浙江大学

计算机图形学作业报告

作业名称：	鼠标事件
姓　　名：	郭炅
学　　号：	3170105370
电子邮箱：	793881807@qq.com
联系电话：	18888923171
指导老师：	张宏鑫

2019 年　12 月 24 日

实验名字

一、 作业已实现的功能简述及运行简要说明

1. 已经实现功能

- 在场景中有拾取反馈的程序，使用鼠标点击物体，改变物体状态
- 在场景中自由漫步的程序，使用鼠标拖动控制视角，同时使用 WSADZC 控制前后左右上下移动

2. 运行简要说明

- VS2017 直接运行即可

二、 作业的开发与运行环境

- VS2017 Enterprise
- Windows10 professional
- Glut
- OpenGL 的 glm 数学运算库

三、 系统或算法的基本思路、原理、及流程或步骤等

1. 基本思路和原理

i. 鼠标拾取：通过点击获取屏幕坐标，与物体投射到屏幕上的坐标进行比对，确定是否选中，选中后，根据鼠标的姿态函数修改绘制的相关变量，改变物体的状态

ii. 自由漫游：注册按键函数获取相关的姿态指令，修改摄像机坐标，对于鼠标视角控制，通过鼠标姿态函数获取相关变量变化，通过矩阵运算进行视图修改，计算移动后的摄像机的相关参数

2. 流程和步骤

i. 注册相关函数

ii. 根据键盘函数获取相关变量变化，根据鼠标点击函数获取点击坐标，通过鼠标姿态函数获取移动矢量

iii. 修改相关绘制变量，根据鼠标的移动矢量计算新的摄像机参数，并进行修正

iv. 重新绘制

四、 具体如何实现，包括关键（伪）代码、主要用到函数与算法等

1. WASDZC 控制摄像机移动

```
glutKeyboardFunc(keyPress);
glutKeyboardUpFunc(keyUp);
```

- 注册键盘函数

```
case 'w':
    keyDown[UPMOVE] = true;
    break;
case 's':
    keyDown[DOWNMOVE] = true;
    break;
case 'a':
    keyDown[LEFTMOVE] = true;
    break;
case 'd':
    keyDown[RIGHTMOVE] = true;
```

- 修改摄像机的参数

2. 鼠标控制视角变化

```
    glutMotionFunc(mouseMove);
```

- 注册鼠标姿态获取函数

```
    vec3 r = normalize(cross(lookDir, up));
    up = normalize(cross(r, lookDir));

    mat4 rot = rotate(mat4(1.0f), radians(-dx*ratio), vec3(0,1,0));

    lookDir = normalize(vec3(rot*vec4(lookDir, 0)));
    up = normalize((vec3(rot*vec4(up, 0))));
    r = normalize((vec3(rot*vec4(r, 0))));

    rot = rotate(mat4(1.0f), radians(-dy*ratio), r);
```

```
        lookDir = normalize((vec3(rot*vec4(lookDir, 0))));
        up = normalize((vec3(rot*vec4(up, 0))));
```

- 获取未修改状况下的摄像机的三维矢量坐标：$\overrightarrow{lookDir}$，$\overrightarrow{up}$，$\vec{r}$
  - 当前的 lookDir 已知，任取向上的矢量坐标，使用叉积获取实际的右手矢量，再次将实际的右手矢量和 lookDir 进行叉积计算得到实际的向上矢量，由此，可以获取完整的摄像机的三向矢量
  - 根据获得的鼠标 X 向姿态位移，创建旋转矩阵，将旋转矩阵与摄像机的三维坐标进行点积获取修正后的三维矢量
  - 根据获得的鼠标 X 向姿态位移，创建旋转矩阵，将旋转矩阵与摄像机的三维坐标进行点积获取修正后的三维矢量
  - 最终计算结束后的三维矢量即为实际的摄像机的三维矢量

3. 鼠标点击拾取，并修改相关的参数

```
        glutMouseFunc(mousePress);
```

- 注册鼠标点击函数

```
        case GLUT_LEFT_BUTTON:{
           if(state==GLUT_DOWN){
               selectedIndex=-1;
               // float minDis=1000;
               vec2 windowSize = { glutGet(GLUT_WINDOW_WIDTH), glutGet(GLUT_WINDOW_HEIGHT) };
               float minDis = 1000;

               for(int i=0;i<objects.size();i++){
               float dis=length(objects[i].screenPos(projectionMat,viewMat)-vec2(x,y));

                   if (minDis > dis)
                   {
                       minDis = dis;
                       if (minDis < 20)
                       {
                           selectedIndex = i;
                       }
                   }
               }
           }
           break;
```

- 比较物体的中心坐标与鼠标点击处的坐标，判定是否选中了物体

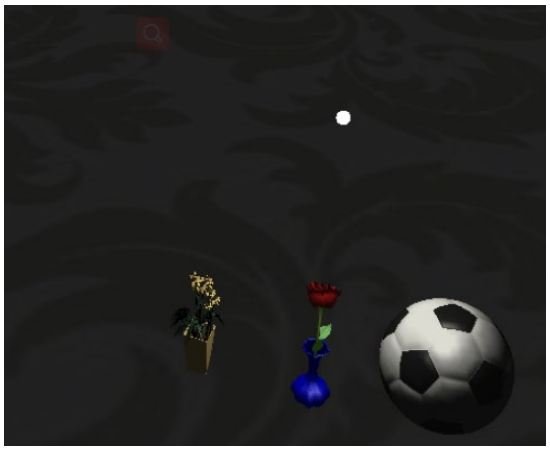
```

```

```
        vec4 lastPos = projectionMat*viewMat*vec4(objects[selectedIndex].position, 1);
        float w = lastPos.w;
        lastPos /= lastPos.w;
        lastPos = lastPos*0.5f + 0.5f;

        vec3 destPos = vec3(float(x) / glutGet(GLUT_WINDOW_WIDTH), 1 - float(y) / glutGet(GLUT
_WINDOW_HEIGHT), lastPos.z);
        destPos = destPos*2.0f - 1.0f;
        vec4 p = inverse(projectionMat*viewMat)*vec4(destPos*w, w);

        objects[selectedIndex].position = vec3(p);
```

- 对已经被选中的物体，创建相关的状态转移矩阵进行状态修正，此处的计算主要用于物体的移动

五、 实验结果与分析

- 初始状态



- 鼠标移动角度后

- 鼠标拾取足球并且选中进行移动

六、 结论与心得体会

- 学习到了如何使用矩阵运算的方式进行摄像机视角的转换和物体的状态的转换

七、 参考文献