学号：E51814014 姓名：吴振龙 专业：数字媒体技术

实验3 交互与动画I

【实验目的】

1.掌握基本交互式程序的编程方法。

2.掌握基本动画程序的编程方法。

【实验题目】

1.阅读squareMouse.c, 回答下面的问题：

（1）glFlush()函数和glClear(GL\_COLOR\_BUFFER\_BIT)函数的作用分别是什么？（可将这两个函数注释掉，和注释前的结果对比）

答：glFlush()函数：清空缓存区，将gl指令送往硬件执行，去掉之后将无法绘制图形；

glClear(GL\_COLOR\_BUFFER\_BIT)：将屏幕清除为当前缓存颜色，可视为设置背景颜色

（2）修改squareMouse.c，分别实现如下功能：

a. 通过利用移动回调函数可以在不释放鼠标按钮的情况下，连续画一系列正方形；

函数：

|  |
| --- |
| def mouse\_move(x, y):        drawSquare(x, y) |

回调函数：

|  |
| --- |
| glutMotionFunc(mouse\_move) |

b. 应用被动移动回调函数，可以不用按鼠标按钮就可以连续画正方形；

改变回调函数为：

|  |
| --- |
| glutPassiveMotionFunc(mouse\_move) |

c. 按下Alt+c或Alt+C时，终止程序。

处理函数：

|  |
| --- |
| *def* keyboard(*key*, *x*, *y*):        if GLUT\_ACTIVE\_ALT == glutGetModifiers():          # ord()转换成字符的二进制          if ord(key) == ord('c'):              exit(0) |

键盘回调函数：

|  |
| --- |
| glutKeyboardFunc(keyboard) |

2. 编写一个程序，实现如下的功能：连续两次单击鼠标左键，以两次单击的位置作为矩形的对角线来绘制一个矩形，且该矩形各边与屏幕对齐。鼠标右键用于程序的退出。

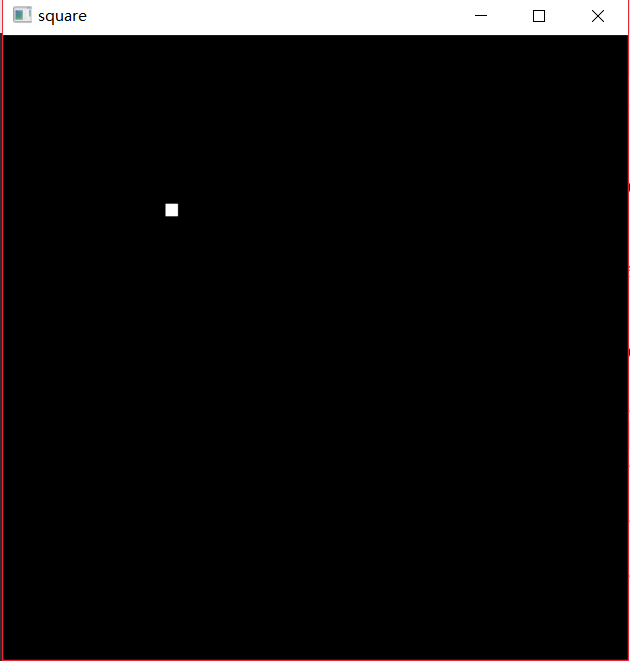
（1）将绘制矩形的函数放在鼠标回调函数中完成。

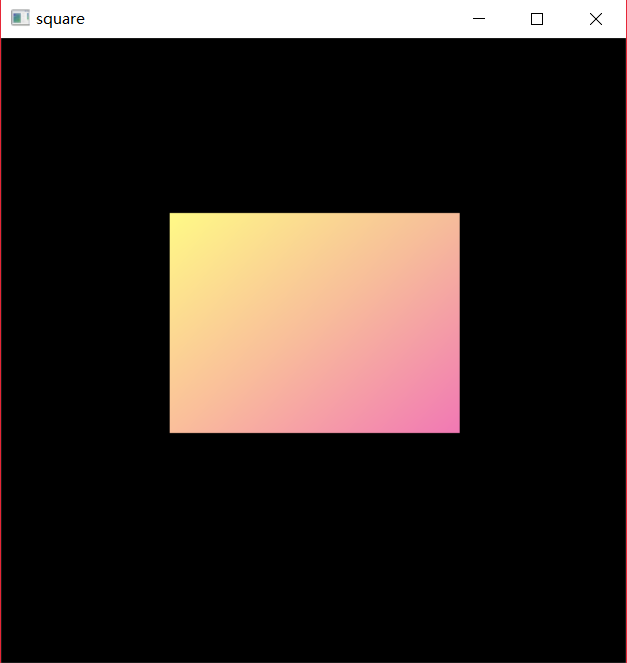
（2）修改（1）中的程序，将绘制矩形的函数放在显示回调函数中完成。鼠标回调函数用于状态的修改，并调用显示回调函数（利用glutPostRedisplay()）。

算法描述：

通过定义一个全局列表（相当于C语言的数组）保存鼠标点击的位置坐标x, y，当保存两对坐标后，绘制矩形并清空列表，进入循环。

(1)演示：





(1)源代码见附录一

(2) 将调用绘矩形函数改成：

|  |
| --- |
| glutPostRedisplay(drawSquare) |

附录一：

from OpenGL.GL import \*

from OpenGL.GLUT import \*

wh = 500

ww = 500

size = 5.0

vexs = []

*def* clear():

    glClearColor(0, 0, 0, 0)

    glClear(GL\_COLOR\_BUFFER\_BIT)

*def* drawSquare():

    # 根据vexs数组绘制矩形

    vexs[1] = wh - vexs[1]

    vexs[3] = wh - vexs[3]

    glBegin(GL\_QUADS)

    glColor3ub(255, 250, 134)

    glVertex2f(vexs[0], vexs[1])

    glColor3ub(250, 190, 155)

    glVertex2f(vexs[0], vexs[3])

    glColor3ub(241, 120, 180)

    glVertex2f(vexs[2], vexs[3])

    glColor3ub(246, 188, 154)

    glVertex2f(vexs[2], vexs[1])

    glEnd()

    glFlush()

*def* myReshape(*w*, *h*):

    glMatrixMode(GL\_PROJECTION)

    glLoadIdentity()

    glOrtho(0.0, w, 0.0, h, -1.0, 1.0)

    glMatrixMode(GL\_MODELVIEW)

    glLoadIdentity()

    glViewport(0,0,w, h)

    glClearColor (0.0, 0.0, 0.0, 1.0)

    glClear(GL\_COLOR\_BUFFER\_BIT)

    glFlush()

*def* myinit():

    #  /\* set clear color to black \*/

    glClearColor(0.0, 0.0, 0.0, 1.0)

*def* draw\_point(*x*, *y*):

    # 绘制一个点

    y = wh - y

    glPointSize(10)

    glBegin(GL\_POINTS)

    glColor3f(1, 1, 1)

    glVertex2f(x, y)

    glEnd()

    glFlush()

*def* mouse(*btn*, *state*, *x*, *y*):

    if btn==GLUT\_RIGHT\_BUTTON and state==GLUT\_DOWN:

        exit(0)

    if btn == GLUT\_LEFT\_BUTTON and state == GLUT\_UP:

        ''' 当单击鼠标左键，先绘制一个点，再将鼠标位置的坐标添加到vexs列表中 '''

        draw\_point(x, y)

        vexs.append(x)

        vexs.append(y)

        if len(vexs) == 4:

            ''' 当列表的长度为4时，代表这时已保存了两对点，清空屏幕再根据两对点的坐标绘制矩形 '''

            clear()

            drawSquare()

            vexs.clear()

*def* keyboard(*key*, *x*, *y*):

    if GLUT\_ACTIVE\_ALT == glutGetModifiers():

        # ord()转换成字符的ASCII码

        if ord(key) == ord('c'):

            exit(0)

*def* display():

    pass

*def* main():

    glutInit()

    glutInitDisplayMode(GLUT\_SINGLE or GLUT\_RGB)

    glutInitWindowSize(500, 500)

    glutInitWindowPosition(0, 0)

    glutCreateWindow("square")

    glutDisplayFunc(display)

    glutReshapeFunc(myReshape)

    glutMouseFunc(mouse)

    # 实现Alt + c关闭程序

    glutKeyboardFunc(keyboard)

    if len(vexs) == 4:

        drawSquare()

    myinit()

    glutMainLoop()

main()