

Introduction to Computer Graphics



Programming with OpenGL: Interaction: Review

Ming Zeng
Software School
Xiamen University

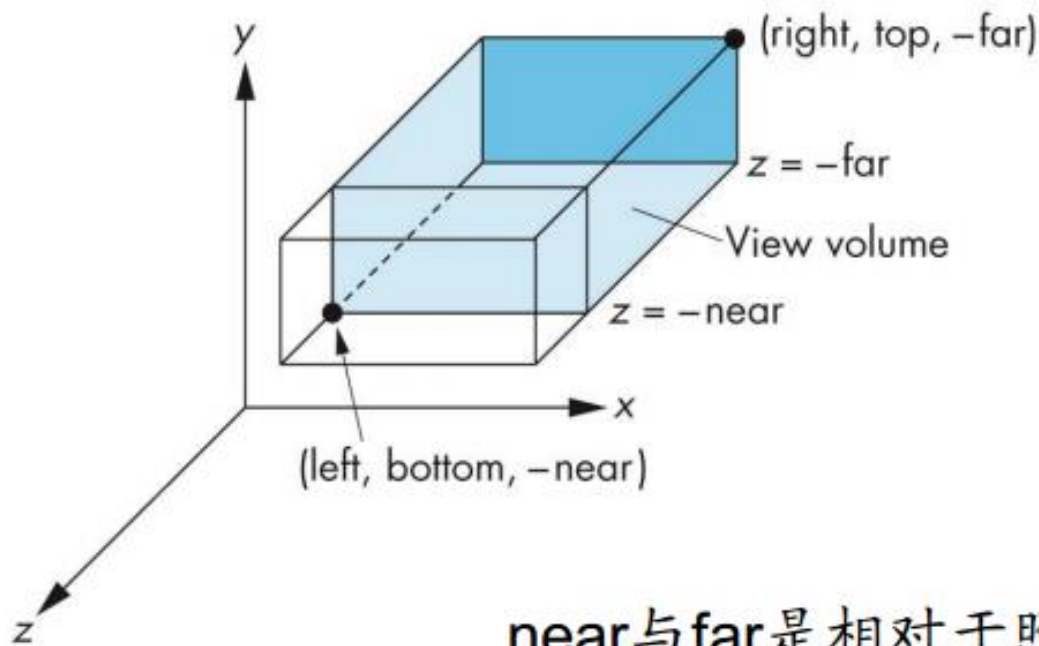
Contact Information:
zengming@xmu.edu.cn

本节内容

- glLookat和glOrtho的用法
- ply的使用
- 菜单
- 橡皮条技术（两种做法）
- 鼠标点击画矩形的原理
- reshape原理

gluLookAt和glOrtho的关系

1. gluLookAt影响的是模型视图矩阵
2. glOrtho影响的是投影矩阵（注意z坐标是反的，左手系）



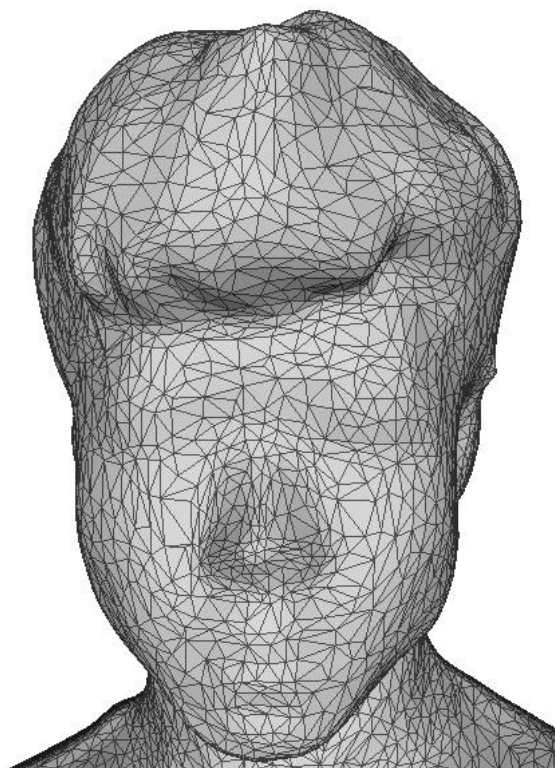
near与far是相对于照相机而言的

3D几何

- 基本几何图元
- 用基本几何图元构建



简
物
Pl
: i



: 球体竿
木
制

球体竿

```
ply
format ascii 1.0
comment VCGLIB generated
element vertex 37702
property float x
property float y
property float z
property float nx
property float ny
property float nz
element face 75404
property list uchar int vertex_indices
end_header
-109.85 -585.583 -913.65 -0.990883 0.0898327 -0.1004
-80.8779 -589.489 -965.09 -0.547469 0.203695 -0.811656
-97.4846 -599.833 -853.546 -0.516241 -0.175639 0.83824
-49.3773 -947.479 -786.993 0.475753 0.717751 0.508421
-97.3486 -588.504 -851.651 -0.529533 -0.183035 0.828307
-85.3551 -596.765 -847.027 -0.382764 -0.209772 0.899715
-30.7061 -959.997 -794.619 0.662137 0.512276 0.546944
-103.878 -957.321 -787.054 -0.565117 0.781235 0.265169
-91.6018 -948.06 -791.949 -0.465062 0.861006 0.205878
-83.9699 -946.093 -787.644 -0.324549 0.912479 0.249098
-81.7526 -943.729 -792.817 -0.296337 0.935378 0.193007
-27.2919 -593.142 -857.568 0.42749 -0.321394 0.84496
-26.6596 -586.995 -855.36 0.457365 -0.374257 0.80669
-45.2669 -594.426 -976.194 0.345885 0.188408 -0.919166
11.3998 -588.406 -906.555 0.992747 0.119602 -0.0121996
-11.6815 -590.988 -867.936 0.672804 -0.178577 0.717945
```

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)

定义一个简单的菜单

- 在main()函数中

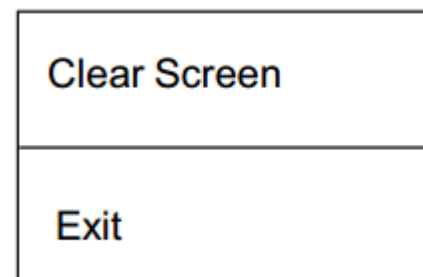
会作为参数返回给回调函数

mymenu

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

菜单回调函数

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



橡皮条功能

- 橡皮条功能
- 橡皮条功能的实现方法
 - display函数里重新绘制
 - 异或操作（回顾何为异或操作？）
- 像素逻辑操作：当前像素与原有像素的逻辑操作
- 逻辑操作的开启
 - glEnable(GL_COLOR_LOGIC_OP);
 - glLogicOp(GL_XOR);

异或操作实现橡皮条直线

- 利用第一次单击鼠标固定线段的一个端点，然后再利用鼠标移动的回调函数连续更新第二个端点
- 每次鼠标移动的时候，重新画一遍原来的直线从而删除它，然后再从固定的第一个端点到新的第二个端点之间画一条直线
- 最后切换回到正常的绘图模式并绘制直线

代码分析

- 详见代码 `rubberline.c`
- 注意事项：
 - `glEnable(GL_COLOR_LOGIC_OP);`
 - 每次移动，先用XOR把前一条线擦除，再画新的线

鼠标回调函数

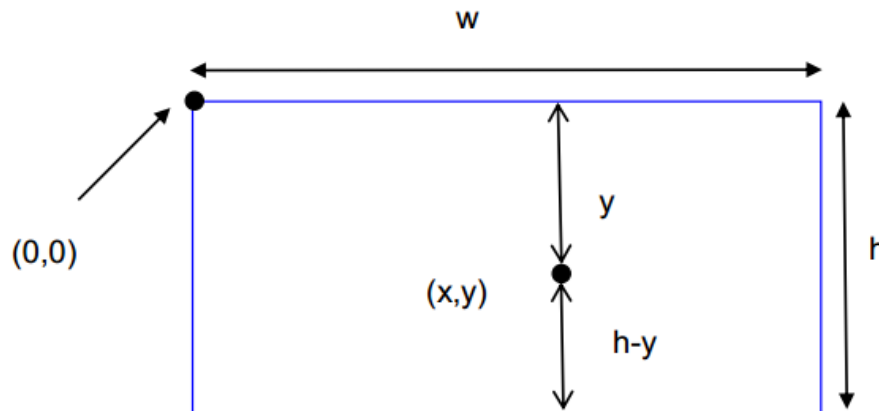
- **glutMouseFunc(mouse)**

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 (**以窗口左上角为原点的坐标**)

鼠标定位

- 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
- 例如：glGetIntegerv (GL_VIEWPORT, viewport)
- 将视口大小传入viewport数组中

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);  
}
```

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);  
}
```

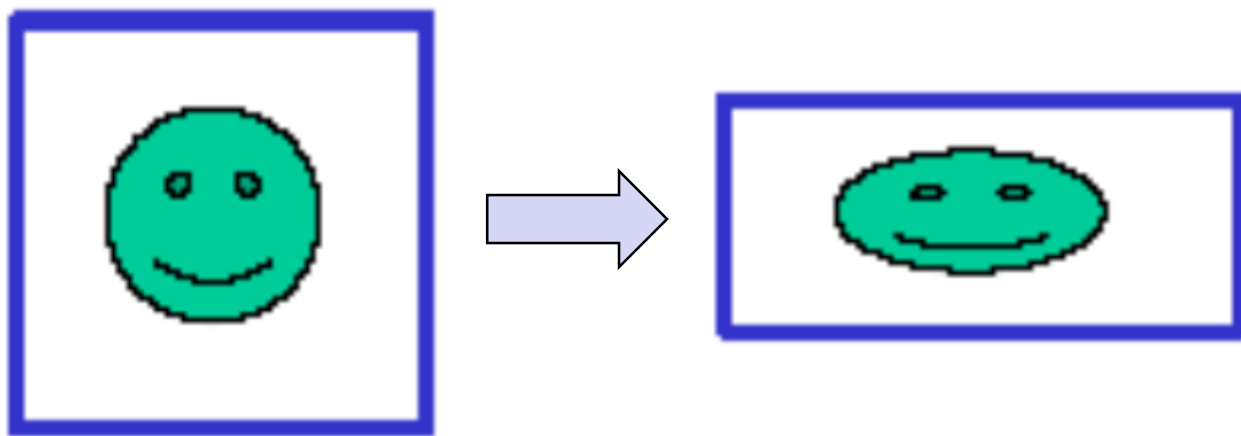
```
void drawSquare(GLint x, GLint y) {  
    y = h-y; //转化y坐标  
    glColor3ub( (char)rand()%256, (char) rand()%256,  
                (char)rand()%256); // 随机颜色  
    glBegin(GL_POLYGON);//为什么这么写  
        glVertex2f(x - size, y - size);  
        glVertex2f(x + size, y - size);  
        glVertex2f(x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

完整代码

- 见mouse.c
- 思考：
 - gluOrtho2D 参数应该怎么设置？
 - glViewport应该怎么设置？（为什么没设置）

改变窗口大小

- 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
 - 结果可能是



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
- 请仔细理解以下程序段

```
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*/  
}
```

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
- 例子：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);
```