实验目的：

1. 熟悉图形学中坐标变换的原理和方法。

2. 通过对交互图形界面的设计领略OpenGL编程的奥妙。

3. 初探真实感图形学的魅力。

实验内容：

1. 根据图形学坐标变换的原理，利用OpenGL函数对图形界面中所加载的几何对象进行交互的平移、旋转、缩放等操作。
2. 利用OpenGL定义光源及几何对象材料。
3. 加载多个几何对象，对比有无消隐处理的结果（利用OpenGL的消隐函数）
4. 采用C/C++ 、OpenGL编写程序（参考本次及前几次实验所提供的程序代码及建立Project的过程说明。）。
5. 选作：
   * 1. 对光照的颜色和位置以及几何对象的材料进行交互修改
     2. 利用OpenGL的功能给几何对象粘贴纹理
     3. 用MFC实现交互界面

注：如能正确实现以上一个或多个选作内容，将视情况获得附加分。

实现方法：

* 说明算法各函数的功能；

1. 利用OpenGL函数对图形界面中所加载的几何对象进行交互的平移、旋转、缩放等操作：

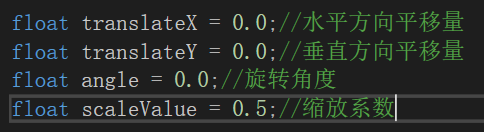
按下“w”，“a”，“s”，“d”键实现对于茶壶的上左下右的移动；

按下“q”，“e”实现了对于茶壶逆时针，顺时针的旋转

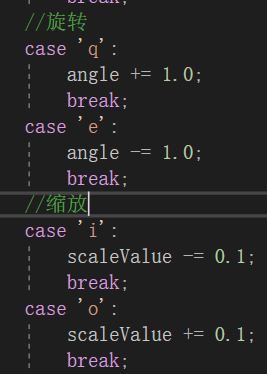
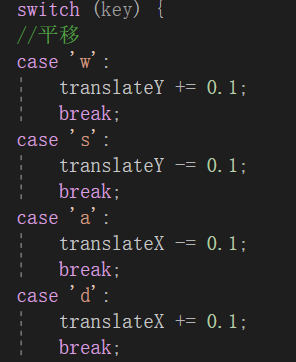
按下“o”键实现了茶壶的放大，按下“i”实现了茶壶的缩小

**具体的函数实现的功能如下：**

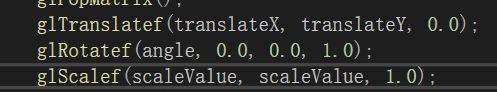
首先定义全局变量，定义平移，旋转，放大和缩小所对应的变量。



然后在keyboard函数中定义相关的按键对于相关的系数进行修改，从而实现对于物体的改变。



在display()函数中定义绘制图形的平移量，旋转量，缩放量。



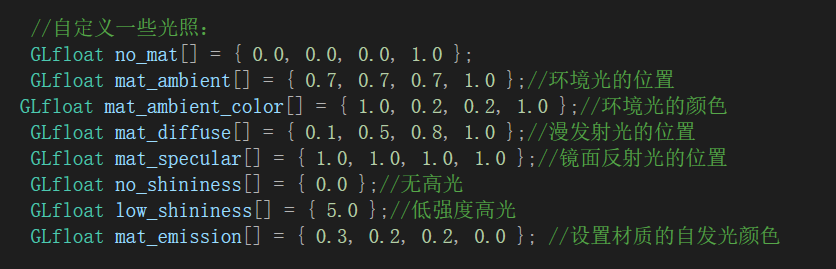
1. 利用OpenGL定义光源及几何对象材料。

在OpenGL中，光照处理系统涉及三个主要组成部分：光源、材质和光照模型。这些组件具有广泛的属性，然而它们的配置仅涉及到少数函数调用。光源属性通过glLight函数进行设置，而材质属性通过glMaterial函数进行定义，光照模型的属性则通过glLightModel\*函数进行规范。

在这些属性中，GL\_AMBIENT、GL\_DIFFUSE和GL\_SPECULAR是光源和材质通用的关键性属性。当光线从光源照射到材质表面时，GL\_DIFFUSE属性的两个分量共同决定了最终的漫反射强度，而GL\_SPECULAR属性的两个分量则共同影响最终的镜面反射强度。

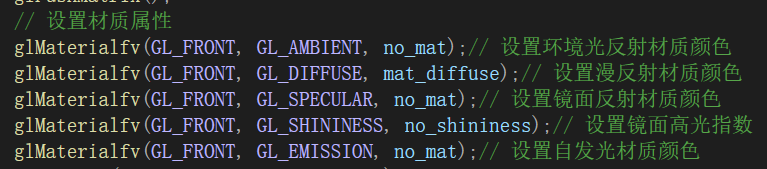
在物体描述中，可通过这三个分量定义材质颜色（Material Color），分别对应于环境光照（Ambient Lighting）、漫反射光照（Diffuse Lighting）和镜面光照（Specular Lighting）。通过为每个分量指定适当的颜色，实现对物体颜色输出的精细控制。这种方式提供了一种学术上严谨的框架，以便深入理解光照在OpenGL中的实现。

我在display函数中自定义了一些光照，使其可以对不同的物体服务，产生更好的视觉效果。

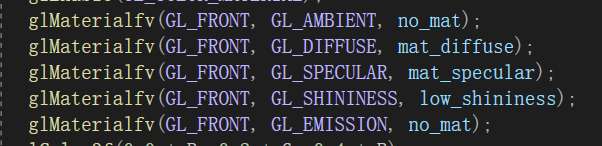


然后依次绘制物品的过程中生成不一样的光线，实现不一样的效果。

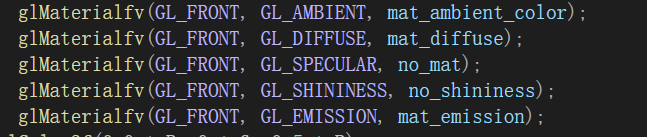
在绘制圆环这个选项时生成漫反射光而无环境光和镜面光



在绘制茶壶这个选项时有漫反射光和镜面光，并有低高光，而无环境光



在绘制十二面体这个选项的时候有漫反射光和有颜色的环境光以及辐射光，而无镜面光

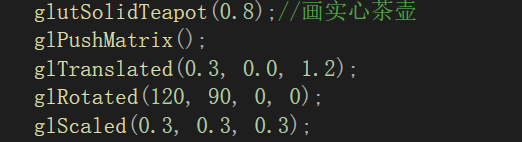


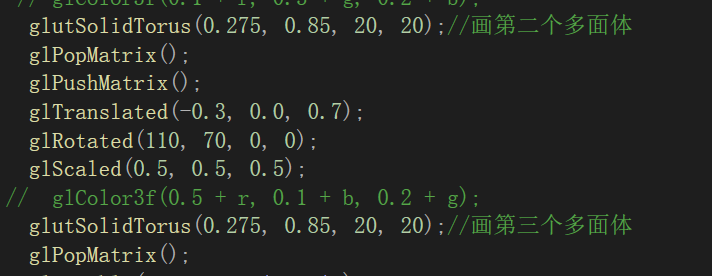
1. 加载多个几何对象，对比有无消隐处理的结果（利用OpenGL的消隐函数）

我在右键生成的菜单中“MutiObject”选项中实现了上述功能：

在display的MutiObject情况中我定义了三个物体：

茶壶，以及两个多面体





**实现的结果见实验结果部分**

**选做部分：**

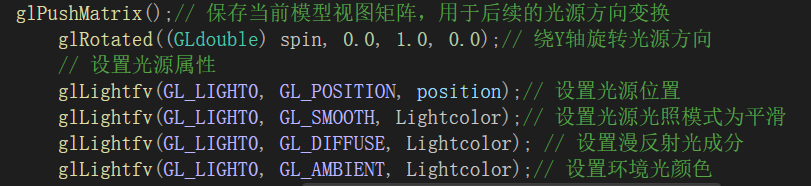
1. 对光照的颜色和位置以及几何对象的材料进行交互修改

**光照的修改：**

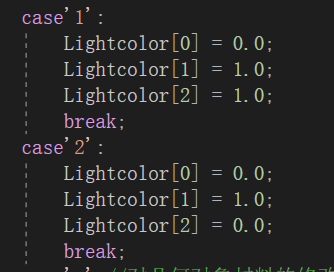
光照位置的修改可以按下鼠标左键从左向右拖动和从右往左拖动实现。

而修改光照的颜色是按下‘1’和‘2’键实现两种不同颜色的灯光

具体代码如下：



设置光源属性，使其按照lightcolor存储的值进行修改

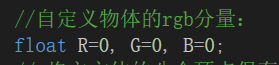


在keyboard函数中，我实现了按下1和2，修改灯光颜色rgb的参数，使其变成不同的颜色

**几何对象材料：**

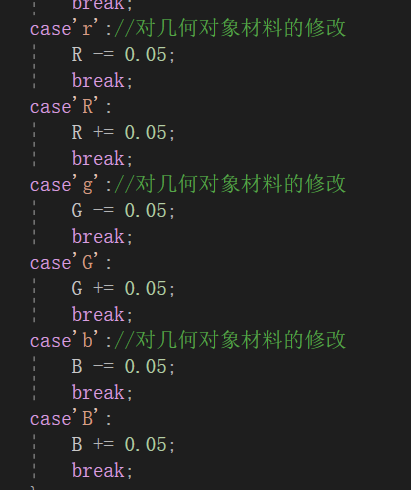
几何材料的交互我利用了修改每个物体的rgb三个分量值对于物体的材料进行修改。

首先定义全局变量R,G,B，分别为每个物体颜色RGB三个变量



然后在keyboard中对于其值进行修改：

输入大写的RGB，为增加其对应的分量值，而输入小写的rgb，则是减小去对应的分量值

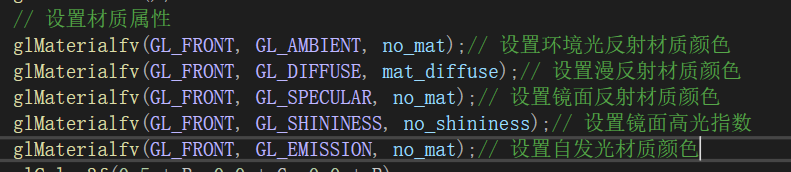


在绘制每一个图像之前，我都会利用glColor3f函数对于其物体的颜色进行定义



绘制茶壶修改其颜色的代码

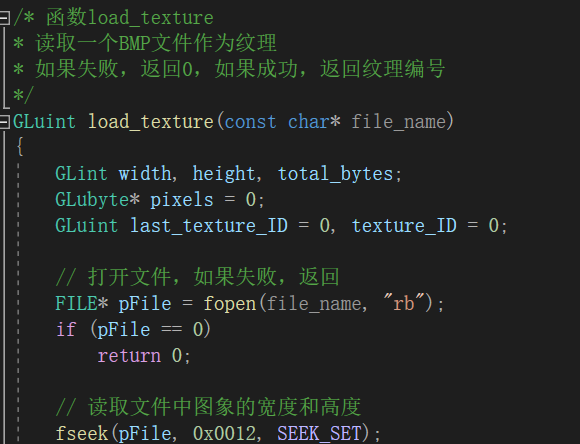
同时我为了更好的视觉呈现也利用glMaterialfv函数针对不同的物体修改其产值参数，修改的代码如下：



1. 利用OpenGL的功能给几何对象粘贴纹理

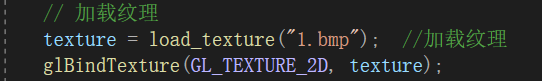
我在右键新创建的Textture选项中创建了一个正方体，将bmp的纹理贴图贴在了正方体的六个面上，为了方便展示，我还制作了一个旋转的函数，实现了这个正方体旋转展示的效果

首先，我自定义一个load\_texture函数加载bmp图片作为作为纹理。

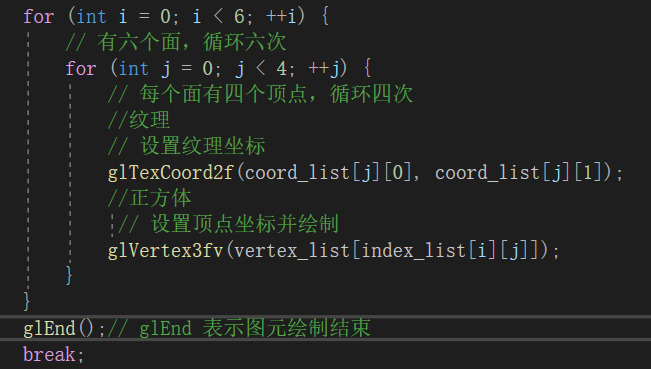


在display函数中定义变量texture读取加载的纹理。

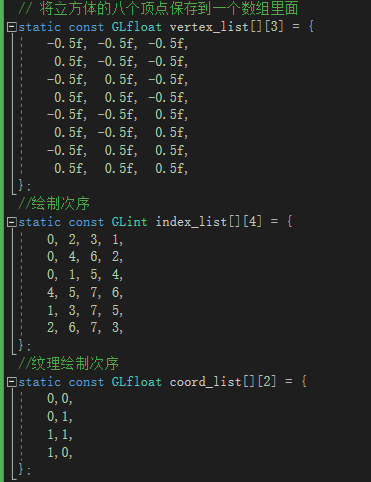




接下来为了让这个贴图可以贴在正方体的六个面上，我利用之前就定义好的纹理坐标和顶点坐标依次绘制，将贴图贴在正方体的六个面上



自定义的数据如下图所示



* 提供程序源代码并进行必要的注释：

这一部分内容在之前的内容中都有所提及。现在就整体程序中的每个函数做详细的功能阐述：

1. **Timer(int id)：**

功能：一个计时器回调函数，递增 **textangle** 变量，该变量用于控制带纹理的立方体的旋转。触发重新绘制以更新屏幕，并在一定时间间隔后再次调用自身。

1. **power\_of\_two(int n)：**

功能：检查给定的整数是否为2的幂。

1. **GLuint load\_texture(const char file\_name)：**

功能：读取 BMP 文件并将其加载为纹理。如果成功，返回纹理 ID；否则返回 0。

1. **void menu\_select(int item)：**

功能：处理菜单选择事件。如果选择的项是“Quit”，则退出程序；否则将 **obj** 变量设置为所选项，用于渲染不同的 3D 对象。

1. **void movelight(int button, int state, int x, int y)：**

功能：处理鼠标事件以移动光源。当按下左鼠标按钮时，记录初始位置。

1. **void motion(int x, int y)：**

功能：处理鼠标移动事件。计算 x 坐标的变化并更新 **spin** 变量，从而控制光源的旋转。

1. **void myinit(void)：**

功能：初始化 OpenGL 设置，包括启用光照、设置光源，并启用纹理。

1. **void display(void)：**

功能：根据当前的 **obj** 变量渲染 3D 对象。包括平移、旋转和缩放的变换。还处理了材质属性和光照条件。

1. **void myReshape(int w, int h)：**

功能：处理窗口调整大小事件，通过更新视口和投影矩阵来调整。

1. **void keyboard(unsigned char key, int x, int y)：**

功能：处理键盘事件，用于平移、旋转和缩放 3D 对象。还允许更改光源颜色和修改几何体的颜色。

1. **int main(int argc, char argv)：**

功能：初始化 GLUT，设置显示模式和窗口大小，创建窗口，初始化 OpenGL 设置，并为各种事件（鼠标、键盘、显示、调整大小、计时器）设置回调函数。然后进入主循环使用 **glutMainLoop()**。

* 实验中的创新点：

1. 为了更好的体现纹理贴图的效果，采用了计时器实时渲染，使立方体不断旋转观察到各个面。



旋转参数的改变写在glutTimerFunc()调用的Timer函数中，如果要写多个，需要多次声明这两个函数，通过glutTimerFunc();最后一个参数来区别是哪个定时器

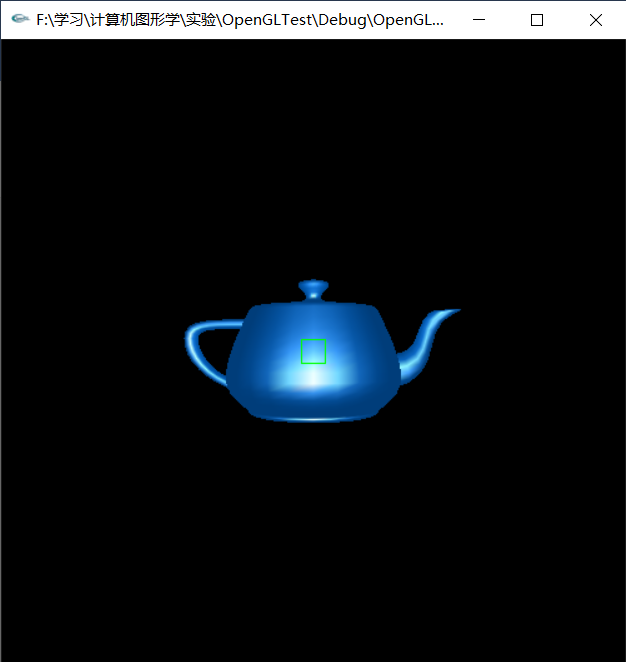


旋转参数需要对360取余以保证不会产生错误的值。

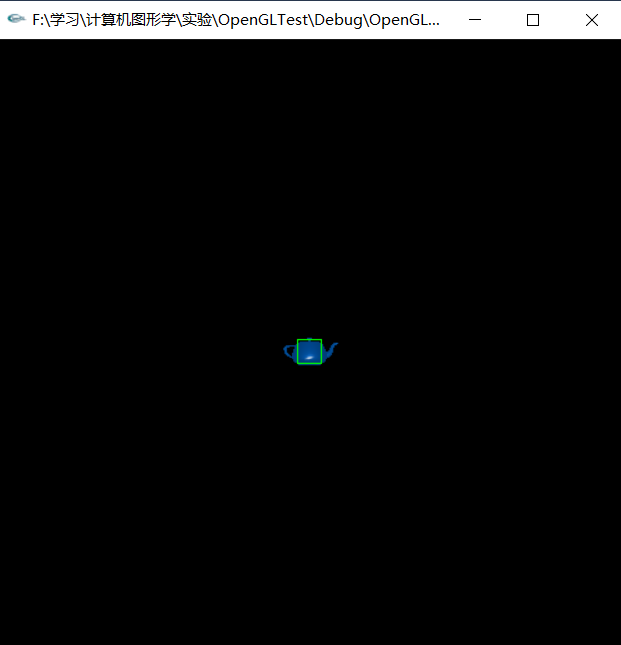
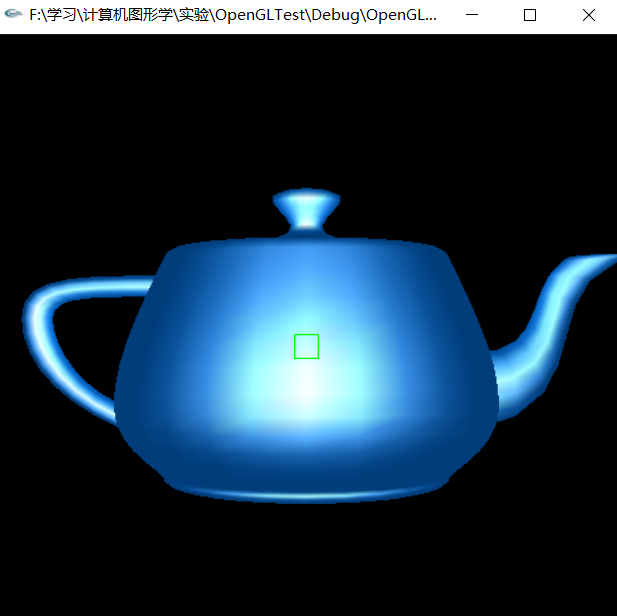
1. 自定义键盘输入交互，利用keyboard函数实现对于任意的物体可以进行平移旋转缩放的交互，具体实现见上文。
2. 自定义灯光模式，使得按下数字1键和2键实现不同灯光模式照射物体，实现不同的效果，具体实现见上文。

实验结果：

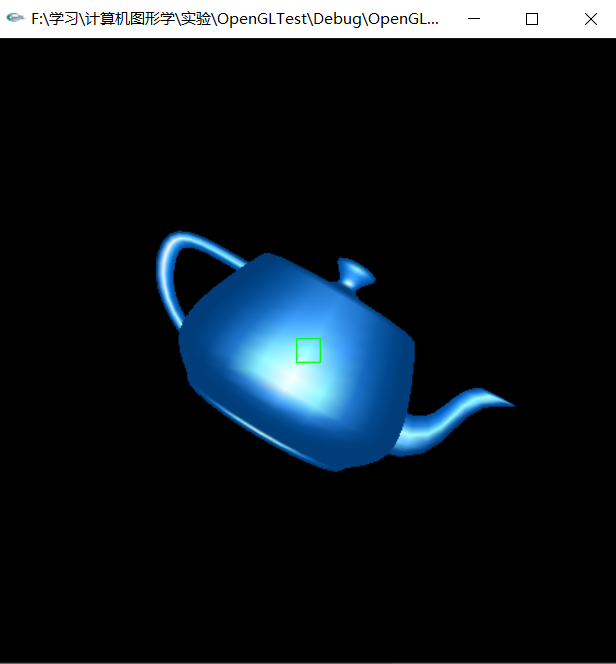
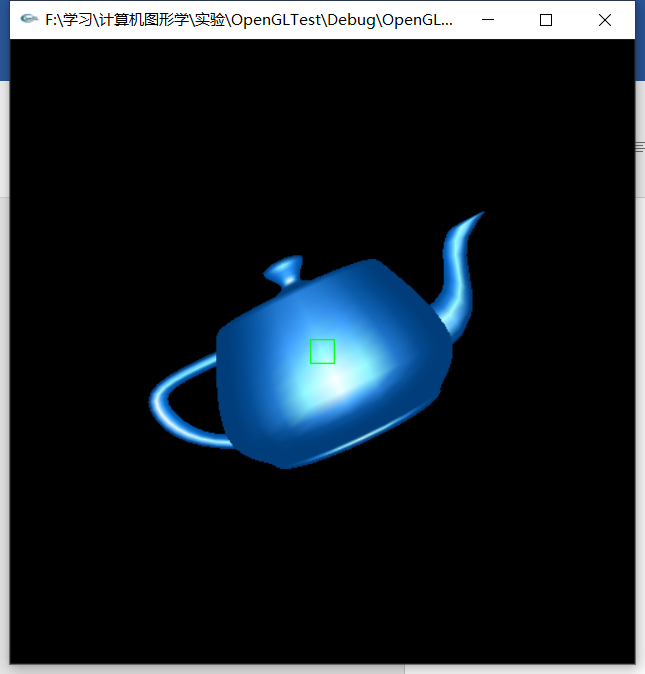
1. 对几何对象进行交互的平移、旋转、缩放等操作：



原始图像

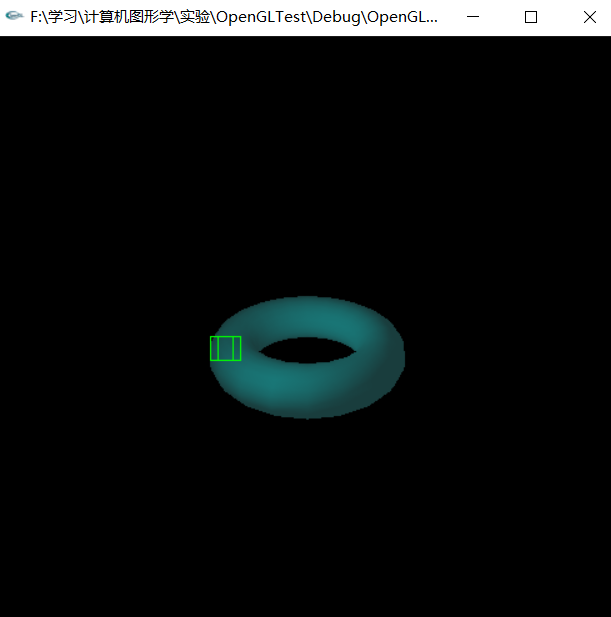
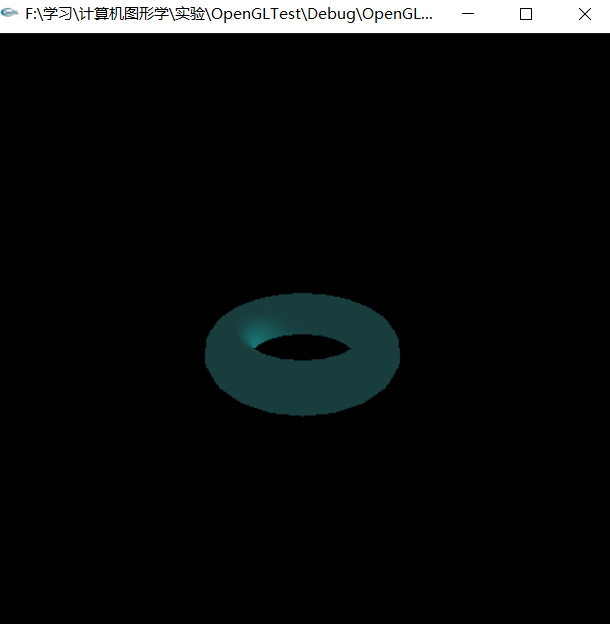
 

按下i键和o键实现了茶壶的放大缩小和放大

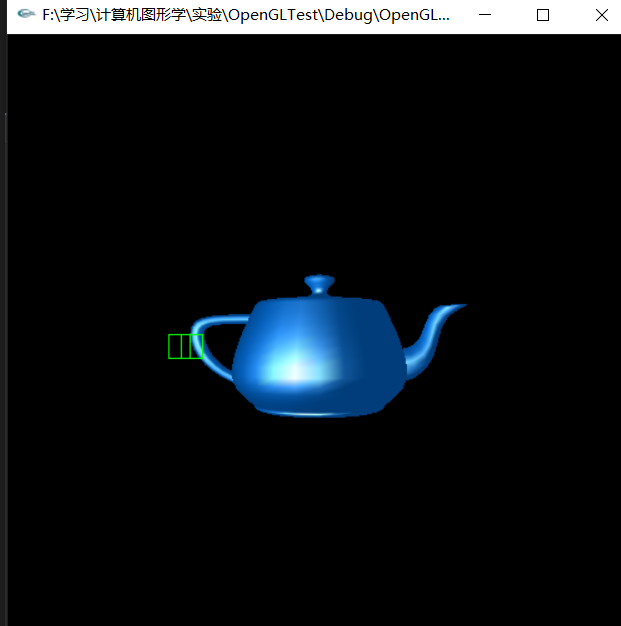


按下q，e键实现了对于茶壶逆时针顺时针的旋转

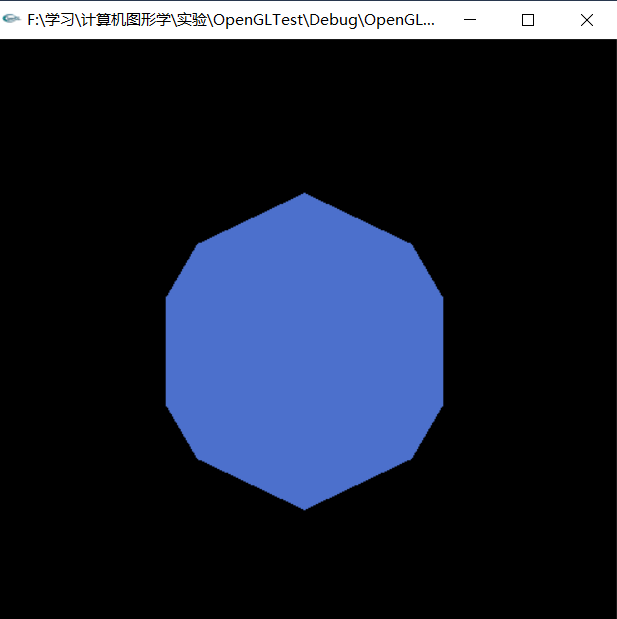
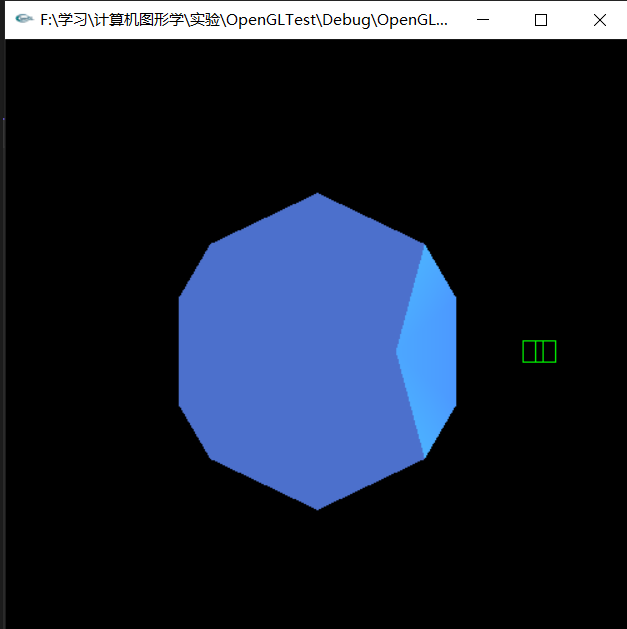
1. 自定义的光照实验结果：

自定义仅有漫反射光而无环境光和镜面，光在背面和光在正面光

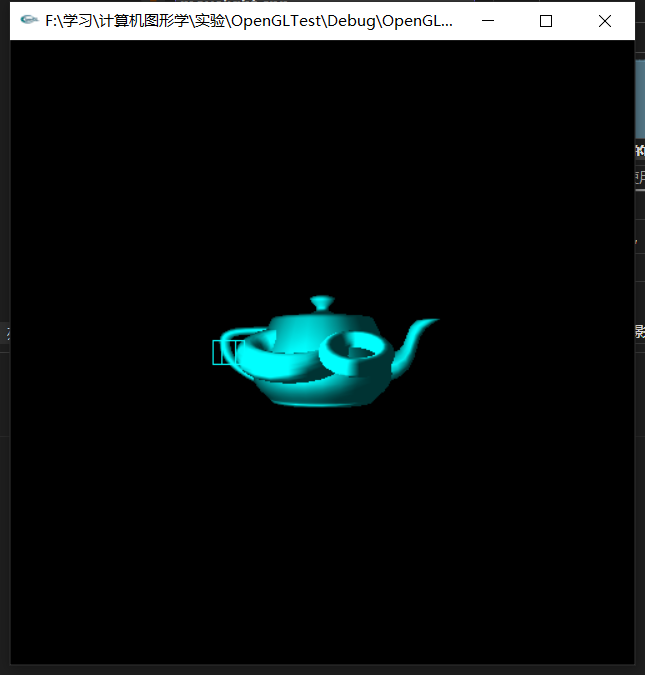
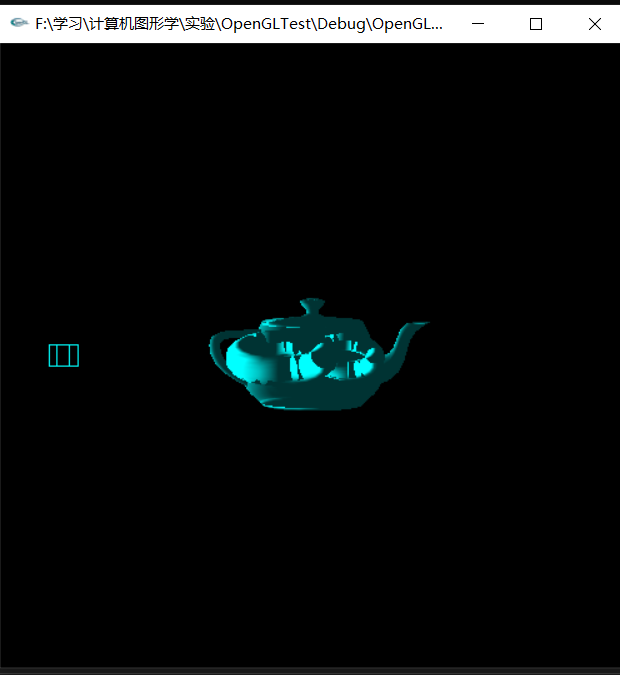
 

自定义有漫反射光和镜面光，并有低高光，而无环境光，光在背面和光在正面光



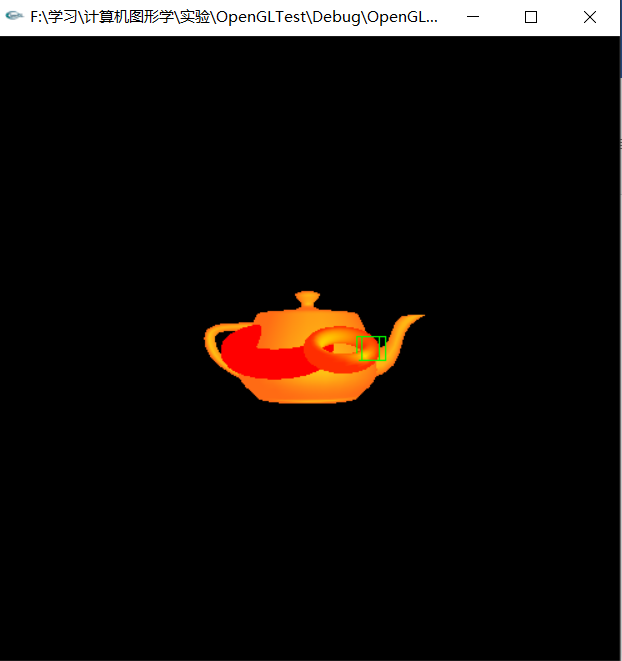
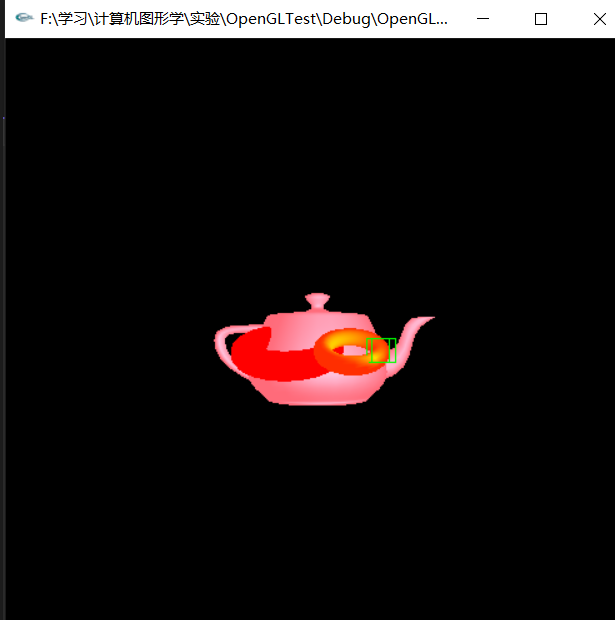
自定义有漫反射光和有颜色的环境光以及辐射光，而无镜面光

1. 加载多个几何对象，对比有无消隐处理的结果：

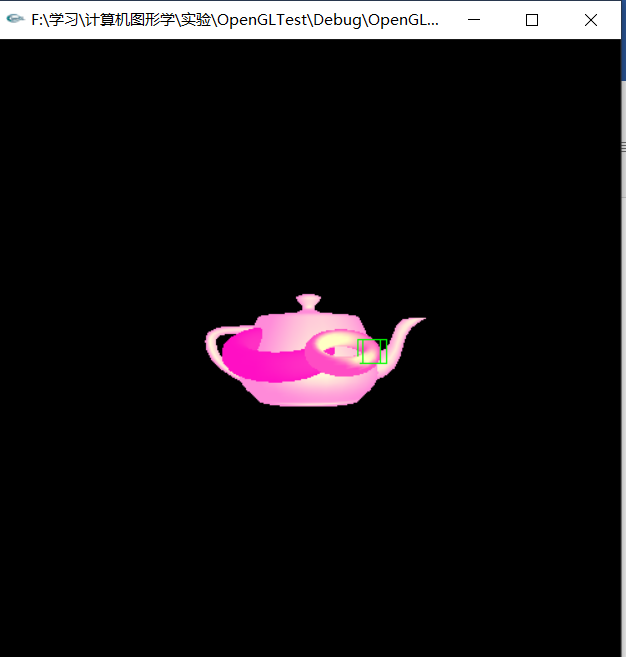
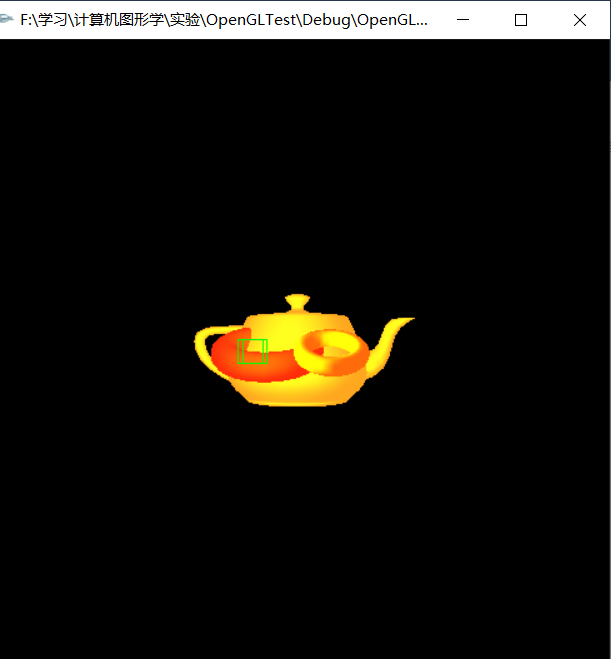
 

有消隐和无消隐的对比

1. 对光照的颜色和位置以及几何对象的材料进行交互修改：

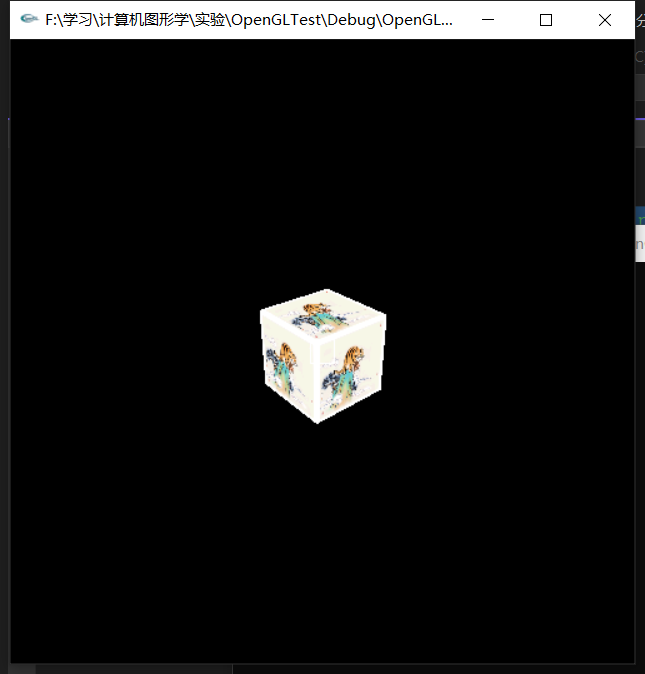
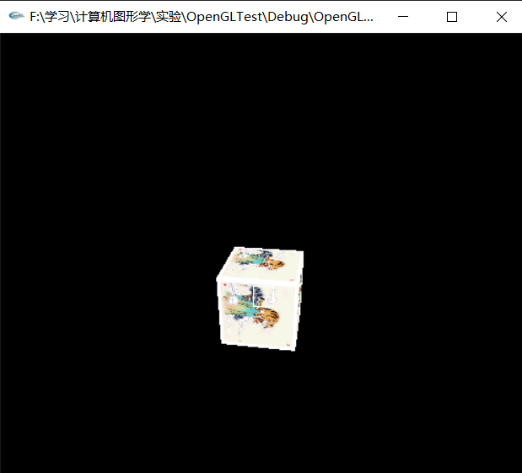


按下1和2键之后打下不同的灯光，效果如图所示



基于上述的物体，分别修改了G分量和B分量之后物品的颜色变化

1. 利用OpenGL的功能给几何对象粘贴纹理：



为正方体贴上贴图之后的效果，并且为了更加方便观察正方体的可以实现旋转操作

结论分析：

熟悉了图形学中坐标变换的原理和方法，通过对交互图形界面的设计领略到了OpenGL编程的奥妙。知道了消隐可以更好地表示对象及对象间的关系，使其表现效果更好。光照对图形真实感来说是很重要的，利用OpenGL的函数对灯光和材质的不断修改，我更加体会到了其中蕴含的原理。体会到了图形真实感绘制的难处，通过自定义数据也更好地体会到了纹理贴图的实现逻辑。

遇到的问题和解决的思路：

1-当放大物体时物体变暗

对物体的放大缩小改变了亮度值，很难看出它的大小变化，是因为对Z轴也进行了缩放，离光源的位置发生了改变，导致渲染出的结果发生变化，我们可以修改glScalef(1.1, 1.1, 1.1)为glScalef(1.1, 1.1, 1.0)，可以更好地看出缩放。

2-无法改变物体颜色：

在一开始我写代码的时候，没有注意代码函数中逻辑的执行顺序，先绘制的物品，然后定义其颜色，这样写代码之后，颜色的定义会给下一个物体而不是这个物体，我没有理解清楚其函数实现了逻辑。