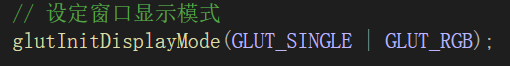
## 初始化glut

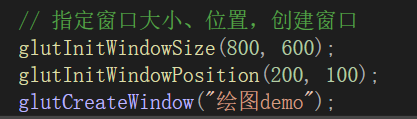


初始化glut库，并且处理命令行启动变量argc与argv。

## 配置和创建窗口



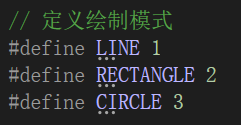
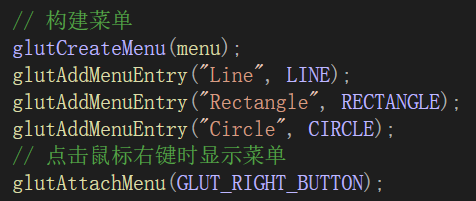
设置窗口显示模式为单通道RGB模式。



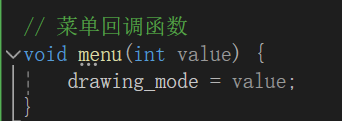
指定窗口大小，长为800、宽为600，窗口位置为（200,100），创建窗口，名称为“绘图demo”。

## 注册事件处理函数

#### 构建菜单

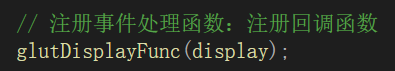


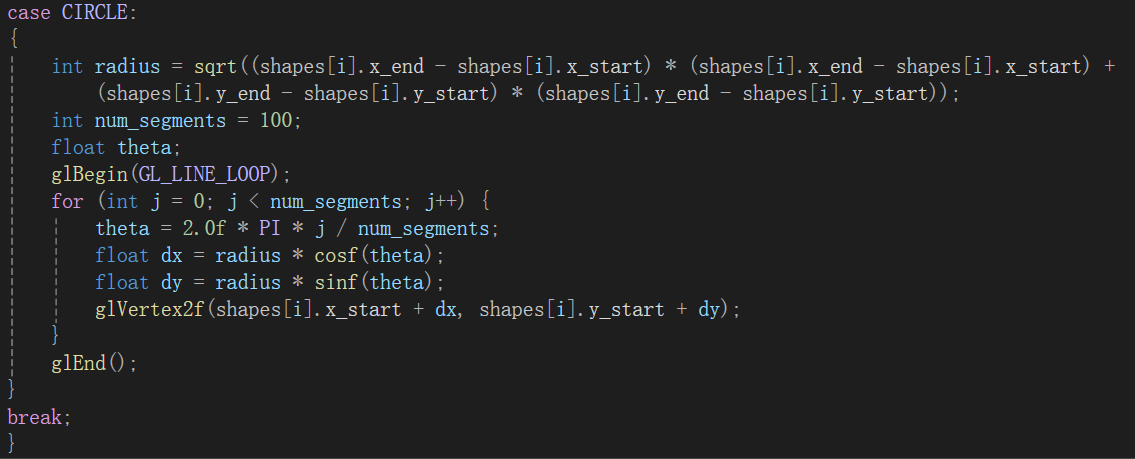
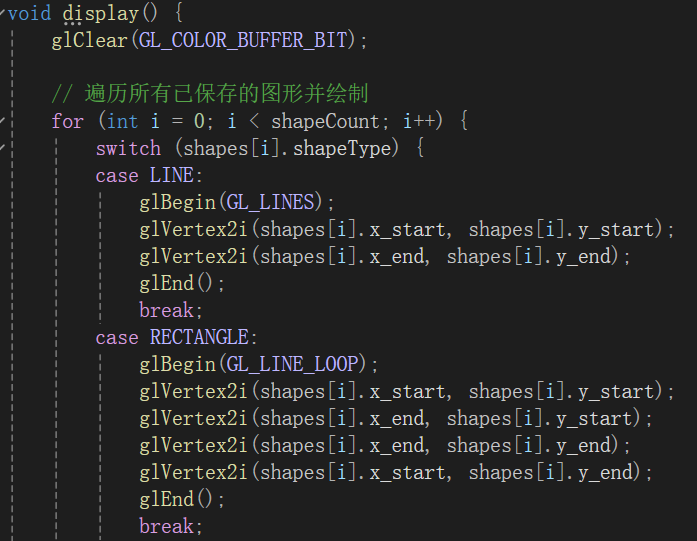
构建菜单，菜单中共有三个选项“Line”、“Rectangle”、“Circle”，并对每个选项进行赋值，值为先前宏定义的内容。将菜单与鼠标右键绑定，点击鼠标右键时显示菜单。



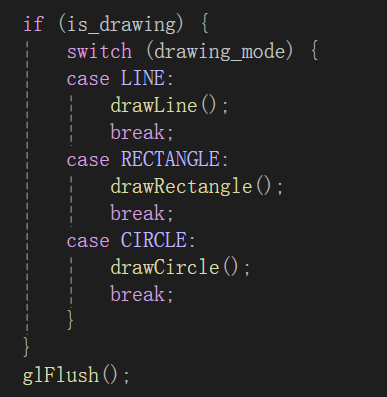
注册菜单回调函数，在回调函数中修改绘画模式的值为点击选项所传入的值。

#### 注册回调函数





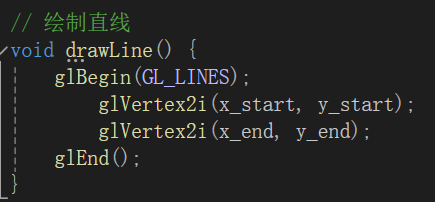
为了使每次绘制的图形保留在屏幕上，display函数会先对存储已绘制图形的数组shape进行遍历，绘制出之前绘制的图形。



绘制当前图形时，在display函数中使用switch case语句调用不同图形的绘制函数。

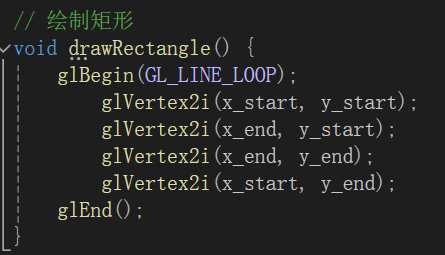
#### 编写不同图形绘制函数

###### 绘制直线



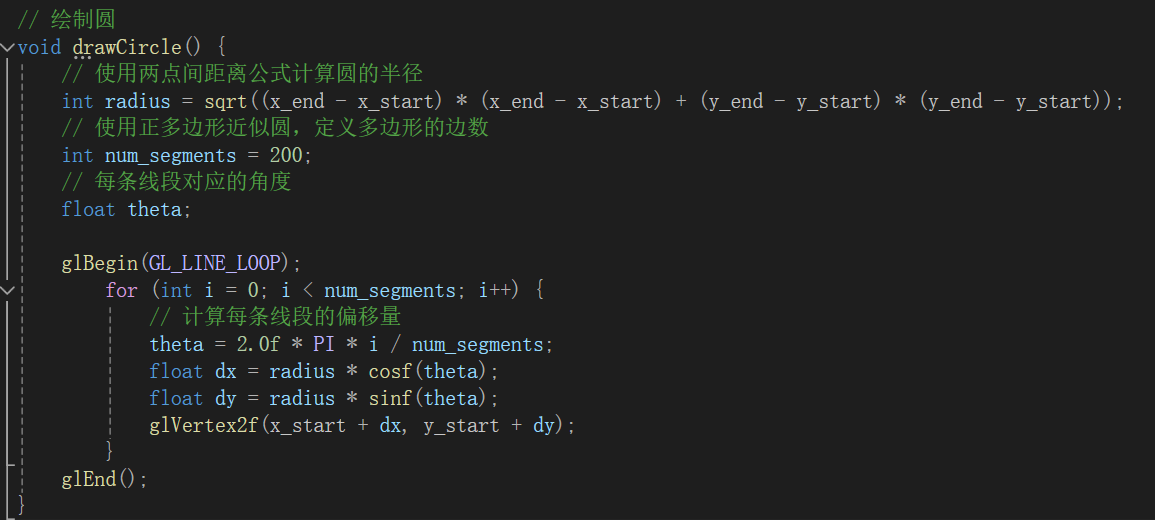
简单地使用两个顶点 (x\_start, y\_start) 和 (x\_end, y\_end) 通过 glBegin(GL\_LINES) 绘制一条线段。

###### 绘制矩形



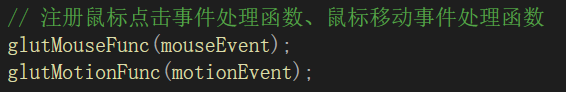
通过四个顶点依次绘制矩形的四条边。使用 GL\_LINE\_LOOP 来确保绘制的是闭合的线条。

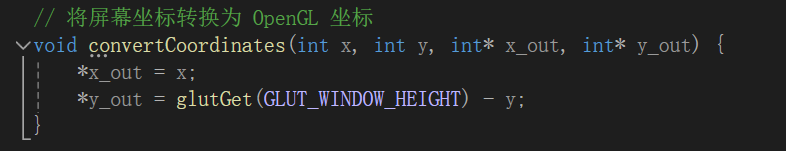
###### 绘制圆形



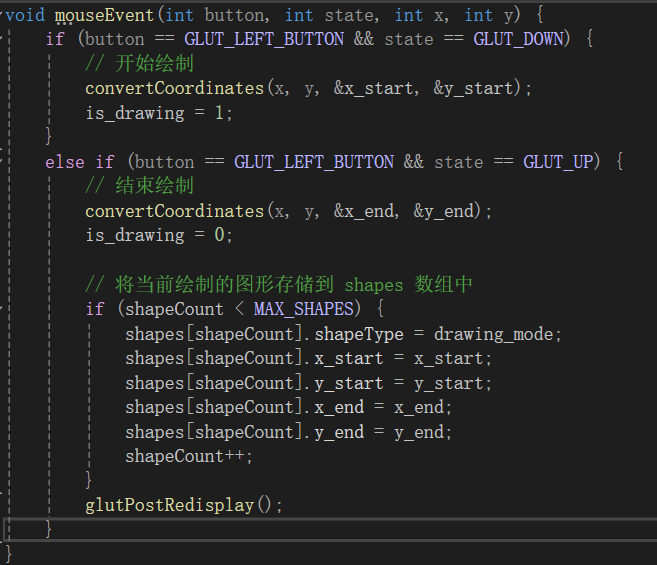
先使用两点间距离公式计算要绘制圆的半径，采用正多边形逼近圆的方式绘制圆，代码中使用正200边形逼近圆，将圆的周长分成 200 段，每段对应一个角度 theta，使用三角函数 cosf() 和 sinf() 分别计算每段对应的 x 和 y 偏移量。最后通过 glVertex2f() 函数，将计算出来的每个顶点坐标添加到 OpenGL 的绘图管线中。在 glBegin(GL\_LINE\_LOOP) 和 glEnd() 之间的所有点最终会连接成一个闭合的多边形，逼近一个圆形。

#### 注册鼠标相关事件函数

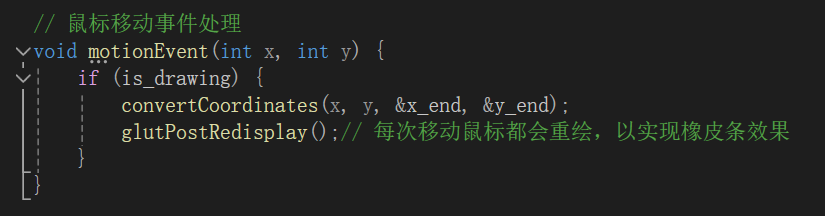




需要注意，由于屏幕坐标与OpenGL坐标的纵坐标不一致，需要将屏幕坐标转化为OpenGL坐标。

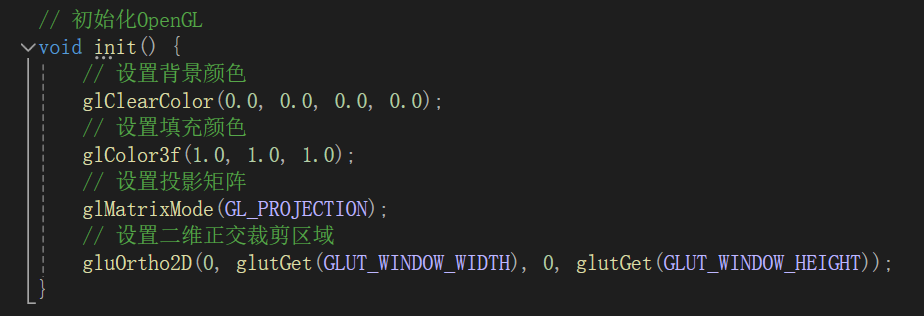


当鼠标左键被按下时，获取起始点的OpenGL坐标，并将绘制状态置为1，开始绘画。当鼠标左键被抬起时，获取结束点的OpenGL坐标，并将绘制状态置为0，结束绘画并调用glutPostRedisplay() 函数绘制最终的图形，同时将当前绘制的图形保存到shape数组中。



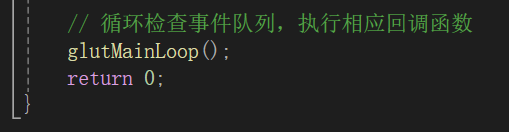
当鼠标发生移动时，更新当前鼠标的位置，并使用 glutPostRedisplay() 函数重绘屏幕，以实现橡皮条效果。

## 初始化OpenGL



设置背景颜色、填充颜色、投影矩阵与二维正交裁剪区域，初始化OpenGL。

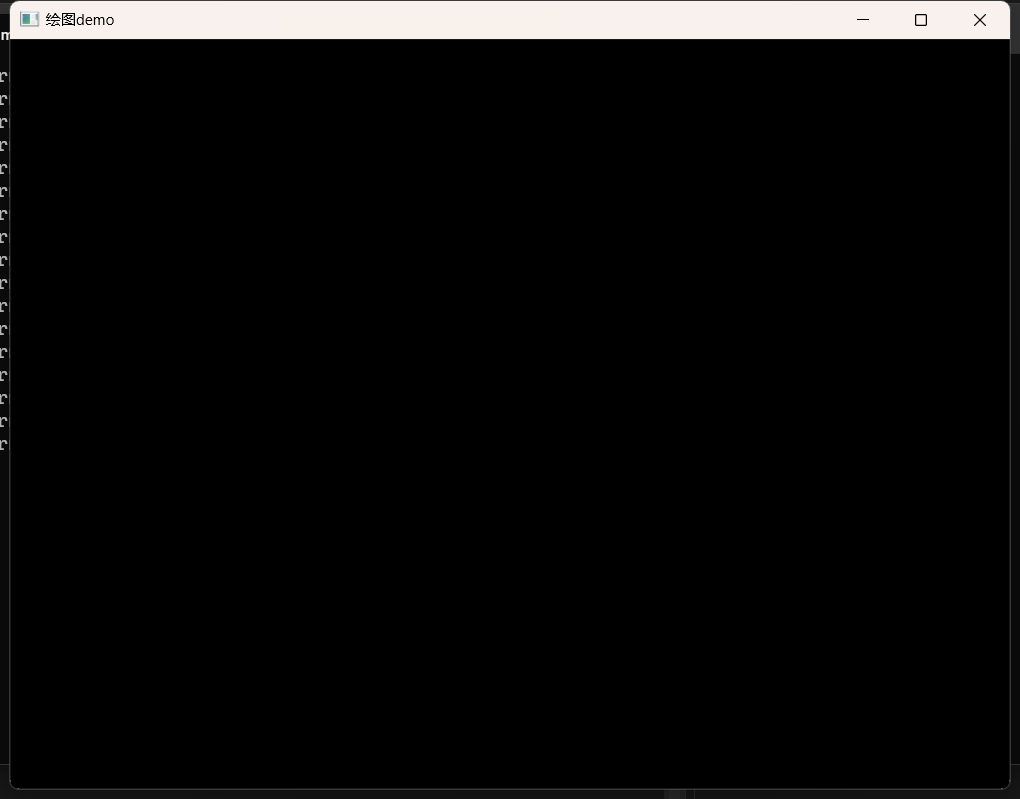
## 进入事件处理循环



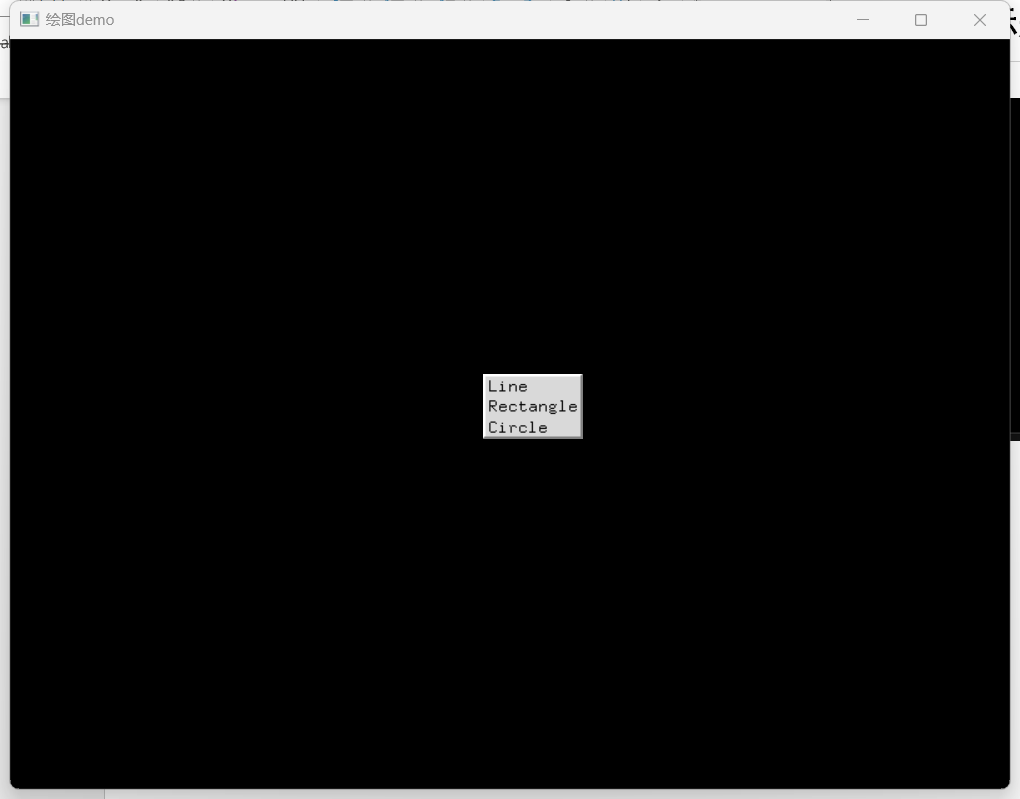
循环接收事件，并根据事件队列中的事件执行相应回调函数。

## 代码执行结果

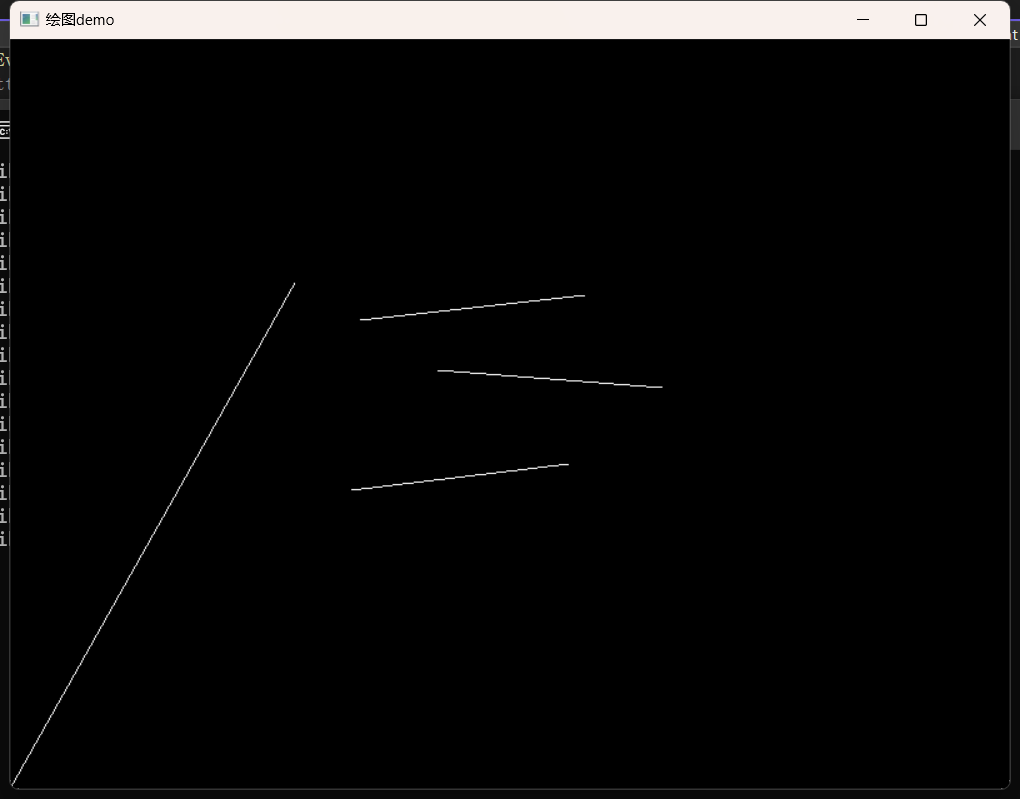
进行调试后，得到以下窗口“绘图demo”：



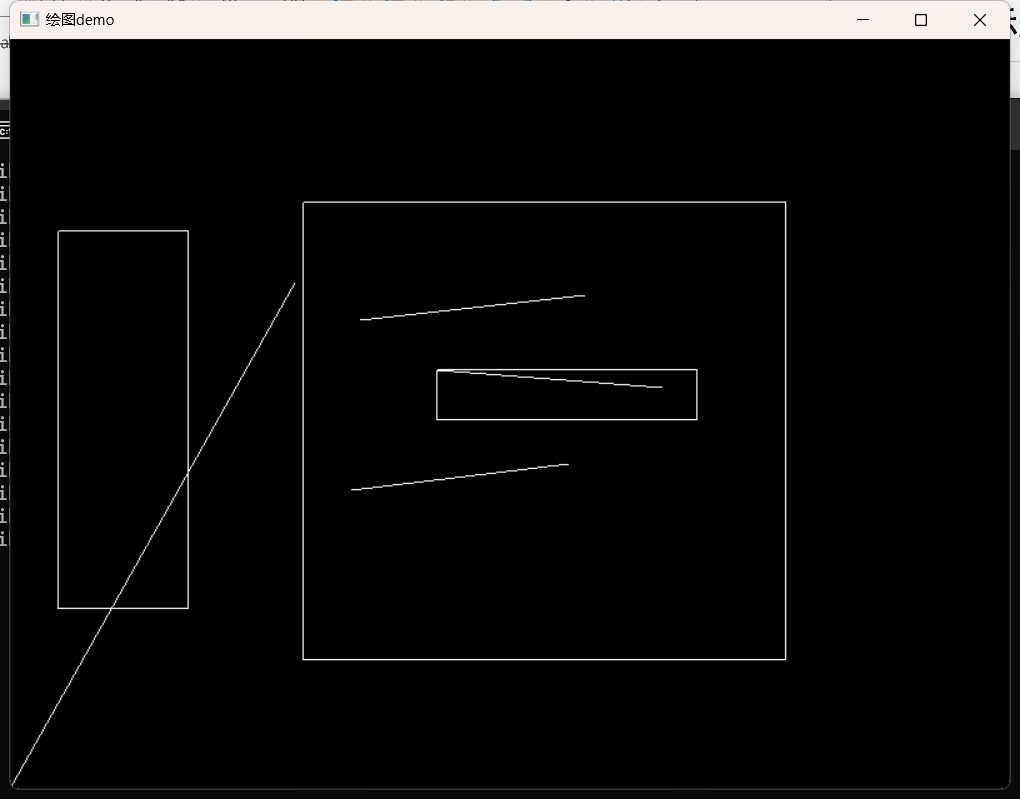
在窗口中点击鼠标右键，呼出以下菜单：



选中“Line”选项后，绘制直线：



选中“Rectangle”选项，绘制矩形：



选中“Circle”选项，绘制圆形：

