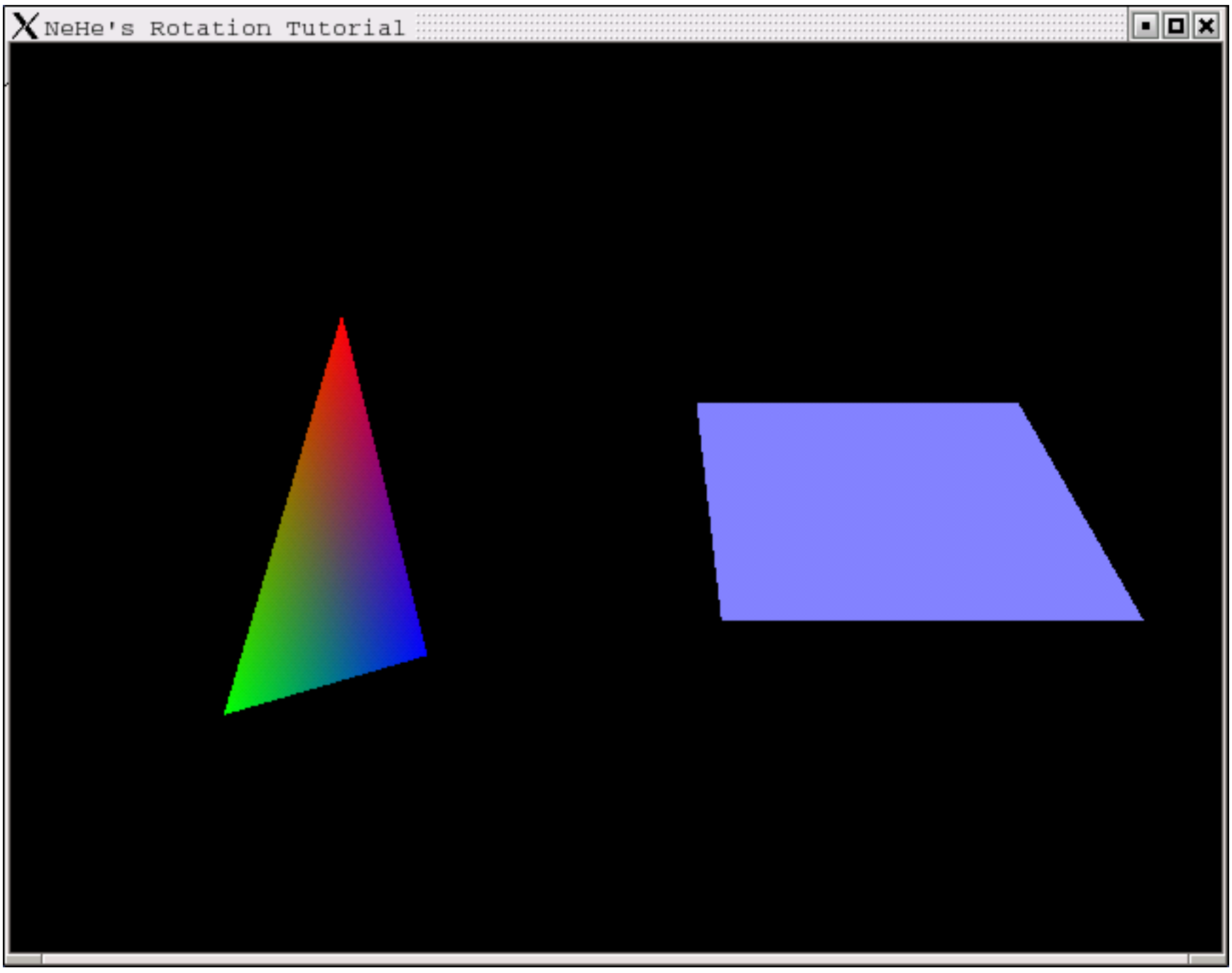
旋转



上一课中我教给您三角形和四边形的着色。这一课我将教您如何将这些彩色对象绕着坐标轴旋转。

其实只需在上节课的代码上增加几行就可以了。

我们将在NeHeWidget类中增加两个变量来控制这两个对象的旋转。它们是浮点类型的变量，使得我们能够非常精确地旋转对象。浮点数包含小数位置，这意味着我们无需使用1、2、3...的角度。你会发现浮点数是OpenGL编程的基础。新变量中叫做rTri的用来旋转三角形，rQuad 旋转四边形。

NeHeWidget类

（由nehewidget.h展开。）

```
protected:

  bool fullscreen;
```

```
  GLfloat rTri;
  GLfloat rQuad;

};
```

上面就是添加的两个变量。rTri是用于三角形的角度，rQuad是用于四边形的角度。

（由nehewidget.cpp展开。）

```
NeHeWidget::NeHeWidget( QWidget* parent, const char* name, bool fs )
    : QGLWidget( parent, name )
{
  rTri = 0.0;
  rQuad = 0.0;
  fullscreen = fs;
  setGeometry( 0, 0, 640, 480 );
  setCaption( "NeHe's Rotation Tutorial" );

  if ( fullscreen )
    showFullScreen();
}
```

我们需要在构造函数中给rTri和rQuad赋初值，都是0.0。

```
void NeHeWidget::paintGL()
{
  glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
  glLoadIdentity();
  glTranslatef( -1.5,  0.0, -6.0 );
  glRotatef( rTri,  0.0,  1.0,  0.0 );
```

glRotatef(Angle, Xvector, Yvector, Zvector)负责让对象绕某个轴旋转。这个函数有很多用处。 Angle通常是个变量代表对象转过的角度。Xvector，Yvector和Zvector三个参数则共同决定旋转轴的方向。比如(1, 0, 0)所描述的矢量经过X坐标轴的1个单位处并且方向向右。(-1, 0, 0)所描述的矢量经过X坐标轴的1个单位处，但方向向左。

D. Michael Traub：提供了对 Xvector , Yvector 和 Zvector 的上述解释。

为了更好的理解X, Y 和 Z的旋转，我举些例子...

X轴- 您正在使用一台台锯。锯片中心的轴从左至右摆放（就像OpenGL中的X轴）。尖利的锯齿绕着X轴狂转，看起来要么向上转，要么向下转。取决于锯片开始转时的方向。这与我们在OpenGL中绕着X轴旋转什么的情形是一样的。（CKer注：这会儿您要把脸蛋凑向显示器的话，保准被锯开了花 ^-^。）

Y轴- 假设您正处于一个巨大的龙卷风中心，龙卷风的中心从地面指向天空（就像OpenGL中的Y轴）。垃圾和碎片围着Y轴从左向右或是从右向左狂转不止。这与我们在OpenGL中绕着Y轴旋转什么的情形是一样的。

Z轴- 您从正前方看着一台风扇。风扇的中心正好朝着您(就像OpenGL中的Z轴)。风扇的叶片绕着Z轴顺时针或逆时针狂转。这与我们在OpenGL中绕着Z轴旋转什么的情形是一样的。

上面的一行代码中，如果rtri等于7，我们将三角形绕着Y轴从左向右旋转7 。您也可以改变参数的值，让三角形绕着X和Y轴同时旋转。

```
  glBegin( GL_TRIANGLES );
    glColor3f( 1.0, 0.0, 0.0 );
    glVertex3f(  0.0,  1.0,  0.0 );
    glColor3f( 0.0, 1.0, 0.0 );
    glVertex3f( -1.0, -1.0,  0.0 );
    glColor3f( 0.0, 0.0, 1.0 );
    glVertex3f(  1.0, -1.0,  0.0 );
  glEnd();
```

上面的绘制三角形的代码没有改变。在屏幕的左面画了一个彩色渐变三角形，并绕着Y轴从左向右旋转。

```
  glLoadIdentity();
```

我们增加了另一个glLoadIdentity()调用。目的是为了重置模型观察矩阵。如果我们没有重置，直接调用glTranslate的话，会出现意料之外的结果。因为坐标轴已经旋转了，很可能没有朝着您所希望的方向。所以我们本来想要左右移动对象的，就可能变成上下移动了，取决于您将坐标轴旋转了多少角度。试试将glLoadIdentity() 注释掉之后，会出现什么结果。

重置模型观察矩阵之后，X、Y、Z轴都以复位，我们调用glTranslate。您会注意到这次我们只向右移了1.5单位，而不是上节课的3.0单位。因为我们重置场景的时候，焦点又回到了场景的中心(0.0)处。这样就只需向右移1.5单位就够了。 当我们移到新位置后，绕X轴旋转四边形。正方形将上下转动。

```
  glTranslatef(  1.5,  0.0, -6.0 );
  glRotatef( rQuad,  1.0,  0.0,  0.0 );
```

绕X轴旋转四边形。

```
  glColor3f( 0.5, 0.5, 1.0 );
  glBegin( GL_QUADS );
    glVertex3f( -1.0,  1.0,  0.0 );
    glVertex3f(  1.0,  1.0,  0.0 );
    glVertex3f(  1.0, -1.0,  0.0 );
    glVertex3f( -1.0, -1.0,  0.0 );
  glEnd();

  rTri += 0.2;
  rQuad -= 0.15;
```

我们在构造函数中已经将rTri和rQuad的值设为0.0，在这里我们每绘制完一次图像，就修改一下这两个变量。两个变量的变化会使对象的旋转角度发生变化。

尝试改变下面代码中的+和-，来体会对象旋转的方向是如何改变的。并试着将0.2改成1.0。这个数字越大，物体就转的越快，这个数字越小，物体转的就越慢。

```
}
```

在这一课中，我试着尽量详细的解释如何让对象绕某个轴转动。改改代码，试着让对象绕着Z轴、

X+Y轴或者所有三个轴来转动:)。

本课程的源代码。

[上一课：上色] [Qt OpenGL教程主页] [下一课：向三维进军]

---

http://www.qiliang.net mailto:cavendish@qiliang.net 2002年12月21日