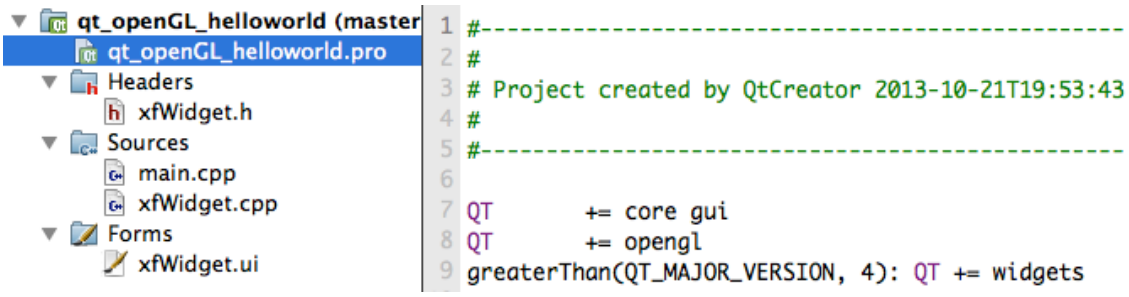
# OpenGL 作业报告 C 类第 3 题

夏斐 2012011417

November 9, 2013

## 1 实验环境

本作业在 Mac OS X10.8.2 系统下用 Qt Creator 2.8.2 完成，程序框架是 Qt GUI，用到了 Qt 的 OpenGL 框架，如图所示。为了程序能有更好的用户交互性和为之后的 A 类和 B 类作业打基础，之后的作业都用 Qt 完成。




## 2 实验原理

本次作业的原理是在立方体上完成贴图，完成贴图有几个方法，一是直接绑定纹理，在绘制面的过程的同时完成贴图。二是用 mipmapping。对于第一种方法，我们需要先载入贴图，在绘制时绑定贴图并且确定映射坐标和映射方式。就可以完成纹理贴图。

## 3 实验步骤

主函数如下，注意到和上一次相比，我们额外增加了 glutReshaoefunction，并且在其中设置透视投影矩阵。

Listing 1: Class xfWidget

```
iclass xfWidget : public QGLWidget
{
        Q_OBJECT
```

```
public:
        xfWidget( QWidget* parent = 0 );
        ~xfWidget();

protected:
        void initializeGL();
        void paintGL();
        void resizeGL( int width, int height );
        void keyPressEvent(QKeyEvent *e);
        void mousePressEvent(QMouseEvent *e);
        void mouseMoveEvent(QMouseEvent *e);
        void wheelEvent(QWheelEvent *e);
        void loadGLTextures();
private:
        int base;
        float angle;
        GLuint texture[6];
        GLfloat scaling;
        GLfloat xrot, yrot, zrot;
        QPoint lastPos;
        GLfloat posx,posy;


};
```

我们放弃了使用 Glut，转而使用 Qt 的 widget 来完成窗口的管理、和用户的交互，鼠标、键盘等时间的监听等，功能比原来更强大。

载入纹理的过程由下面一段程序实现：

Listing 2: Load Texture

```
void xfWidget::loadGLTextures()
{
    QString images[6] = {"1.png", "2.png", "3.png","4.png", "5.
       png", "6.png"};
    glGenTextures(6, &texture[0]);
    for (int i = 0; i < 6; ++i) {
        QImage t;
        QString now = QString("/Volumes/Macintosh HD/dev/OpenGL
           /OpenGL_Development/qt_openGL_helloworld/images/")+
```

```
            images[i];
        QImage b(now);
        t = QGLWidget::convertToGLFormat(b);
        glBindTexture(GL_TEXTURE_2D, texture[i]);
       glTexImage2D(GL_TEXTURE_2D, 0, 3, t.width(), t.height(),
            0, GL_RGBA, GL_UNSIGNED_BYTE, t.bits());
        glTexImage3D(GL_TEXTURE_3D, 0, 3, t.width(), t.height()
            ,0, 0, GL_RGBA, GL_UNSIGNED_BYTE, t.bits());
       //gluBuild2DMipmaps(GL_TEXTURE_2D,GL_RGBA,16,16,GL_RGBA,
          GL_UNSIGNED_BYTE,t.bits());
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
            GL_NEAREST);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
            GL_LINEAR);
    }
}
```

注意最后三行有一行被注释掉了，这是用于切换贴图模式的，将在效果展示中展示不同的效果。如果都改成 NEAREST 则不会作线性插值，放大之后会不够清晰，如果是 LINEAR 则会做线性插值。Mipmaping 则相当于选择了另一类的贴图模式。完成贴图部分用了以下的代码：

Listing 3: 贴图

```
    glBindTexture(GL_TEXTURE_2D, texture[5]);
    glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
    glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f,  1.0f);
    glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f,  1.0f,  1.0f);
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f,  1.0f, -1.0f);
    glEnd();
```

另外值得一提的是，我做了整个系统对于鼠标的响应，

Listing 4: mouseevent

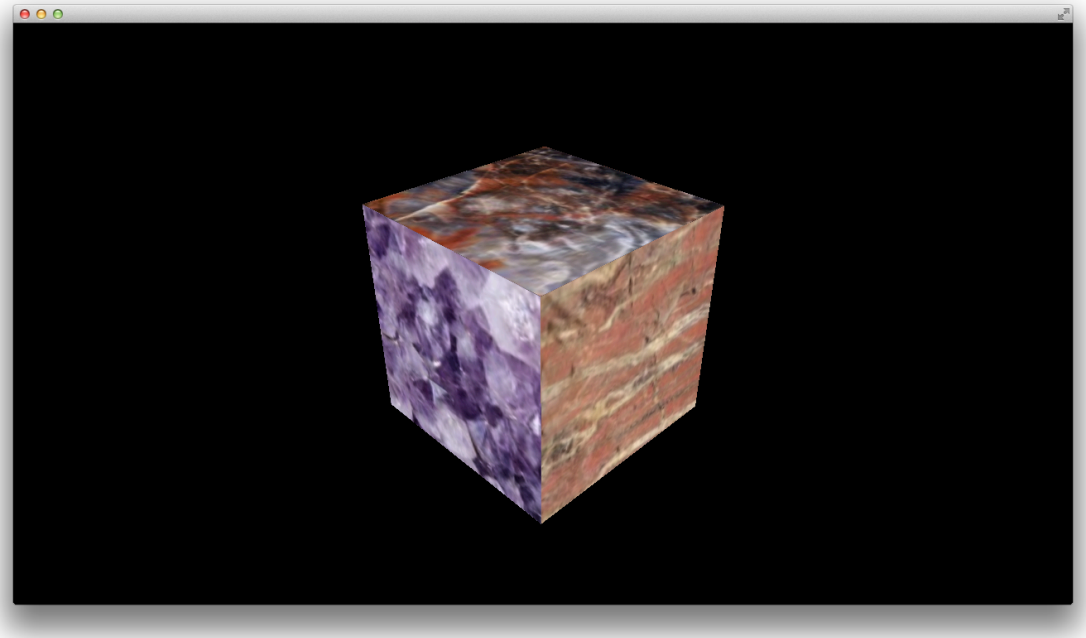
```
void xfWidget::mouseMoveEvent(QMouseEvent *e)
{
    GLfloat dx = GLfloat(e->x() - lastPos.x()) / width();
    GLfloat dy = GLfloat(e->y() - lastPos.y()) / height();
    if (e->buttons() & Qt::LeftButton) {
        xrot += 180 * dy;
```

```
        yrot += 180 * dx;
        updateGL();
    } else if (e->buttons() & Qt::RightButton) {
        xrot -= 180 * dy;
        zrot -= 180 * dx;
        updateGL();
    } else if (e->buttons() & Qt::MiddleButton) {
        posx+=dx*10.24;
        posy+=dy*5.76;
     updateGL();
     }
    lastPos = e->pos();
}
```

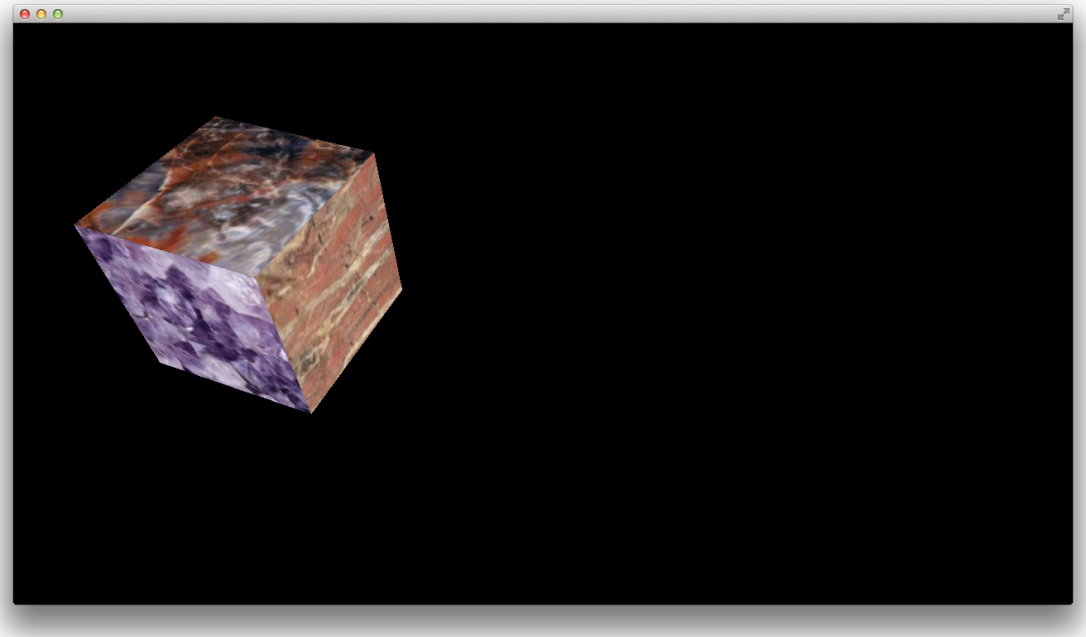
左键按下和右键按下可以做旋转，中间键按下可以做拖动。滚动滚轮可以缩放。

# 4 效果展示

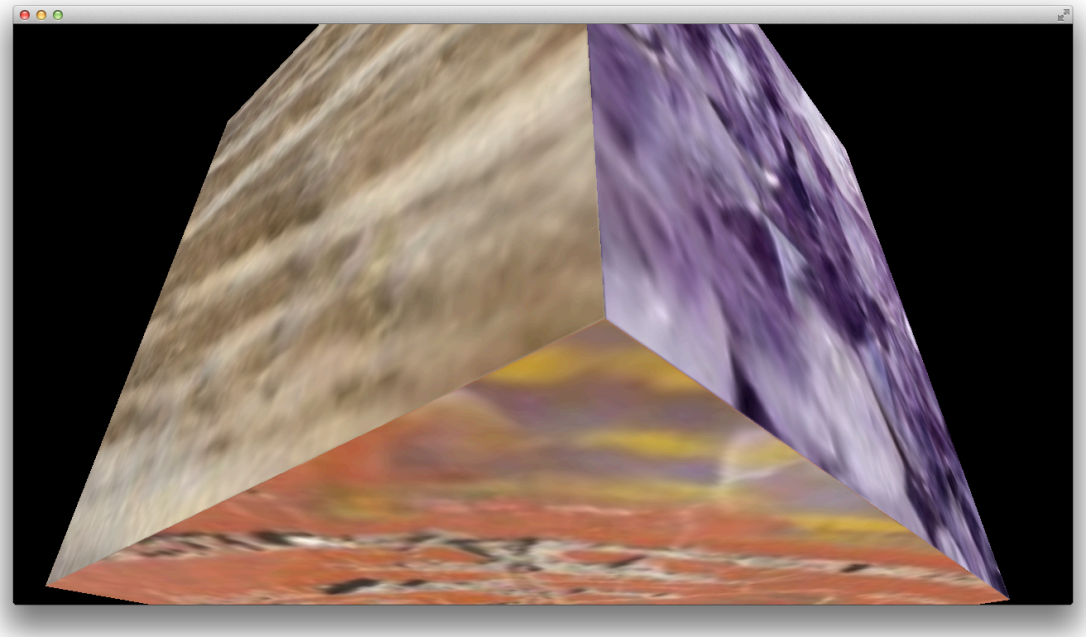
对六个面采用不同的图案贴图：



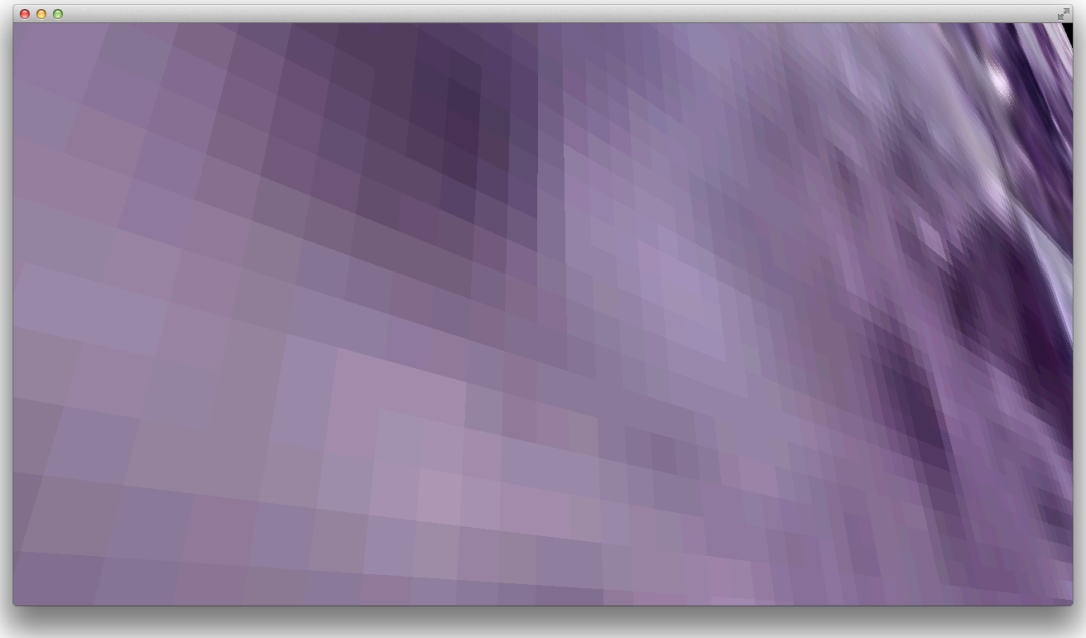
用鼠标旋转、移动立方体：



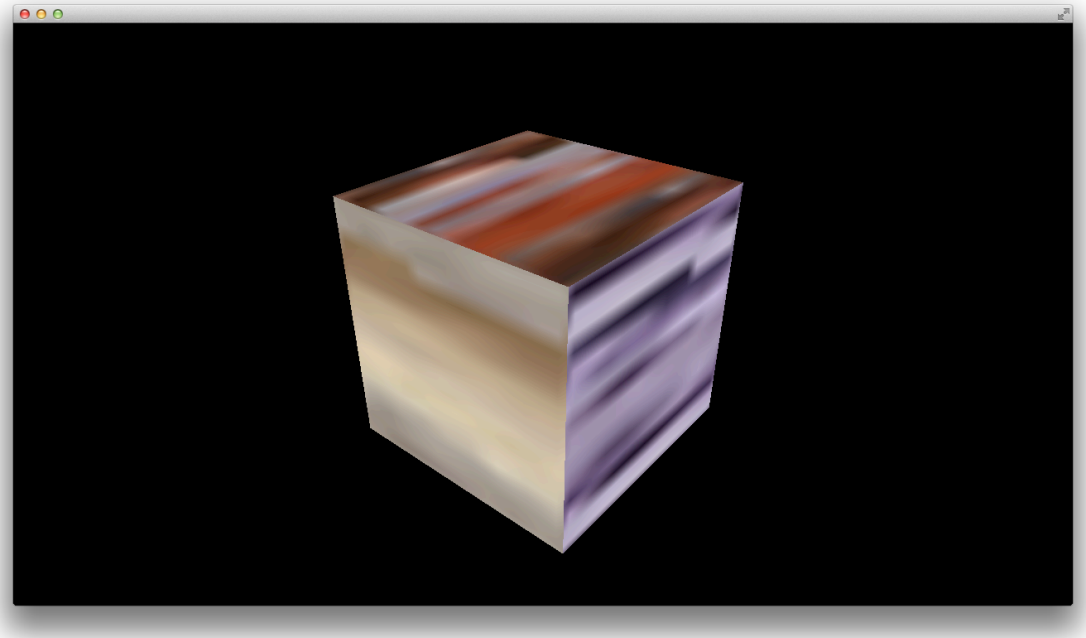
贴图时纹理滤波选择线性插值，贴图细腻，放大后不失真：



贴图时纹理滤波选择 NEAREST，贴图放大后可以看到马赛克：



采用 mipmapping 贴图：



# 5 参考文献

[1] 来自百度文库 OpenGL 入门教程 (精)
[2]Nehe OpenGL 中文教程

# 6 个人信息

夏斐
清华大学自动化系 2012 级
手机:(+86)15652799536
邮箱:xf1280@gmail.com