OpenGL ES中具有的两种不同的视口类型：正交和透视。为更好地理解，我们先看看铁路轨道，铁路的两条铁轨之间具有固定的距离。其固定的距离是由铁轨根据承载什么样的火车而决定。

设置视口

透视

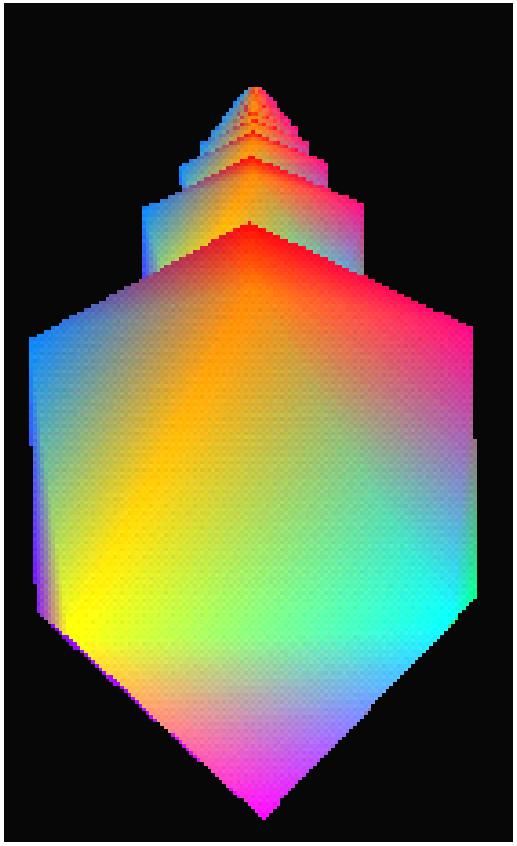
OpenGL可以设定的视口中的一种就是使用透视（perspective）。当你这样设置视口时，物体会随着移远而越来越小，视线会在物体移离观察者时最终交汇。这是对真实视觉的模拟；人们就是以这种方式观察世界的。

正交

另一种设置视口称为正交（orthogonal） 视口。这种类型的视口，视线永远不会交汇而且物体不会改变其大小。所以看上去是不真实的，通常也不是你所希望的。

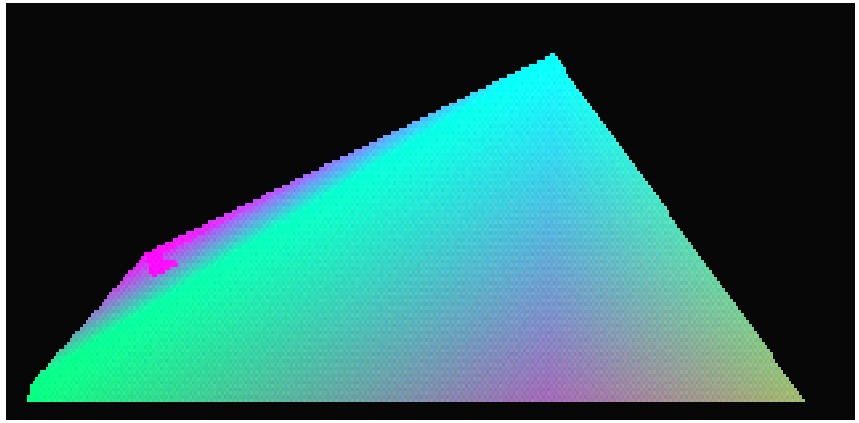
透视视口

```
glFrustumf(
        -ratio,
        ratio,
        -1,
        1,
        1.0f,
        1000.0f)
```

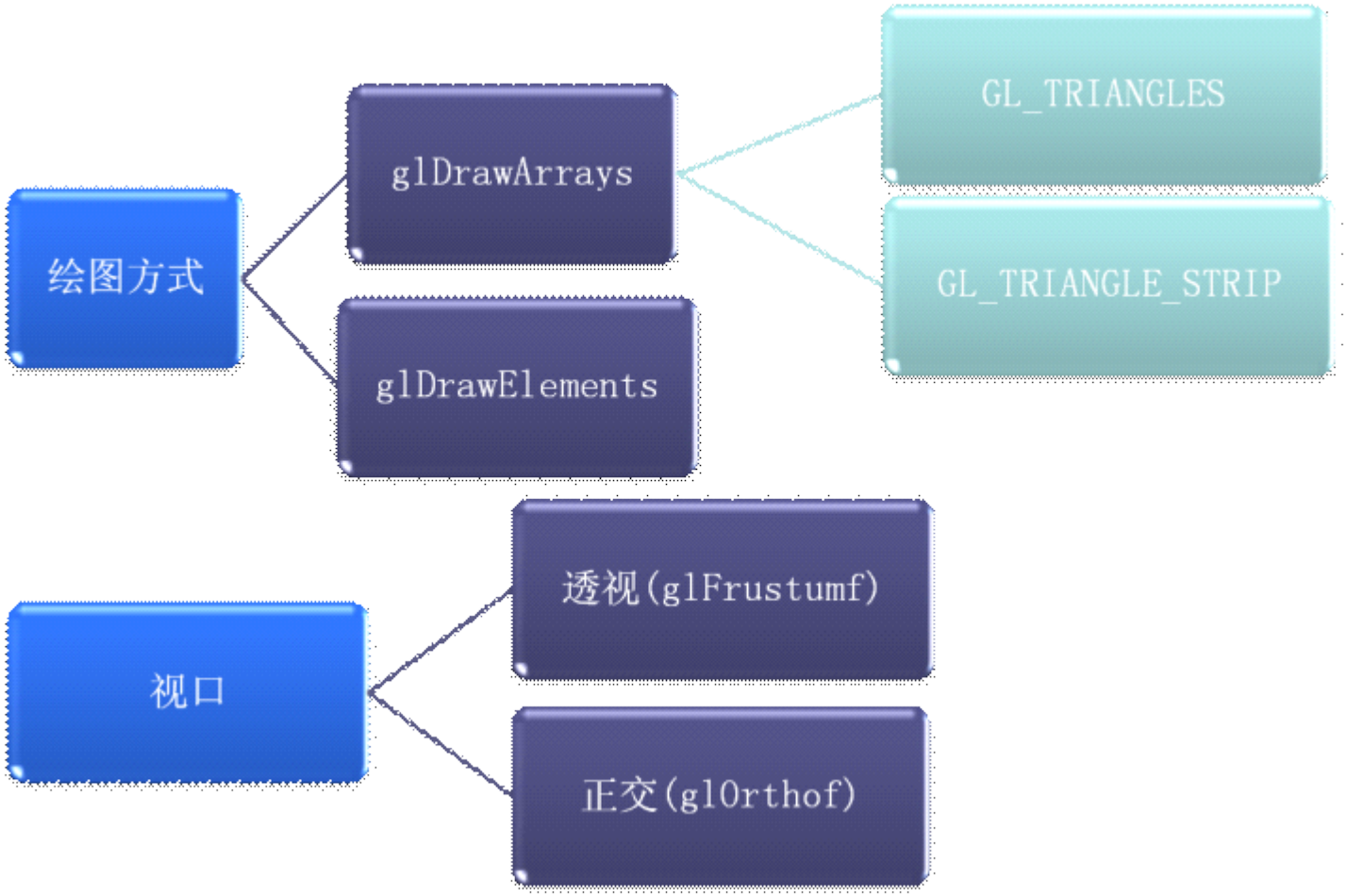


正交视口

```
glOrthof(
        -ratio,
        ratio,
        -1,
        1,
        1.0f,
        1000.0f)
```



小结


绘图方式
glDrawArrays
glDrawElements
GL_TRIANGLES
GL_TRIANGLE_STRIP
视口
透视(glFrustumf)
正交(glOrthof)

作业

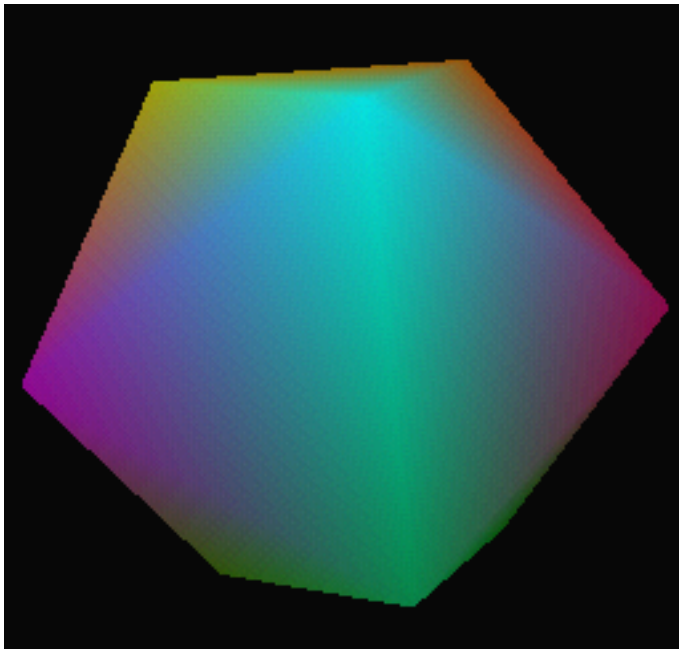
思考？

- 在OpenGL ES中如何为物体添加和设置光效。

课程目标

阴影模型

GL_FLAT（恒定）、 GL_SMOOTH（光滑）



光效三要素

- 环境元素（ambient component）
- 散射元素（diffuse component）
- 高光元素（specular component）

光源的属性

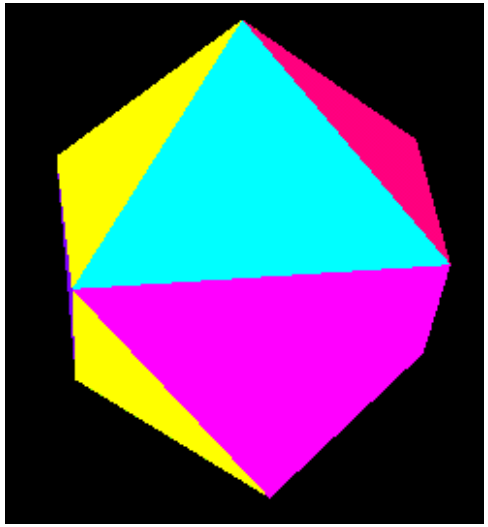
位置、方向、角度

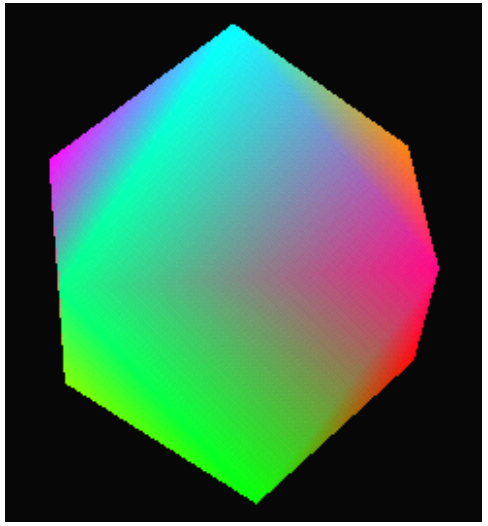
顶点法线

阴影模型

GL_FLAT将指定三角形上的每个像素都同等对待。多边形上的每个像素都具有相同的颜色，阴影等。在这种方式下，物体看上去极为不真实。

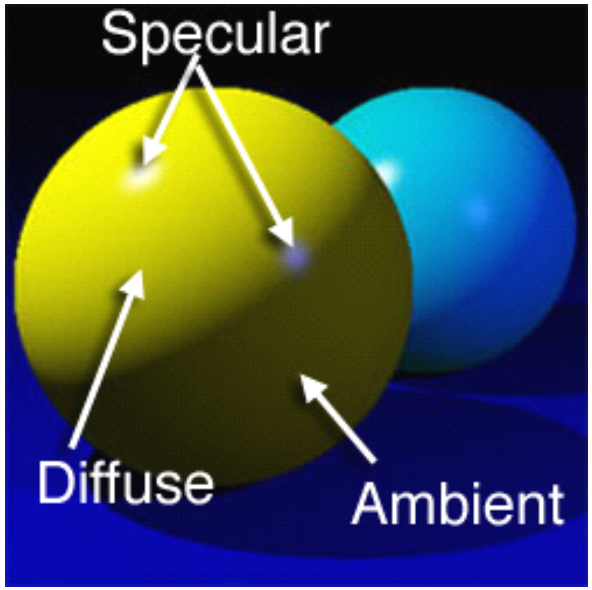
GL_SMOOTH模式，它使用了一种平滑快速的阴影算法，称为Gouraud算法。OpenGL ES默认为GL_SMOOTH模式。





光效三要素

在 OpenGL ES中，光由三个元素组成，分别是环境元素（ambient component），散射元素（diffuse component）和 高光元素（specular component）。我们使用颜色来设定光线元素。我们可以指定各光线元素的颜色和相对强度，比如：明亮的白色光定义为白色 （{1.0, 1.0, 1.0, 1.0}），而暗白色可能定义为灰色 （{0.3, 0.3, 0.3 1.0}）。 你还可以通过改变红，绿，蓝元素的百分比来调整色偏。



环境光

环境光没有明显的光源。其光线折射于许多物体，因此无法确定其来源。环境元素平均作用于场景中的所有物体的所有面。

```
//环境光的颜色
FloatBuffer lightOAmbient = FloatBuffer.wrap(new
float[]{0.1f, 0.1f, 0.1f, 1.0f});
//设置环境光
gl.glLightfv(GL10.GL_LIGHTO, GL10.GL_AMBIENT,
lightOAmbient);
```

散射光

在OpenGL ES中可以设定的第二个光线元素是 散射元素（diffuse component）。在现实世界里，散射光线是诸如穿透光纤或从一堵白墙反射的光线。散射光线是发散的，因而参数较柔和的光，一般不会像直射光一样产生光斑。散射元素定义了比较平均的定向光源，在物体面向光线的一面具有光泽。

```
FloatBuffer light0Diffuse = FloatBuffer.wrap(new
float[]{0.7f, 0.7f, 0.7f, 1.0f});
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE,
light0Diffuse);
```

高光

高光元素定义了光线直接照射并反射到观察者从而形成了物体上的“热点”或光泽。光点的大小取决于一些因素，但是如果你看到如上图黄球所示一个区域明显的光斑，那通常就是来自于一个或多个光源的高光部分。这种类型的光是十分直接的，它们会以热点和光晕的形式反射到观察者的眼中。如果你想产生聚光灯的效果，那么应该设置一个很大的高光元素值及很小的散射和环境元素值（还需要定义其他一些参数，等下会有介绍）。

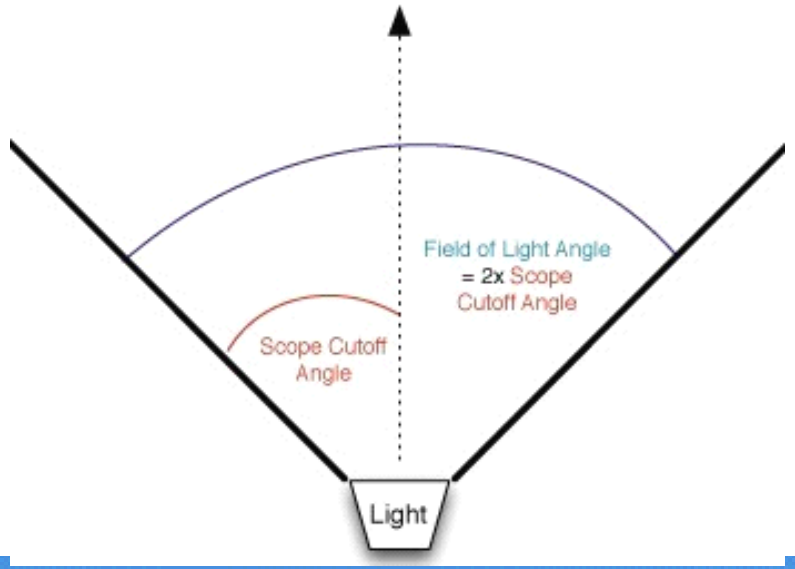
```
FloatBuffer light0Specular = FloatBuffer.wrap(new
float[]{0.7f, 0.7f, 0.7f, 1.0f});
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_SPECULAR,
light0Specular);
```

光源的属性

➢位置（GL_POSITION）：3D空间中光源的位置。

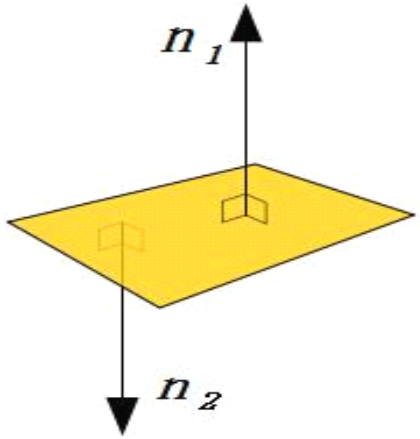
➢方向（GL_SPOT_DIRECTION）：由x,y,z向量来确定光线的方向。

➢角度（*GL_SPOT_CUTOFF*）：指定 GL_SPOT_CUTOFF 值时，它定义了中心线两边的角度，所以如果你指定截止角时，它必须小于 180度。如果你的定义为45度，那么实际上你创建了一个总角度为90度的点光源。这意味着你可设定的 GL_SPOT_CUTOFF 的最大值为 180度。



顶点法线

法线：一个垂直于指定多边形表面的向量（或直线）。



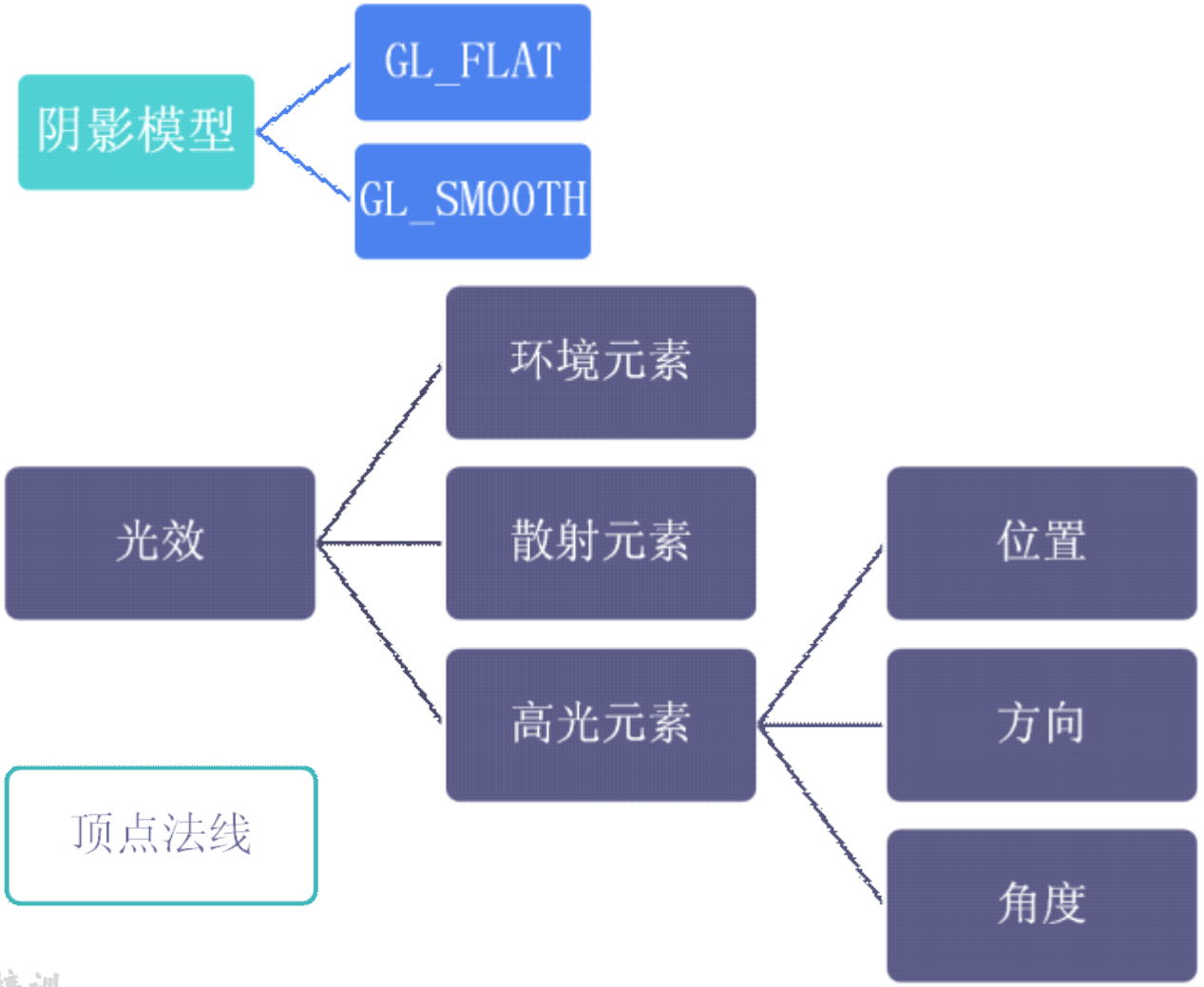
使用 GL_SMOOTH 渲染，所以OpenGL ES需要知道**顶点法线**（vertex normal） 而不是表面法线，一个复杂的物体顶点很多，所以计算量也就很大。

设置顶点法线

```
//允许设置法线数组
gl.glEnableClientState(GL10.GL_NORMAL_ARRAY);
//设置法线数组
gl.glNormalPointer(GL10.GL_FLOAT, 0, normals);

// 开启颜色材质
gl.glEnable(GL10.GL_COLOR_MATERIAL);
```

小结



作业

思考？

- 物体的表面效果实际上是由场景中的光和多边形的材质决定的，我们如何为一个3D空间中的物体设置材质呢？
- 材质又由什么样的元素构成呢？