# 实验目的

学习并掌握如何对几何对象进行**纹理映射**；了解如何在纹理映射下进行光照计算。

# 实验内容

1、课本上8.6节“一个带有纹理的旋转立方体”。

① 将代码实现完整，要求能够用鼠标或键盘控制；

② 运行程序并给出结果；

③ 详细分析程序中**如何将纹理映射到立方体**，从而形成最终效果的。

④ 在场景中加入一个光源，并进行**光照计算**，使纹理在光照下产生更逼真的效果。

# 实验步骤

## 纹理映射

### 定义纹理

如果不使用纹理对象，则直接定义纹理；如果使用纹理对象，则先创建纹理对象，再为对象定义纹理

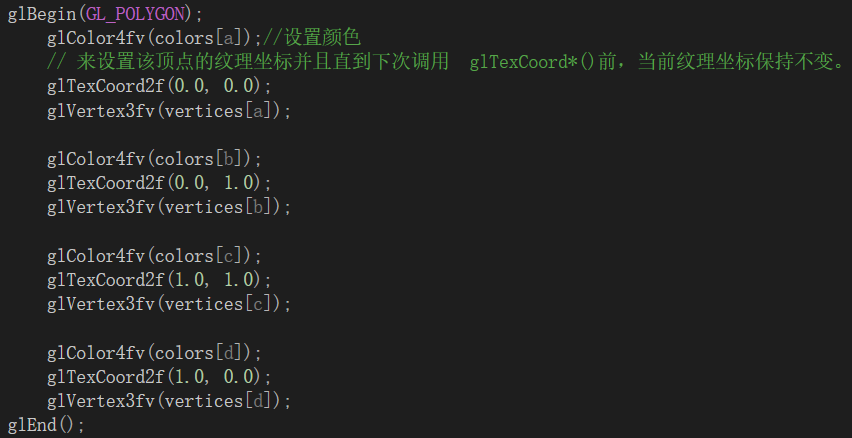
这行代码是属于那种还没搞清楚



### **指定纹理参数**

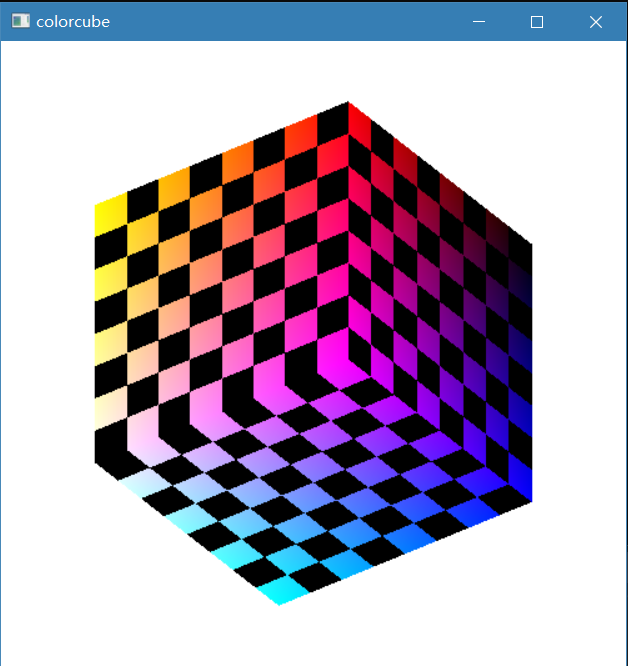


### **为顶点赋予纹理坐标**



### **效果图与一些问题**

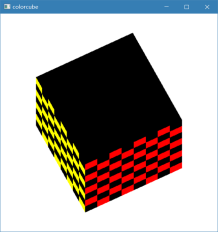
#### 效果图



#### cube颜色为什么是渐变的

#### 

渐变是因为默认的是平滑着色模式，如果改成平面着色模式，效果如下：



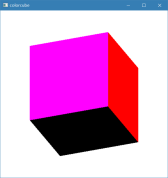
#### 把glTexCoord2f注释掉就变成全黑了

#### 

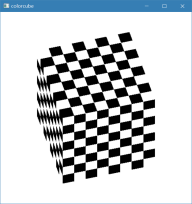
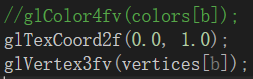
**解答：**

因为启用了纹理后，没有设置纹理图元默认是黑色的（相当于默认纹理是黑色的），使用了纹理贴图才会有颜色，把启用二维纹理注释后，cube显示出了颜色。





#### 把glColor4fv注释掉就变成黑白相间



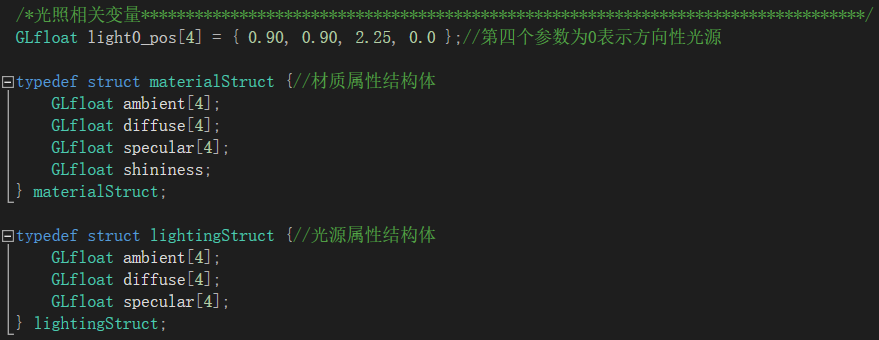
**解答：**

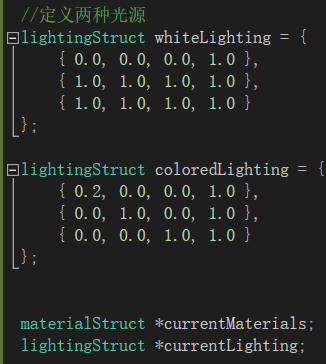
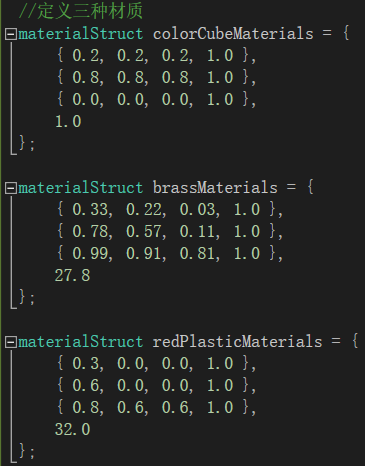
因为纹理贴图是黑色方形和透明方形相间的，默认的颜色是白色，所以使用纹理贴图后看到的是黑白相间的方形。

启用颜色后，由于纹理贴图有透明的部分，所以启用的颜色得到了保留。

## 启用光照

### 添加全局变量





### init函数中添加光照并设置光照和材质属性



### 效果图

由于启用了光照，glColor4fv所上设置的颜色都会失效，显示的是材质的颜色



# 实验心得：

## 纹理

纹理（Texture）——应用到几何图元上的图像数据， 也称纹理贴图或纹理图（Texture map）

## 纹理空间

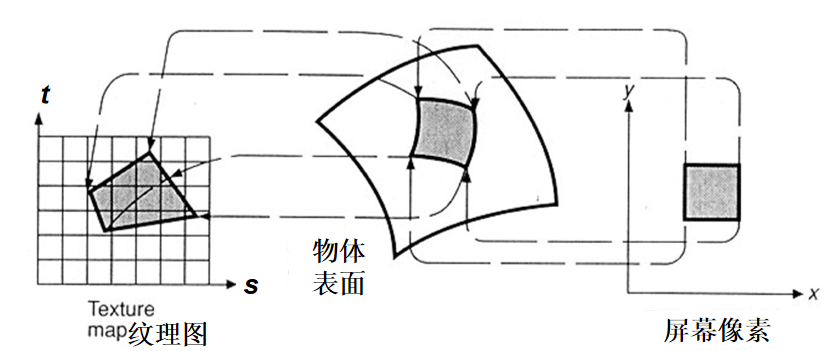
二维纹理在坐标为(s, t)的二维空间中描述，该空间称为纹理空间，该空间中的坐标称为**纹理坐标**，而此描述可以用数学函数解析表达，亦可以用各种数字化图像来离散定义。

## 纹理映射

纹理映射——将**纹理空间**中的纹理单元（texel）映射到**屏幕空间**中的像素的过程，具体包括两个映射。

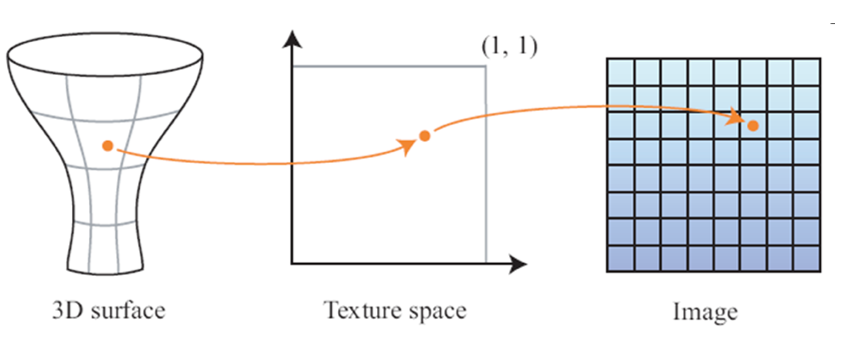
映射1：屏幕空间→物体空间（三维几何空间）

映射2：物体空间→纹理空间 (s, t) = F(x, y, z)



## 纹理映射关键

物体空间→纹理空间→图像像素空间



## 纹理映射的步骤

### 定义纹理

### 指定纹理参数

### 为顶点赋予**纹理坐标**

纹理坐标——也称**纹理空间坐标**，用于建立几何图元上的顶点与纹理图中的纹理单元间的映射关系，进而告诉OpenGL如何将纹理图贴在几何图元上。

纹理坐标内部采用齐次坐标形式，这与顶点的存储方式一致

一维纹理坐标：(s, 0, 0, 1)

二维纹理坐标：(s, t, 0, 1)

三维纹理坐标：(s, t, r, 1)

## 纹理环境

前面例子中，纹理图像表面的颜色值都直接作为被贴图表面的颜色。

OpenGL中还可以使用纹理图像**对物体表面的颜色进行调整**，而不是**直接贴图**；也可以把纹理图像的颜色与物体表面的原先颜色进行**混合**。

上述内容可以通过调用glTexEnv\*()设置纹理环境来实现。