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。

实验目的：

1. 掌握如何建立包含OpenGL库函数的工程文件；
2. 熟悉如何利用glut进行OpenGL应用程序的窗口及输入、输出管理等；
3. 熟悉OpenGL的函数。

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：
2. glTranslatef(float x, float y, float z)：

用于对于待渲染物体进行平移变换，或者说是对坐标系进行平移变换。x、y 和 z 分别表示在 x、y 和 z 轴上的平移距离。

1. glVertex3f(float x, float y, float z)：

指定一个三维坐标点，用于绘制图形中的顶点（三维坐标系中）。x、y 和 z 是顶点的坐标值。

1. glColor3f(float a, float b, float c)：

设置当前绘制颜色，a、b 和 c 表示红、绿和蓝色分量。

1. glLineWidth(float a)：

设置线条的宽度，a 表示线条的宽度，单位为像素。

1. glutInitWindowSize(int a, int b)：

初始化窗口大小，a 和 b 分别是窗口的宽度和高度，单位为像素。

1. glutInitWindowPosition(int a, int b)：

初始化窗口的位置。a 和 b 分别是窗口在屏幕上的左上角位置的 x 和 y 坐标。

1. 分析以下两个函数的区别：

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

glVertex2f(float x, float y);

glVertex3f 用于指定三维坐标的顶点，而 glVertex2f 则只指定了二维坐标的顶点。所以glVertex3f 在三维空间中使用，而 glVertex2f 在二维空间中使用。

在示例代码中，glVertex2f在绘制line的程序中被使用，该cpp文件中使用 gluOrtho2D() 函数定义二维正交投影，这样就可以在一个二维坐标系中绘制图形。

gluOrtho2D(0.0, 400.0, 0.0, 150.0) 将屏幕区域映射到了一个 400x150 的二维坐标系，使得坐标范围在 x 轴从 0 到 400，y 轴从 0 到 150。

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. 窗口管理函数：

glutInit(&argc, argv): 初始化 GLUT 库，传递命令行参数。

glutInitWindowSize(int a, int b): 初始化窗口大小，设置窗口的宽度和高度。

glutInitWindowPosition(int a, int b): 初始化窗口的位置，设置窗口在屏幕上的位置。

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB): 定义窗口显示模式，设置窗口的显示属性。

glutCreateWindow(const char \*title): 创建窗口并指定窗口标题，生成一个具有指定标题的新窗口。

1. 输入管理函数：

glutMouseFunc(void (\*func)(int button, int state, int x, int y)): 处理鼠标事件，指定鼠标事件的回调函数。

glutKeyboardFunc(void (\*func)(unsigned char key, int x, int y)): 处理键盘按键事件，指定键盘按键事件的回调函数。

glutSpecialFunc(void (\*func)(int key, int x, int y)): 处理特殊键（如方向键）事件，指定特殊键事件的回调函数。

1. 输出管理函数：

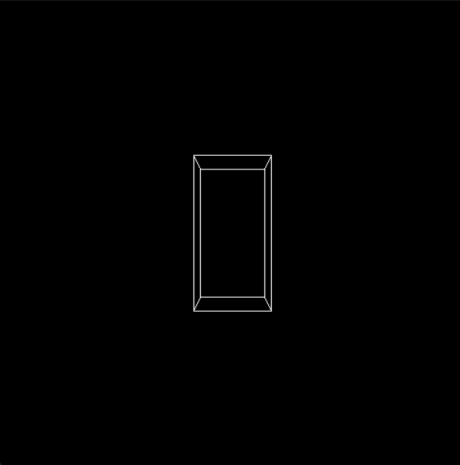
glutReshapeFunc(void (\*func)(int width, int height)): 指定窗口重塑函数，设置窗口大小变化时的回调函数。

glutDisplayFunc(display): 指定显示回调函数，设置窗口内容的绘制函数。

glutMouseFunc(mouseDown): 指定鼠标回调函数，处理鼠标事件的回调函数。

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）。
2. cube.cpp

原始运行结果如下图所示。

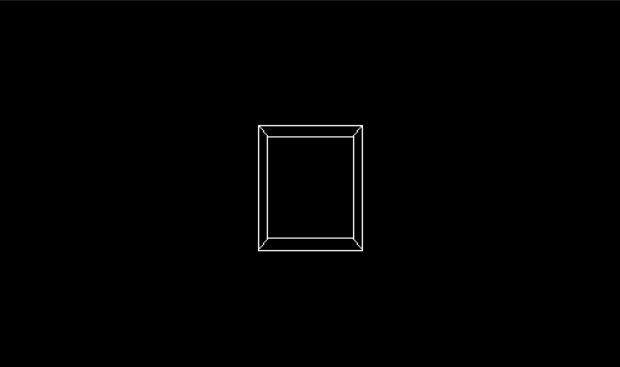


该代码实现了在display函数中调用glutWireCube (1.0)绘制一个线框方块显示在屏幕上，并在窗口的glutDisplayFunc函数中调用display函数。下面是对代码中部分函数进行修改的例子。

* glutInitWindowSize (500, 300);

原代码：glutInitWindowSize (500, 500);

运行结果如下，可以见到该线框方块被压缩，这是因为OpenGL在渲染时会根据窗口的宽高比来显示场景。当窗口的高度减少而保持宽度不变时，如果没有相应地调整投影矩阵，图形就会发生类似下图的形变。

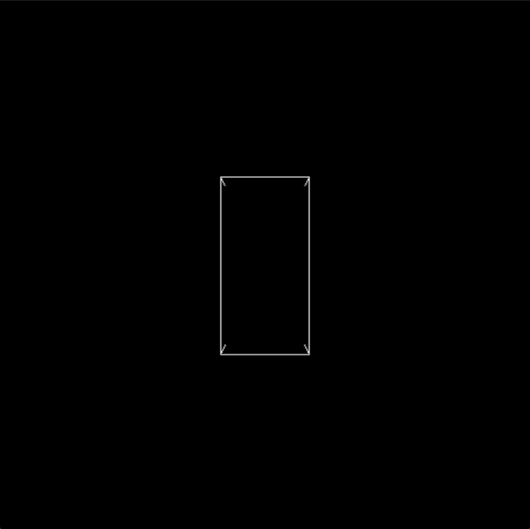


* glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 5.0);

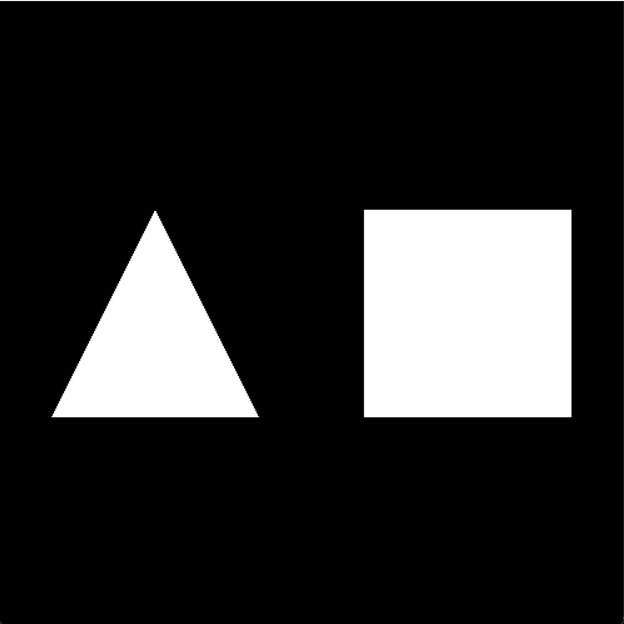
原代码：glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);

glFrustum() 是OpenGL中用于创建透视投影矩阵的函数之一。它定义了一个视景体Frustum，将场景中的物体投影到视景体内。参数列表（left, right, bottom, top, near, far）表示了视景体的六个面的位置：左右边界、上下边界、近远距离，

这些参数定义的区域内的场景会被投影，而在之外的物体都会被裁剪，当改变了参数far的值为5.0后，cube的后半部分被裁减掉，如下图所示。



1. aim.cpp



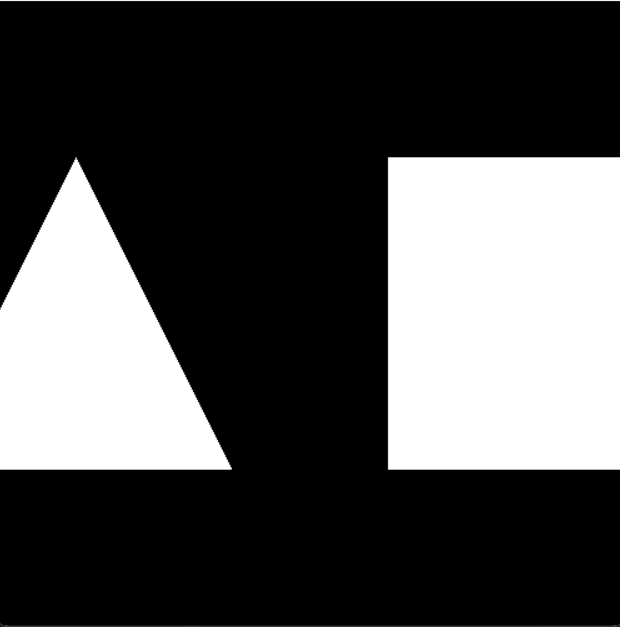
该代码实现了在display函数中调用glBegin绘制一个三角形和一个正方形显示在屏幕上，并在窗口的glutDisplayFunc函数中调用display函数。glBegin通过传入不同的参数（GL\_TRIANGLES、GL\_QUADS等）来决定绘制的几何形状，通过定义顶点（glVertex3f实现）并连接这些顶点的方式实现绘制。

下面是对代码中部分函数进行修改的例子。

* glTranslatef(-1.5f, 0.0f,-10.0f);

原代码：glTranslatef(-1.5f, 0.0f, -15.0f);

原代码执行了一个平移变换，将当前坐标系沿着 X 轴负方向平移了 1.5 个单位，同时沿着 Z 轴负方向平移了 15 个单位。glTranslatef通常用于将场景中的对象定位到一个特定的位置以便于观察和渲染。此处沿着 Z 轴负方向的移动是为了将要绘制的物体放置在摄像机的视野内，使得物体可见。更改-15.0f为-10.0f后，物体摄像机的距离变小，一部分被裁剪在视野外，结果如下所示。



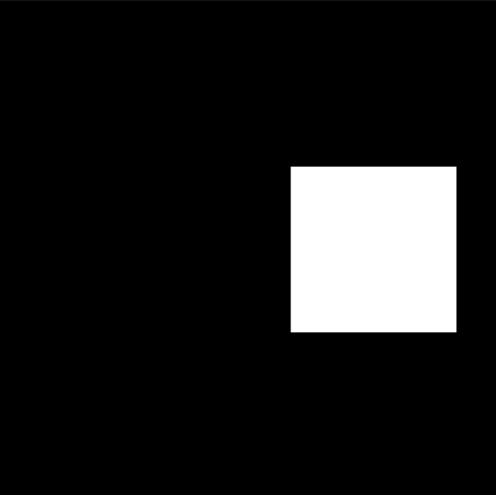
* 在画三角形前调用glFrontFace(GL\_CCW)而不是在画矩形前调用

glFrontFace是OpenGL中用于设置多边形正面的顺时针或逆时针方向的函数，指定了多边形正面的方向。glEnable(GL\_CULL\_FACE)已经声明接下来的渲染中只会渲染正面，glFrontFace让OpenGL知道哪一面是多边形的正面以便正确渲染。

GL\_CCW 表示逆时针方向，即按照顺序连接多边形顶点时，以逆时针方向为多边形的正面。

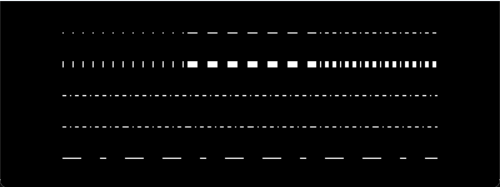
原代码中在调用glBegin(GL\_TRIANGLES)来绘制多边形的时候，默认为按照顺时针glFrontFace(GL\_CW)来确定正反面，在设置完状态glBegin(GL\_TRIANGLES)后，定义三角形的顶点（按照顺时针方向），glEnd()结束该绘制状态，这样渲染中就可以正常显示三角形。而对于glBegin(GL\_QUADS)设置绘制方形前，先执行了glFrontFace(GL\_CW)语句，这样就需要逆时针来定义四个顶点，方形才能被正常显示。

在进行了更改后（在画三角形前调用glFrontFace(GL\_CCW)），得到的结果如下所示，屏幕中只正常绘制了方形，而三角形不见了，这是因为三角形是顺时针方向定义的顶点，在glFrontFace(GL\_CCW)状态下，它的正面在屏幕另一边，三角形的背面不被绘制，我们不能在屏幕上看到它。



1. Line.cpp

代码运行结果如图所示。



该代码实现了在display函数中调用glBegin(GL\_LINE\_STRIP)绘制线条显示在屏幕上，并在窗口的glutDisplayFunc函数中调用display函数。

其中线条的绘制方法如下：glEnable(GL\_LINE\_STIPPLE)启用线条的点画功能，通过后续的 glLineStipple 函数来指定点画的具体样式，drawOneLine 函数是一个宏，在实际编译时展开为使用 glBegin(GL\_LINES) 开始绘制线条，然后使用 glVertex2f 指定两个点的坐标，最后使用 glEnd() 结束绘制。

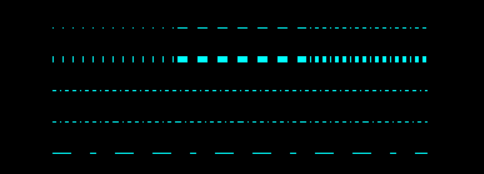
举例：glLineStipple(1, 0x1C47)设置线条的点画样式，第一个参数 1 指定了该样式重复使用的次数，第二个参数用于设置点画的模式，0x1C47 模式指定了dash-dot-dash的效果，这种模式只使用一次。

另外，下面是对代码中部分其他函数进行修改的例子。

* glColor3f(0.0, 1.0, 1.0);

原代码为glColor3f(1.0, 1.0, 1.0);

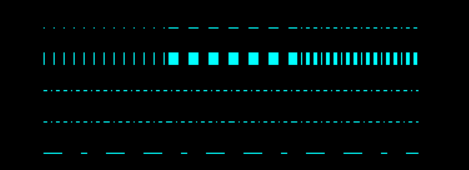
glColor3f(r, g, b) 函数用于设置绘制颜色，这里相当于是设定了一种状态，影响接下来在调用绘制函数时使用的颜色，例如 glVertex 来绘制线条或图形时会应用这个颜色。将颜色从白色改为青色，如下图所示。



* glLineWidth(10.0);

原代码为glLineWidth(5.0);

glLineWidth(width) 函数用于设置线条的宽度，参数 width 以像素为单位。原始代码中的 glLineWidth(5.0) 将线条宽度设置为 5.0 像素，而修改后的 glLineWidth(10.0) 将线条宽度设置为 10.0 像素。下图中为更改后的结果，第二行的线条看起来更粗。



结论分析

1. GLUT的应用：

GLUT库简化了OpenGL开发，能够在不同操作系统上实现一致的窗口和事件管理，它提供了方便的窗口创建和管理功能，能够捕获用户的输入事件，实现交互式操作。GLUT可以用于定义渲染窗口的显示模式，以及指定绘制回调函数，用于渲染图形场景。

1. OpenGL应用

OpenGL实现了：顶点绘制（glVertex绘制点），几何图元绘制（glBegin和glEnd包裹的代码块来绘制多边形），矩阵变换（glTranslate、glRotate、glScale进行对象的平移、旋转和缩放），投影变换（gluPerspective、glOrtho）等方法，

并且通过缓存来实现pipeline操作，设置状态来定义绘制图元的材质等。这些方法和功能为OpenGL提供了丰富的绘图工具和控制手段。