实验内容：

1. 解压GlutLib.zip文件，并按照“glut库安装.txt”中的说明，安装glut库。
2. 阅读文档CreateProject.txt，建立自己的工程文件。
3. 基于给定的源代码cube.cpp等分别建立工程文件，并编译执行。
4. 仔细阅读程序注释，了解利用OpenGL进行计算机图形学编程的框架。
5. 基于aim.cpp、lines.cpp两个源文件分别建立工程文件，调试通过并执行。
6. 通过对给定代码进行局部修改，进一步了解部分函数的意义。
7. 选作：基于glut 和OpenGL实现用橡皮筋交互定义矩形的功能。

实验要求：

1. **通过对给定程序的试运行，解释以下几个函数的作用及各参数的意义：**

glTranslatef(float x, float y, float z);

定义一个平移矩阵，该矩阵与当前矩阵相乘，使后续的图形进行平移变换。

参数x,y,z分别指定沿x,y,z轴方向的平移分量。

glVertex3f(float x, float y, float z);

该函数用于指定一个三维点，表示物体的顶点。

x,y,z分别为该顶点的x，y，z坐标

glColor3f (float a, float b, float c);

该函数用于设置当前颜色。

参数abc依次是RGB三个分量的值，范围是0.0-1.0。

glLineWidth (float a);

设置线的宽度。参数a越大则宽度越宽

glutInitWindowSize (int a,int b);

重置窗口大小，参数a,b分别表示窗口宽度和高度

glutInitWindowPosition (int a,int b);

初始化窗口的位置。具体是窗口左上角相对于桌面坐标(a , b)

坐标(0,0)是指屏幕左上角的位置。

1. **分析以下两个函数的区别：**

**glVertex3f(float x, float y, float z);**

**glVertex2f(float x, float y);**

显然，主要区别在于指定的维度。glVertex3f用于表示三维空间中的顶点，其中包括 x、y 和 z 三个坐标。而glVertex2f用于表示二维空间中的顶点，只包括 x 和 y 两个坐标。选择使用哪个函数取决于你要绘制的图形的维度。如果是在三维空间中进行绘制，你可能会使用glVertex3f，而在二维空间中进行绘制时，你可以使用glVertex2f。

最后的字母表示参数的类型：

s表示16位整数（OpenGL中将这个类型定义为GLshort），

i表示32位整数（OpenGL中将这个类型定义为GLint和GLsizei），

f表示32位浮点数（OpenGL中将这个类型定义为GLfloat和GLclampf），

d表示64位浮点数（OpenGL中将这个类型定义为GLdouble和GLclampd）。

v表示传递的几个参数将使用指针的方式。

这些函数除了参数的类型和个数不同以外，功能是相同的。

1. **说明如何利用glut进行OpenGL应用程序的窗口及输入、输出管理。**
2. 包含头文件：

#include <GL/glut.h>

1. 初始化 GLUT：

在 main 函数中初始化 GLUT。这个初始化包括对 GLUT 库进行初始化和设置窗口的一些属性。

glutInit(&argc, argv);

glutCreateWindow (argv[0]);

1. 设置窗口属性：

通过 glutInitWindowSize 和 glutInitWindowPosition 设置窗口的大小和位置。

glutInitWindowSize (500, 500);

glutInitWindowPosition (100, 100);

通过glutInitDisplayMode 函数用于设置窗口的显示模式，它定义了窗口的一些基本属性。GLUT\_SINGLE | GLUT\_RGB两个参数表示设置窗口为单缓冲模式，并使用 RGB 颜色模式。

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

1. 创建窗口：

glutCreateWindow (argv[0]);

argv[0] 将程序的名称作为窗口的标题。

1. 注册回调函数：

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMouseFunc(mouse);

处理显示、窗口改变和鼠标，键盘事件。

1. 设置绘图环境：

在 display 回调函数中设置 OpenGL 的绘图环境，并进行绘制。

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

// 绘制图形

glutSwapBuffers();

}

1. 处理输入事件：

在注册的键盘和鼠标回调函数中处理输入事件。

void keyboard(unsigned char key, int x, int y) {

// 处理键盘事件

}

key为键盘按键的ASCII码

x： 表示触发键盘事件时鼠标的横坐标。

y： 表示触发键盘事件时鼠标的纵坐标。

void glutMouseFunc(void(\*func)(int button,int state,int x,int y));

处理鼠标输入的事件；

button为GLUT\_LEFT\_BUTTON或GLUT\_RIGHT\_BUTTON,分别表示左右按键。

state为按键的状态，若为按下则为GLUT\_DOWN

x, y 表示鼠标的位置，

1. 进入 GLUT 主循环：

glutMainLoop();

在所有初始化工作完成后，进入 GLUT 的主循环，等待事件发生。

②处理输入的部分包括：

以及在主函数中注册回调函数：

glutKeyboardFunc(keyboard);

glutMouseFunc(mouse);

注册的键盘和鼠标回调函数

void keyboard(unsigned char key, int x, int y)；

void glutMouseFunc(void(\*func)(int button,int state,int x,int y));

③处理输出的部分：

创建窗口

glutCreateWindow(“窗口名”);

显示回调，GLUT将调用该窗口，显示回调函数。

glutDisplayFunc(display);

glutReshapeFunc(reshape);

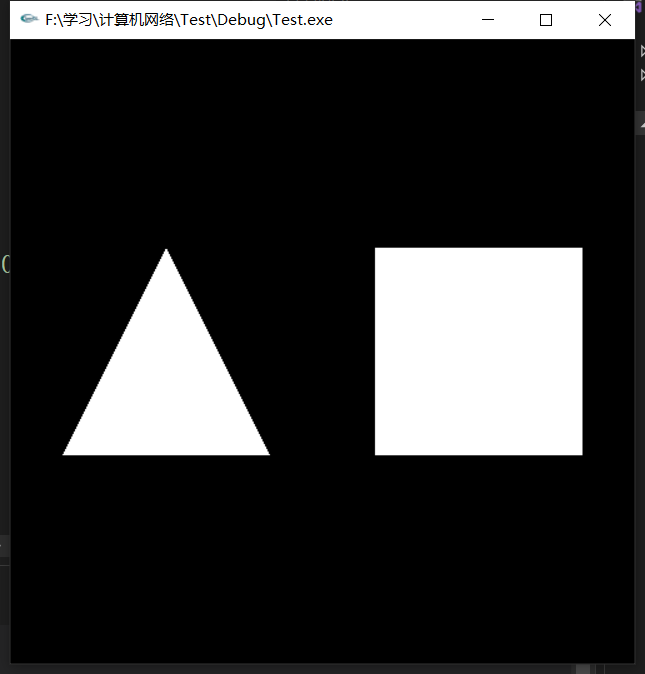
glutKeyboardFunc(keyboard);

glutMouseFunc(mouse);

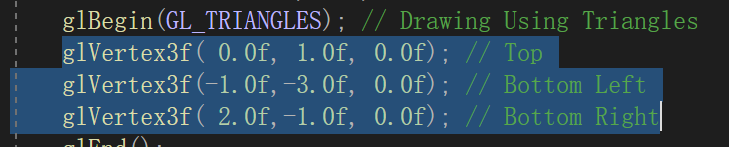
1. 分别展示对三个代码的修改，以及执行结果，对比与原来代码执行结果的不同，并说明原因（展示2-3个即可）。

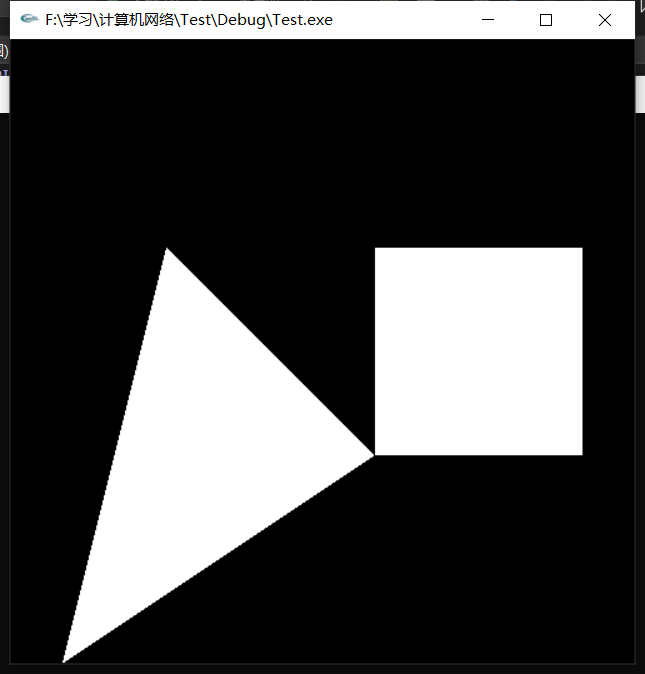
Aim文件：

原图像展示：

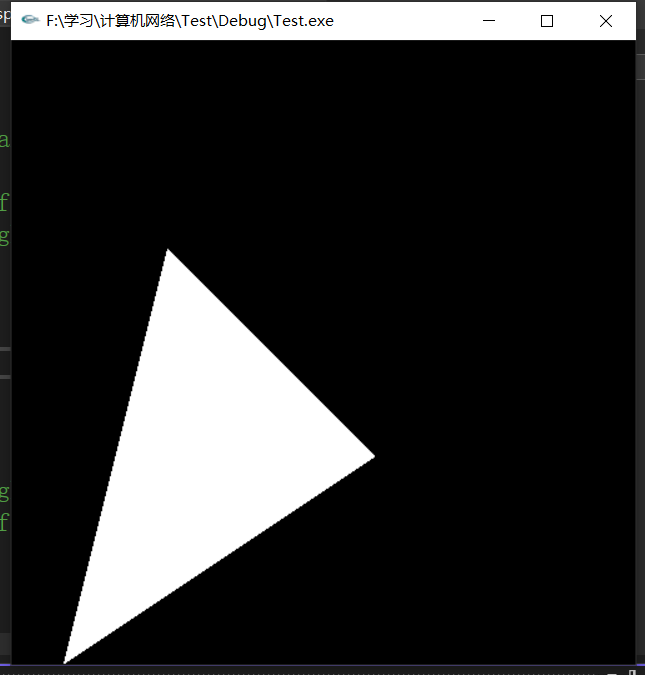


修改三角形的三个顶点坐标之后：





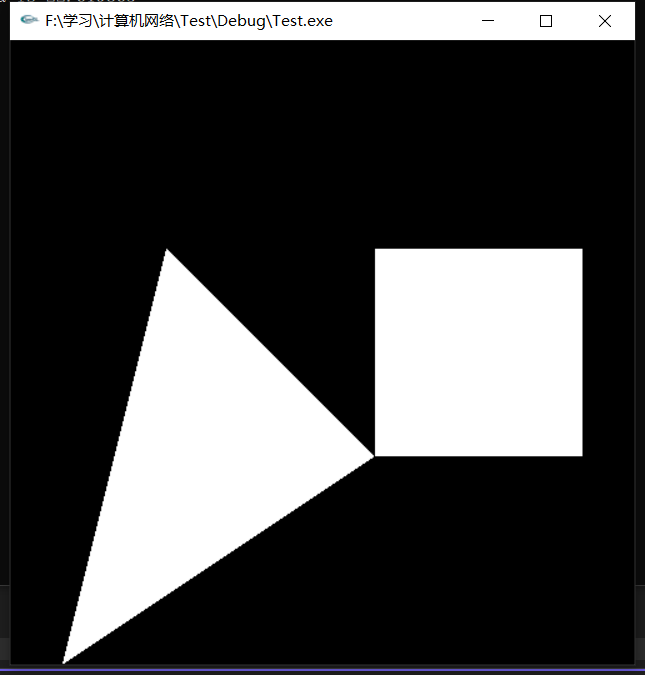
修改正方形的绘制顺序：glFrontFace(GL\_CCW); 使其按顺时针定义点的图形不能正常显示。因为正方形逆时针为正面。



这时添加代码glCullFace(GL\_FRONT);就可以重新设置顺时针为正

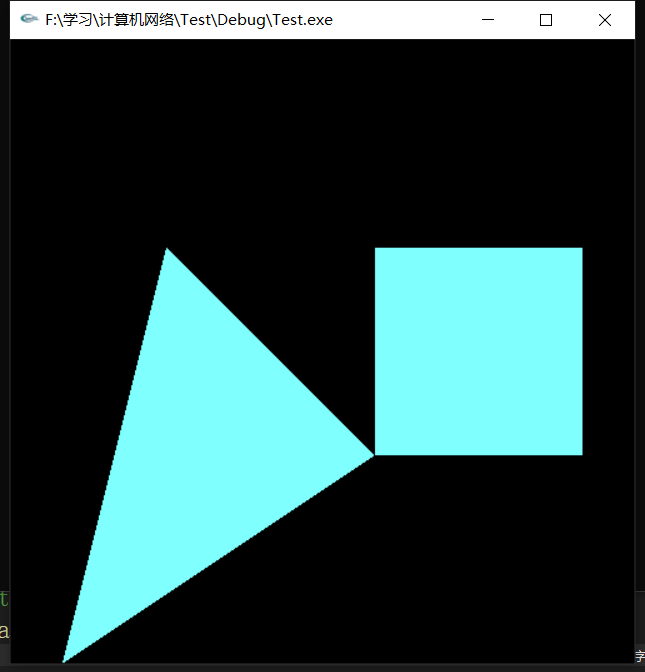
glFrontFace(GL\_CCW);

glCullFace(GL\_FRONT);

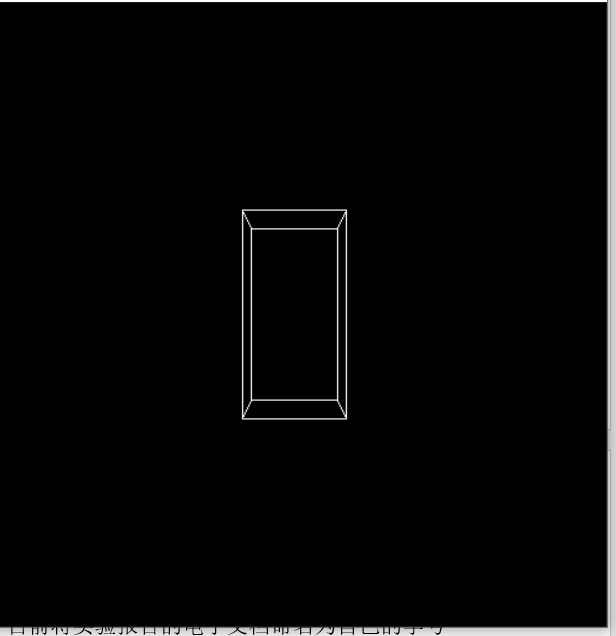


修改颜色：

glColor3f (0.5, 1.0, 1.0);



Cube.cpp:



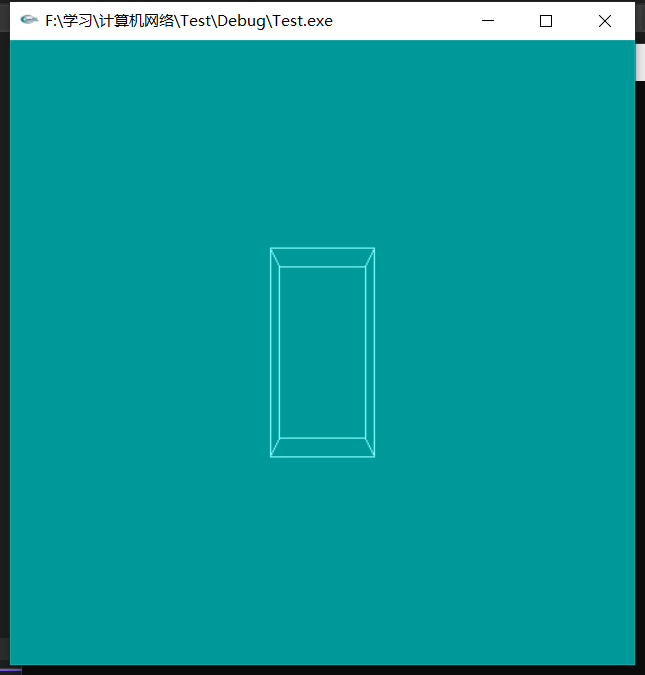
修改背景颜色和绘制图形的颜色：

在init()函数修改窗口（背景）颜色

glClearColor (0.0, 0.6, 0.6, 0.0);

在display()函数中修改绘制图形颜色：

glColor3f (0.5, 1.0, 1.0);

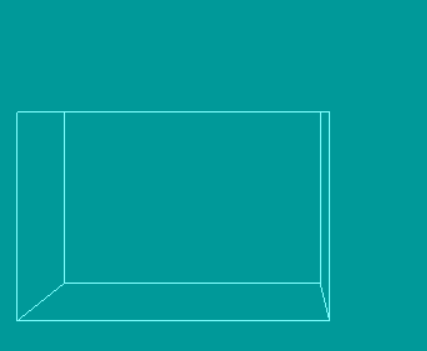


平移坐标系，将物体移动到指定位置

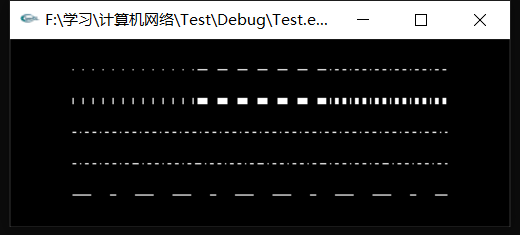
glTranslatef (-1, -1, -5.0);

缩放坐标系，使物体在某个方向上变形

glScalef (3.0, 2.0, 1.0);



Line.cpp:

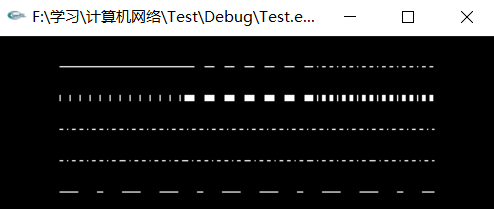


原图像：

修改代码：第一行直线的第一部分：

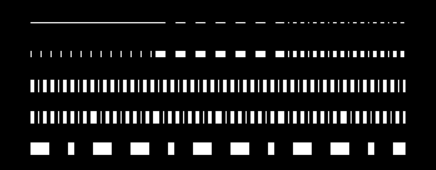
创建一个点线型的线段，每个点的间隔为2个像素，线段中的所有像素都会显示。

glLineStipple (2, 0xFFFF); /\* dotted \*/



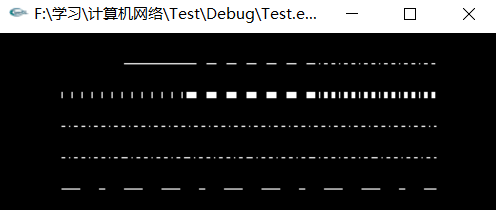
修改直线宽度：

glLineWidth (10.0);



改变直线的绘制起点：（第一条直线）

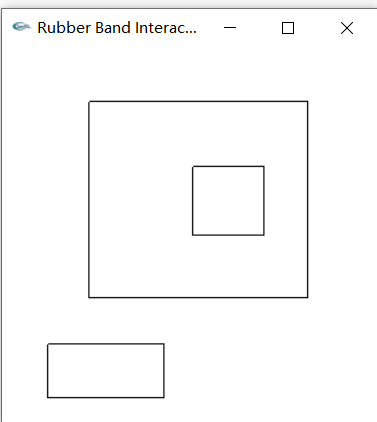
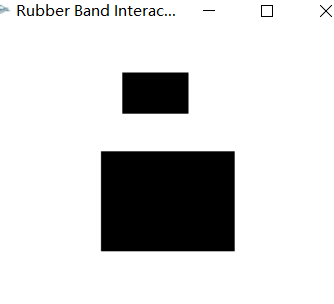
drawOneLine (100.0, 125.0, 150.0, 125.0);



1. 选作：基于glut 和OpenGL实现用橡皮筋交互定义矩形的功能。

源文件存储在rubber.cpp文件中

实验效果：

绘制矩形框和实体矩形

在实现这个功能的时候，我利用了rectangles向量，每当释放鼠标按钮时，生成的矩形坐标都会保存在其中，并在display回调中绘制所有矩形，实现了在绘制过程中所有绘制图形的暂存。

在drawRectangle函数中，我通过修改glBegin的参数值，实现了绘制矩形框和填充矩形的不同效果。

GL\_LINE\_LOOP为绘制矩形框，GL\_QUADS为绘制填充矩形。

代码大致思路：

* 这段代码定义了一个结构体 Rectangle 用于表示矩形，包含矩形的两个对角点坐标。
* 使用全局变量和标志位来跟踪程序的状态。rectangles 是一个存储所有矩形的容器，rubberBandActive 用于标志当前是否正在进行橡皮筋交互。
* drawRectangle 函数用于绘制矩形，这里使用 GL\_QUADS 绘制填充矩形，并添加了颜色填充。
* display 函数负责清除屏幕并绘制所有矩形。如果橡皮筋交互活跃，也绘制当前的橡皮筋。
* mouse 函数处理鼠标按下和释放事件。在鼠标按下时，记录起始坐标，并启动橡皮筋交互。在鼠标释放时，记录结束坐标，停止橡皮筋交互，并将生成的矩形坐标存储到 rectangles 容器中。
* 鼠标移动回调函数： motion 函数在橡皮筋交互过程中更新结束坐标，从而实现实时显示橡皮筋。
* 在 main 函数中进行OpenGL和GLUT的初始化，创建窗口，并设置相关回调函数。然后通过 glutMainLoop 进入主事件循环。

实验结果及结论分析：

通过本次实验我熟悉并掌握如何建立包含OpenGL库函数的工程文件，阅读和修改所给代码让我了熟悉如何利用glut进行OpenGL应用程序的窗口及输入、输出管理，学会了搭建OpenGL框架，在框架里添加内容来绘制图形，这一过程十分有趣，让我对图形学的兴趣大增。

在实验中修改函数，熟悉并了解OpenGL各种库函数的实际功能，对运用库函数进行日后的创作大有脾益，过程发现了几个需要注意的函数，比如glBegin和glEnd是成对出现的函数，用来限制单独的图形绘制，如果只出现其中一个最后绘制出来的应该不会是想要的效果。还有glFrontFace和glCullFace，这两个函数对图形前后面判断起着重要作用，在图形学中，不特殊声明一般绘制单面图形，这是背面就不会填充像素也就是当背面面向屏幕时看不到图形，通过设置这两个函数的参数，就能控制图形显示背面还是正面。

选做实现了橡皮筋绘制图形的方法，代码中尝试了多种图形的情况。在实验中出现了显示不出图形的情况，因此我定义了一个结构体存储每一个矩形的顶点信息，以便能够正常的显示矩形。同时，在编写代码的过程中遇到了exit重定义的报错，到网上查找了资料后发现，是没有包含指定的头文件，因此就会出现了问题。加上头文件之后程序正常运行。在编写代码的过程中逐渐熟悉并掌握了glut库函数鼠标键盘的输入方法以及如何定义逻辑来绘制图形。