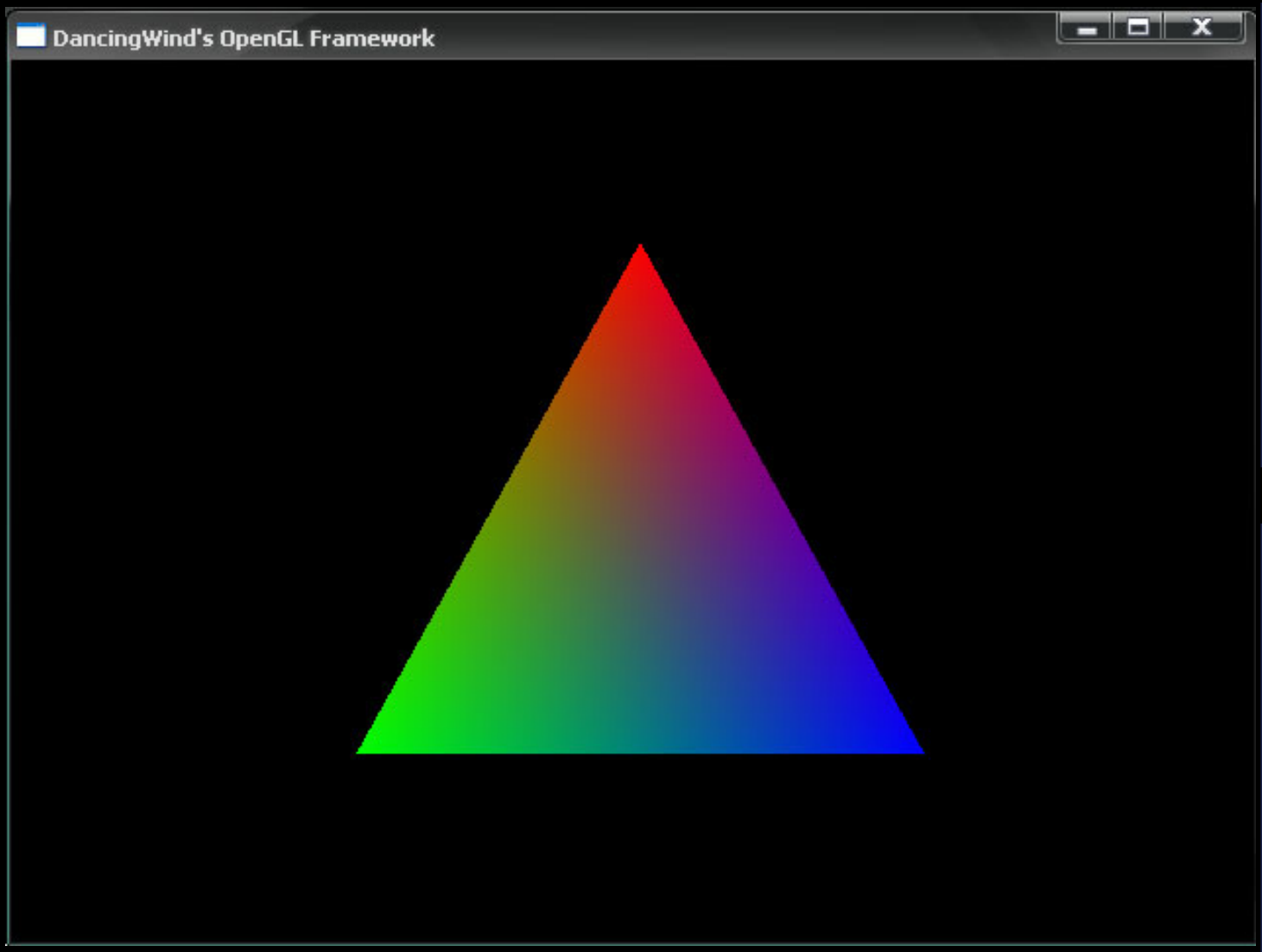
2、设置视口和重载你的绘制函数



NeHe SDK是把Nehe的教程中所介绍的所有功能，以面向对象的形式，提供给编程人员快速开发的一套编程接口。在下面的教程中，我将按NeHe SDK源码的功能分类，一步一步把这套api介绍给大家。如果你觉得有更好的学习方法，或者有其他有益的建议，请联系我。zhouwei02@mails.tsinghua.edu.cn, zhouwei506@mails.gucas.ac.cn

程序结构：
我们在第一课程序的基础上添加一下功能：

1. 创建一个全局的视口类，控制场景中可见的范围
2. 创建初始化函数，用来完成绘制的初始化工作
3. 创建一个绘制三棱锥的函数
4. 设置默认的视口棱台体

5. 重栽我们的绘制函数，这里将绘制一个简单的三棱锥

为了使用视口类,我们需要包含下面的头文件(view.h)，并声明一个全局的视口变量view。

```
/**********************************新增的代码：包含视口类的声明******************/
#include "view.h"                                   // 包含视口类的声明
/**********************************新增的代码：包含视口类的声明:结束*****************/

/**********************************新增的代码：创建View类**************************/
View    view;                               // 创建视口类
/**********************************新增的代码：创建View类:结束*********************/
```

接着创建一个新的cpp文件，用来重栽我们的绘制函数。这里我把它命名为Draw.cpp

为了使用OpenGL类和View类，我们需要包含头文件opengl.h和view.h。
为了方便起见，我们启用NeHe名字空间。
接着我们使用extern关键字来使用全局变量view，最后我们创建一个变量initialize来记录是否需要初始化。

整个代码如下：

```
#include "opengl.h"                         // 包含创建OpenGL程序的框架类
#include "view.h"                               // 包含视口类的声明

#pragma comment( lib, "NeheSDK.lib" )       // 包含NeheSDK.lib库

using namespace NeHe;                       // 使用NeHe名字空间

extern View view;                               // 使用全局变量view类
static bool initialize = true;                  // 记录是否初始化
```

2、设置初始化函数

这里我创建一个空的初始化函数，以方便以后的使用

```
/*****************************************************初始化场景
*****************************************************************************/
void IniScene(OpenGL* gl, ControlData* cont)
{
```

```
}
/*********************************************初始化场景:结束
*****************************************************************/
```

3、创建一个绘制三棱锥的函数

我们使用标准的OpenGL代码创建这个函数

```
/*********************************************绘制三棱锥
*****************************************************************/
// 绘制三棱锥
void DrawTri(void)
{
    glBegin(GL_TRIANGLES);
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 0.0f, 1.0f, 0.0f);
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 0.0f, 1.0f, 0.0f);
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f( 1.0f,-1.0f, 1.0f);
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f( 1.0f,-1.0f, -1.0f);
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 0.0f, 1.0f, 0.0f);
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f( 1.0f,-1.0f, -1.0f);
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f(-1.0f,-1.0f, -1.0f);
        glColor3f(1.0f,0.0f,0.0f);
        glVertex3f( 0.0f, 1.0f, 0.0f);
        glColor3f(0.0f,0.0f,1.0f);
        glVertex3f(-1.0f,-1.0f,-1.0f);
        glColor3f(0.0f,1.0f,0.0f);
        glVertex3f(-1.0f,-1.0f, 1.0f);
    glEnd();
}
/*********************************************绘制三棱锥:结束
*****************************************************************/
```

4、设置默认视口棱台体

默认的视口棱台体的范围是，视角45度，近切面距离视点0.1，远切面距离视点100。
默认在z=0的平面上的可见范围是，Y轴方向(-2,+2),X轴方向(-2,+2)*宽高比
宽高比由窗口大小决定，如果窗口大小为800x600，则宽高比为4/3，X轴的可见范围是(-2.67,+2.67)

代码如下：

```
/*****************************************************设置默认视口棱台体
*************************************************************************/
void SetDefaultView()
{
    view.Reset();                                   // 重置视口
    view.LookAt(Vector(0,0,5),Vector(0,0,0),Vector(0,1,0));      // 设定视口在(0,0,5),朝向-Z轴，上方向量为Y轴
}
/*****************************************************设置默认视口棱台体：结束
*************************************************************************/
```

5、绘制场景

在这个例子里我们通过绘制一个三角形来说明这个函数。它完成一下的功能 ：

1. 初始化场景
2. 设置默认的视口棱台体
3. 调用自定义的绘制函数

代码如下：

```
/*****************************************************绘制场景
*************************************************************************/
void DrawScene(OpenGL *gl,ControlData *cont)
{
    // 初始化
    if(initialize)
    {
        IniScene(gl,cont);
        initialize = false;
    }


    SetDefaultView();

    DrawTri();                  // 绘制金字塔
```

```
}
/*****************************************************绘制场景:结束
*************************************************************************/
```

好了,上面就是简单的使用view类和重载你的绘制函数的步骤,很简单吧:)