

计算机图形学

Computer Graphics

陈中贵

chenzhonggui@xmu.edu.cn

<http://graphics.xmu.edu.cn/~zgchen>

Graphics@XMU



第三章 第二节

回调函数的应用

主要内容

- 事件驱动编程
 - 鼠标
 - 键盘
 - 窗口改变
- 弹出式菜单

使用定位设备

- 一般把定位设备假定成鼠标，不过定位设备也可以是跟踪球或数据板
- 有两类事件和定位设备相关联
 - 鼠标事件(mouse event): 当鼠标的的一个按键被按下或释放。
 - 返回的信息包括: 生成这个事件的鼠标按键、事件之后按键的状态（释放或按下）、窗口坐标系（原点位于左上角）中鼠标光标位置
 - 移动事件: 如果鼠标在某个按键被按下时移动，就发生了移动事件(move event); 如果移动时没有按键被按下，这个事件称为被动移动事件(passive move event)

鼠标回调函数

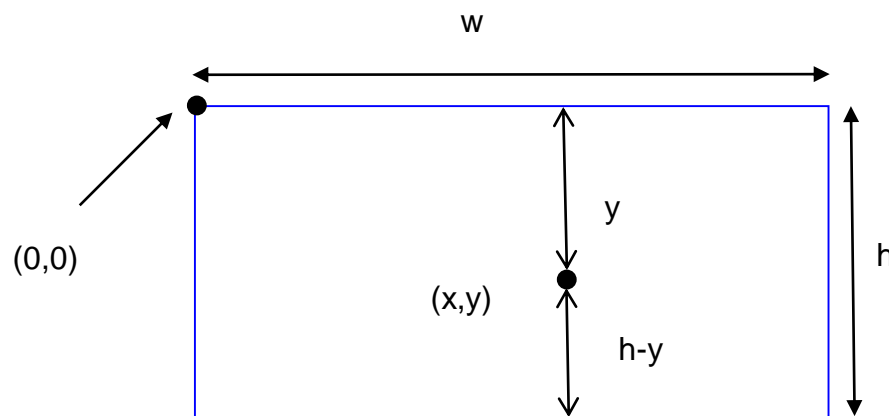
glutMouseFunc (mouse)

void mouse(int button, int state, int x, int y)

- 其中button的值可能是
GLUT_LEFT_BUTTON,
GLUT_MIDDLE_BUTTON,
GLUT_RIGHT_BUTTON
表示哪个按钮导致了事件发生
- state表示相应按钮的状态:
GLUT_UP, GLUT_DOWN
- x, y表示在窗口中的位置

定位

- 在屏幕上的位置通常是以像素为单位的，原点在左上角
 - 因为显示器自顶向下刷新显示内容
- 在OpenGL中使用的世界坐标系，其原点在左下角
- 在这个坐标系中的y坐标需要从窗口高度中减去回调函数返回的y值： $y = h - y$



获取窗口尺寸

- 为了完成y坐标的转换，需要知道窗口的尺寸
 - 在程序执行过程中，高度可能发生改变
 - 需要利用一个全局变量跟踪其变化
 - 新高度值返回给形状改变回调函数(见后)
 - 也可以用查询函数`glGetIntegerv()`和`glGetFloatv()` 获取，因为高度是状态的一部分

结束程序

- 在以前的程序中没有办法通过OpenGL结束当前程序
- 可以利用简单的鼠标回调函数做到这一点

```
void mouse(int btn, int state, int x, int y)
{
    if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
        exit(0);
}
```


鼠标位置的应用

- 在要构造的例子中，每单击一次鼠标左按钮，就会在当前鼠标位置处画一个小方框
- 在这个例子中并没有用到显示回调函数，但是由于GLUT要求必须有一个显示回调函数，因此要定义一个空函数

```
display() {}
```

在指针处画方框

```
void mouse(int btn, int state, int x, int y) {
    if(btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) exit(0);
    if(btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
        drawSquare(x, y);
}

void drawSquare(GLint x, GLint y) {
    y = h-y; //转化y坐标
    glColor3ub( (char)rand()%256, (char) rand()%256,
(char)rand()%256); // 随机颜色
    glBegin(GL_POLYGON);
        glVertex2f(x - size, y - size);
        glVertex2f(x + size, y - size);
        glVertex2f(x + size, y + size);
        glVertex2f(x - size, y + size);
    glEnd();
}
```

鼠标移动回调函数的应用

- 通过利用移动回调函数可以在不释放鼠标按钮的情况下，连续画一系列方框 (square.c)
 - **glutMotionFunc (drawSquare)**
 - 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方框
 - **glutPassiveMotionFunc (drawSquare)**
-
- `void glutMotionFunc(void (*f)(int x,int y))`
 - `void glutPassiveMotionFunc(void (*f)(int x,int y))`

键盘的使用

当鼠标位于窗口内，并且键盘有某个键被按下或释放，就会产生键盘事件

glutKeyboardFunc (keyboard)

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

- 返回键盘上被按下键的ASCII码和鼠标位置
- 注意在GLUT中并不把释放键做为一个事件

```
void keyboard(unsigned char key, int x, int y)
{
    // 按下Q、q或ESC键时，终止程序
    if(key == 'Q' || key == 'q' || key == '\27')
        exit(0);
}
```

特殊按键

- GLUT在glut.h中定义了特殊按键:

- 功能键1: GLUT_KEY_F1
- 向上方向键: GLUT_KEY_UP

```
void glutSpecialFunc(void  
(*func)(int key, int x, int y))
```

- 回调函数内

```
if(key == GLUT_KEY_F1) .....
```

```
if(key == GLUT_KEY_UP) .....
```

修饰键 – Ctrl/Alt/Shift

int glutGetModifiers()

- 如果在鼠标或键盘事件产生时修饰键被按下，该函数返回 GLUT_ACTIVE_SHIFT, GLUT_ACTIVE_CTRL, GLUT_ACTIVE_ALT的逻辑与结果

- 可以利用单键或双键鼠标模拟三键鼠标

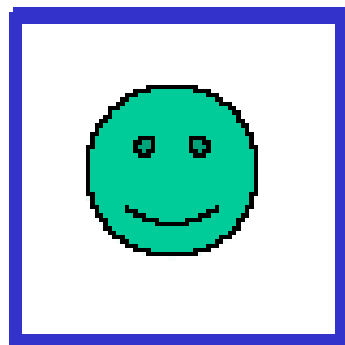
- 让Ctrl+c或Ctrl+C终止程序，在键盘回调函数中

```
if (glutGetModifiers() == GLUT_ACTIVE_CTRL) &&  
    (key == 'c' || key == 'C'))  
    exit(0);
```

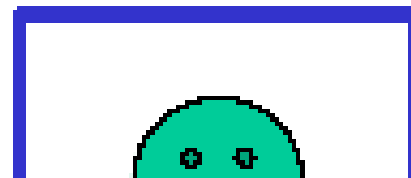
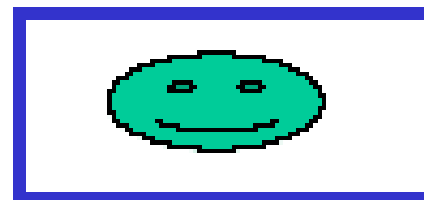
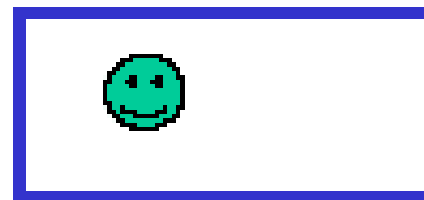
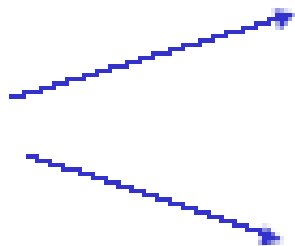
窗口形状的改变

- 通过拖动窗口的角点可以改变窗口的形状和尺寸
- 那么其中的显示内容该如何处理？
 - 必须由应用程序重新绘制
 - 有两种可能性
 - 显示原来内容的一部分
 - 通过强迫适应新窗口来显示所有内容
 - 可能会改变了显示长宽比率

形状改变的可能性



初始图像



改变后的图像

形状改变回调函数

glutReshapeFunc (reshape)

void reshape(int w, int h)

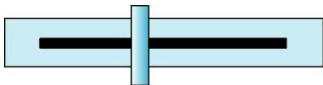
- 返回新窗口的宽度与高度（单位：像素）
- 回调函数执行后自动发送刷新显示事件，触发显示回调
- GLUT有一个缺省的形状改变的回调函数，调用 `glViewport(0,0,w,h)` 把视口设置为新窗口
- 这个回调函数是放置照相机函数的恰当地方，因为当窗口首次创建时就会调用它。
 - 当窗口形状尺寸发生改变时，可在回调函数里对观察条件进行修改

例子

通过保持视口和世界窗口的长宽比一样，使得物体不变形

```
void reshape(int w, int h) {  
    glViewport(0, 0, w, h); // 设置视口为整个窗口  
    glMatrixMode(GL_PROJECTION); // 调整裁剪窗口  
    glLoadIdentity();  
    if(w<=h) // 宽小于高，裁剪矩形宽度置为4  
        gluOrtho2D(-2.0, 2.0, -2.0*(GLfloat)h/(GLfloat)w,  
                    2.0*(GLfloat)h/(GLfloat)w);  
    else // 高小于宽，裁剪矩形高度置为4  
        gluOrtho2D(-2.0*(GLfloat)w/(GLfloat)h,  
                    2.0*(GLfloat)w/(GLfloat)h, -2.0, 2.0);  
    glMatrixMode(GL_MODELVIEW); /*return to modelview mode*/  
}
```

构件(widgets)

- 许多窗口系统提供了一个工具包或者一组库函数用来建立用户界面，界面中用到了一些特殊类型的窗口，称为构件(widgets)
- 构件集中包含下述工具：
 - 菜单
 - 滑动条 
 - 对话框
 - 文本输入框
- 但上述构件通常都是与平台相关的
- GLUT只提供了包含菜单在内的很少几个构件

菜单

- GLUT支持弹出式菜单
 - 可以有子菜单
- 创建弹出式菜单的三个步骤
 - 定义菜单内各条目
 - 定义每个菜单项的行为
 - 如果条目被选择执行的操作
 - 把菜单连接到鼠标按钮上

菜单函数

- **int glutCreateMenu(void (*f)(int value))**
 - 创建一个使用回调函数f()的菜单，并返回菜单的整数标识符
- **void glutAddMenuEntry(char *name, int value)**
 - 为当前菜单增加一个名为name的菜单项；value值在选中时返回给菜单回调函数
- **void glutAttachMenu(int button)**
 - 将当前菜单关联到鼠标按钮button上 (GLUT_RIGHT_BUTTON、GLUT_MIDDLE_BUTTON、GLUT_LEFT_BUTTON)

定义一个简单菜单

- 在main()函数中

```
menu_id = glutCreateMenu(mymenu);  
glutAddMenuEntry("Clear Screen", 1);  
glutAddMenuEntry("Exit", 2);  
glutAttachMenu(GLUT_RIGHT_BUTTON);
```

当按下右按钮时，就
会出现的条目

标识符

Clear Screen
Exit

菜单的行为


- 菜单回调函数

```
void mymenu(int value) {  
    if(value==1) glClear(GL_COLOR_BUFFER_BIT);  
    if(value==2) exit(0);  
}
```

- 添加子菜单

```
void glutAddSubMenu(char *submenu_name,  
    int submenu_id)
```

- 增加一个子菜单项 `submenu_name` 作为当前菜单的一项，子菜单创建时返回的标识符为 `submenu_id`
- 必须先创建子菜单

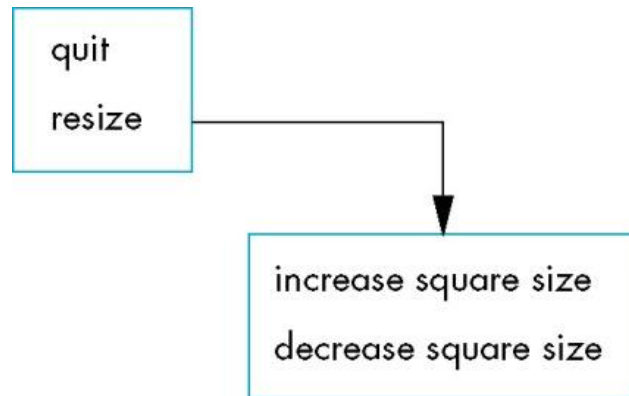


父菜单中一项

子菜单

在main()函数中创建一个层级菜单

```
int sub_menu;  
  
sub_menu = glutCreateMenu(processSizeMenu);  
glutAddMenuEntry("increase square size",2);  
glutAddMenuEntry("decrease square size",3);  
  
glutCreateMenu(processTopMenu);  
glutAddMenuEntry("quit",1);  
glutAddSubMenu("resize",sub_menu);  
glutAttachMenu(GLUT_RIGHT_BUTTON);
```



子窗口与多窗口

- **int glutCreateWindow(char *name)**
 - 创建一个顶层窗口name，并为其返回一个整数标识符
- **void glutDestroyWindow(int id)**
 - 销毁标识符为id的窗口
- **void glutSetWindow(int id)**
 - 把当前窗口设为标识符为id的窗口
- **int glutCreateSubWindow(int parent, int x, int y, int width, int height)**
 - 为parent窗口创建一个子窗口，返回子窗口的标识符。子窗口原点位于(x,y)，宽度为width，高度为height
- **void glutPostWindowRedisplay(int id)**
 - 通知标识符为id的窗口重新显示

GLUT中其它的函数

- 动态窗口
 - 在执行期间创建或关闭窗口
- 在执行期间改变回调函数
- 计时器
- 字体
 - glutBitmapCharacter
 - glutStrokeCharacter
- GLUT教程
<http://www.lighthouse3d.com/opengl/glut/>