专业：计算机科学与技术
姓名：余启航
学号：3190103324
日期：2021.12.13
地点：宿舍

浙江大学实验报告

课程名称：计算机图形学　指导老师：童若锋　成绩：
实验名称：OpenGL 纹理　实验类型：基础实验　同组学生姓名：

装

订

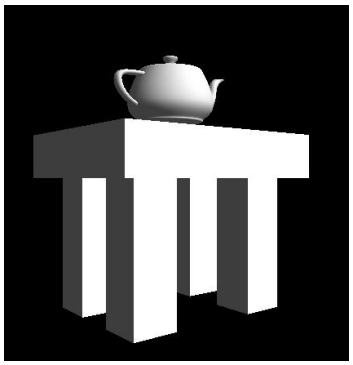
线

一、实验目的和要求

在 OpenGL 消隐和光照实验的基础上，通过实现实验内容，掌握 OpenGL 中纹理的使用，并验证课程中关于纹理的内容。

二、实验内容和原理

使用 Visual Studio C++编译已有项目工程。



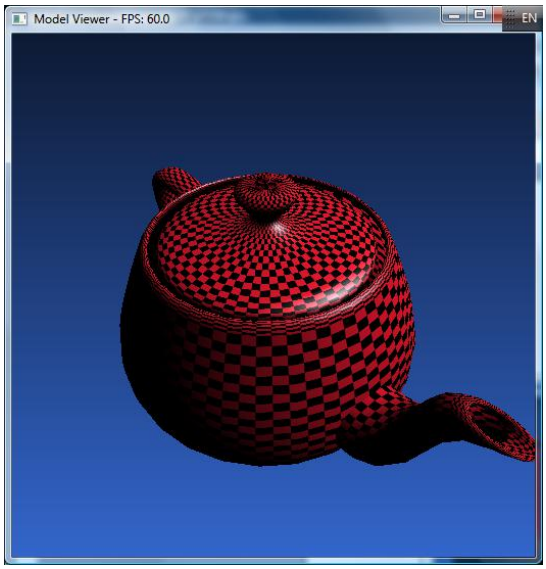
模型尺寸不做具体要求。要求修改代码达到以下要求：

1. 通过设置纹理，使得茶壶纹理为：

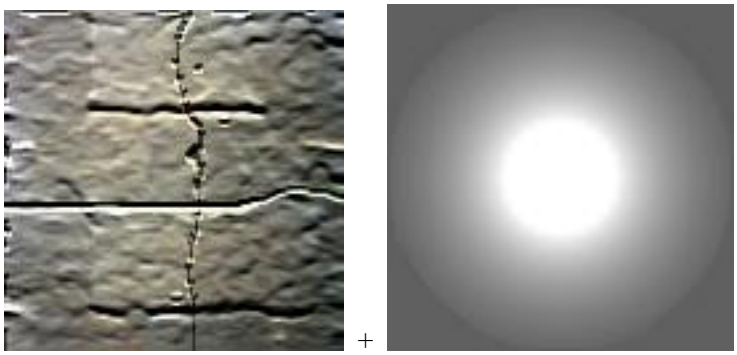
2. 使得桌子纹理为：

3. 对茶壶实现纹理和光照效果的混合

4. 自己用代码产生一张纹理，并贴在茶壶表面，效果类似：



5. 在桌面上实现两张纹理的叠合效果(附加项，完成可加分)：



提示：glSolidCube()并不会为多边形指定纹理坐标，因此需要自己重写一个有纹理坐标的方块函数。另外，此实验需要用到 #include <windows.h>

三、主要仪器设备

Visual Studio C++
glut.zip
模板工程

四、操作方法和实验步骤

1．茶壶纹理

纹理来源于 bmp 文件，因此先了解 bmp 文件的存储方式。一般来说，bmp 文件由 FileHeader, InfoHeader, PixelData 三部分组成，本实验需要用到后两部分。另外，图像存储数据的顺序是 BGR，因此读入之后需要进行转换，否则得不到希望的图像。另外，图像 上下和左右也是反的，但是这里不影响效果，所以没有进行转换。借助参考文档给出的函数进行图像读入之后，就可以开始进行纹理创建了。首先需要创建纹理标识符，类似于实验四中显示列表的 lid：

```
unsigned int texture[5];
glGenTextures(5, texture);// 第一参数是需要标示符的个数，第二参数返回标示符数组
```

之后进行纹理的绑定和纹理过滤，在纹理过滤时需要指定两种过滤方式，应对纹理像素比实际像素大或者小的情况。

```
glBindTexture(GL_TEXTURE_2D, texture[i]);
// 指定当前纹理的放大/缩小过滤方式
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

然后根据图像数据生成纹理

```
glTexImage2D(GL_TEXTURE_2D,
    0,      //mipmap 层次(通常为，表示最上层)
    GL_RGB, //我们希望该纹理有红、绿、蓝数据
    bitmapInfoHeader.biWidth, //纹理宽带，必须是 n，若有边框+2
    bitmapInfoHeader.biHeight, //纹理高度，必须是 n，若有边框+2
    0, //边框(0=无边框，1=有边框)
    GL_RGB, //bitmap 数据的格式
    GL_UNSIGNED_BYTE, //每个颜色数据的类型
    bitmapptr[i]);  //bitmap 数据指针
```

最后进行绘制

```
glEnable(GL_TEXTURE_2D);//打开纹理
glBindTexture(GL_TEXTURE_2D, texture[teapot_bmp]); //选择纹理
/*具体的绘制*/
glDisable(GL_TEXTURE_2D); //关闭纹理
```

2．桌子纹理

桌子的纹理需要将纹理与桌子的每个角进行绑定，采取实验二中的方式，设计长方体和纹理的绑定函数：

```
void Draw_cube() {
    //设置立方体的顶点坐标
    GLfloat x1 = -0.5, x2 = 0.5;
    GLfloat y1 = -0.5, y2 = 0.5;
    GLfloat z1 = -0.5, z2 = 0.5;
    GLfloat V[8][3]{
        x1 , y1 , z1,
        x1 , y2 , z1,
        x2 , y2 , z1,
        x2 , y1 , z1,
        x2 , y1 , z2,
        x2 , y2 , z2,
        x1 , y2 , z2,
        x1 , y1 , z2,
    };
    //设置立方体的六个面对应顶点
    GLint Planes[6][4]{
        0 , 1 , 2 , 3,
        4 , 5 , 6 , 7,
        2 , 3 , 4 , 5,
        0 , 1 , 6 , 7,
        1 , 2 , 5 , 6,
        0 , 3 , 4 , 7,
    };
    GLint textmap[4][2] = { {1, 1}, {1, 0}, {0, 0}, {0, 1} };
    //对应纹理和长方形
    glBegin(GL_QUADS);
    for (int i = 0; i < 6; ++i)
        for (int j = 0; j < 4; ++j) {
            glTexCoord2iv(textmap[j]);
            glVertex3fv(V[Planes[i][j]]);
        }
    glEnd();
}
```

3．纹理混合

（1）光照与纹理

```
if(mixlight)  //设置纹理受光照影响
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
else  //设置纹理不受光照影响
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
```

采用回调函数控制全局变量，对比显示有无光照影响情况下的茶壶显示

（2）纹理和纹理

只需要将获取的两个纹理的数据进行平均就可以得到纹理混合之后的纹理数据，之后进行绑定过滤生成就可以得到混合纹理，注意需要给指针指向分配空间：

```
bitmapptr[3] = new unsigned char[49152];
for (int i = 0; i < 49152; i ++) {
    bitmapptr[3][i] = (bitmapptr[1][i] + bitmapptr[2][i]) / 2;
}
```

4．纹理生成

需要生成红黑间隔的格子纹理，那么就去准备对应的纹理数据，考虑到间隔着色，可以采取横纵下标相加为奇数则着色的方式，避免过于细密，对横纵下标都进行整除因此，每个大格子的边长都是 32 像素：

```
for (int i = 0; i < TEXH; ++i)
    for (int j = 0; j < TEXW; ++j) {
        tex[i][j][0] = GLubyte((i/16 + j/16) % 2 * 255);
    }
```

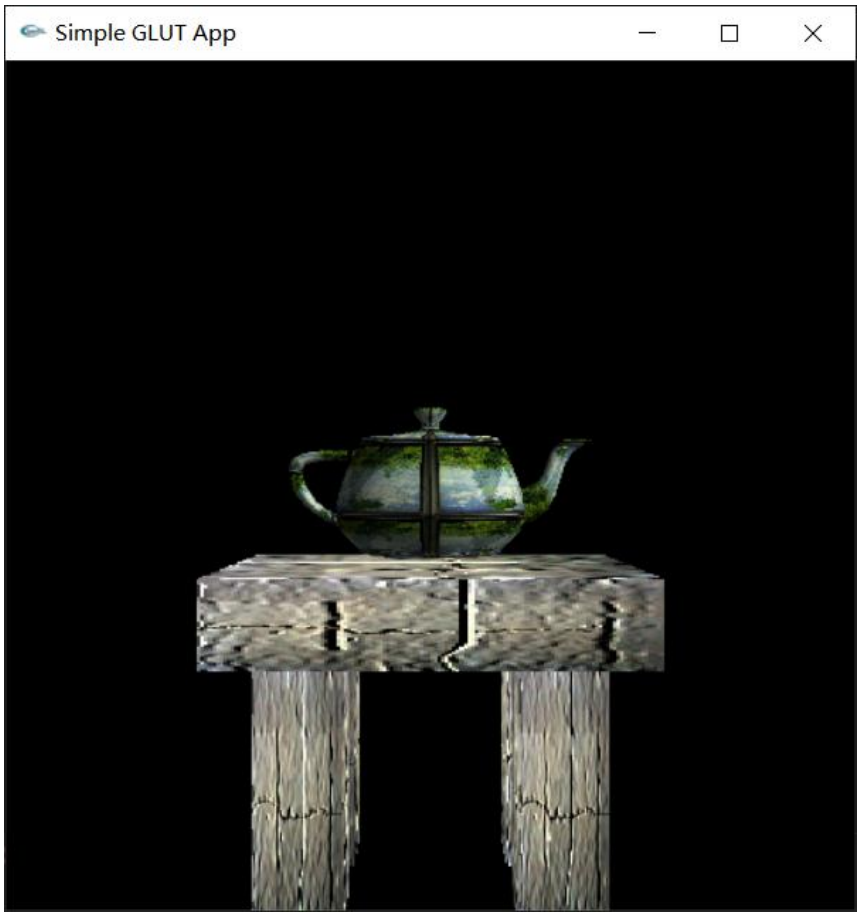
数据初始值全部是 0，也就是全黑色，将 RGB 的 R 改为 255 就可以得到红色。之后进行和前面相同的绑定过滤创建即可。

五、实验数据记录和处理

无数据

六、实验结果与分析

1．初始图像



2．去掉混合的光

3．改变茶壶纹理

4．混合纹理

七、讨论、心得

这次实验还是比较有趣的。绘制的过程与前面实验没有什么不同，只是增加了纹理的内容，只要弄清楚纹理的使用步骤时读取数据，绑定标识符，过滤纹理，创建纹理和使用纹理，就能轻松解决问题。采用纹理贴图比单纯的着色产生的效果更加逼真，也更具观赏性。本实验为大作业打下了基础。此外，纹理混合也使得我去接触 bmp 文件结构，对于文件的存储方式有了更多理解。