计算机图形学实验报告
hierarchical modeling

课程名称：　　计算机图形学

实验名称：hierarchical modeling

学生学院：　　泰山学堂

学生班级：2017 级计算机取向

学生学号：　201705301350

学生姓名：　　宋建涛

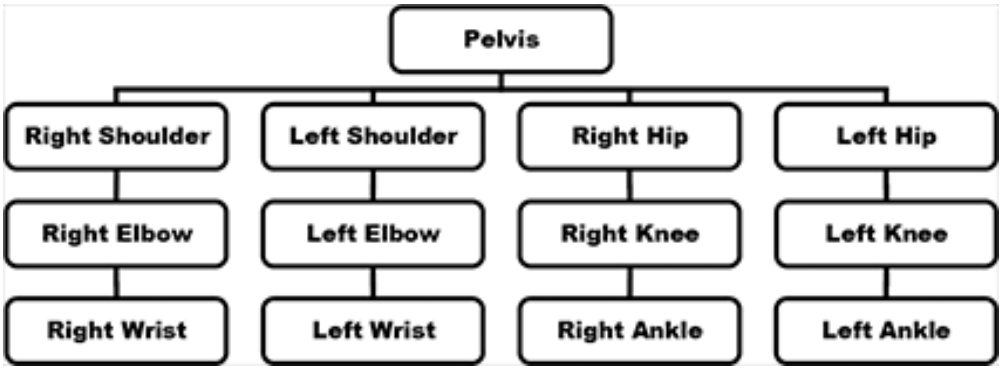
提交日期：　 2019.11.9

目录

实验目标 3
实验概览 3
画出初始状态下的火柴人 3
画出火柴人 4
实现 WASD 控制火柴人方向转变 8
实验代码结构 9
实验结果和收获 9
实验源代码 11

实验目标

本实验要求实现层次结构，具体实现是实现一个火柴人的走路动作，根据试验提示我们可以知道我们至少要在实验中画出火柴人的如下结构：



本实验的难点主要在于实现关节的逐步角度变换，下面我们在实验概览中介绍实验思路。

实验概览

根据实验要求，我们可以把本次大实验分为主要的几个部分：

1. 画出初始状态下的火柴人
2. 推导火柴人行走动作是各部位的旋转平移情况
3. 实现使用“wasd”控制火柴人面朝方向的改变

画出初始状态下的火柴人

我们需要建立一个类来存储火柴人的状态参数，在本实验中我们主要需要的参数包括各个肢体的旋转角度等，根据需求我们可以给出火柴人的结构定义：

```
class People {
public:
    float angleoflarma,//左大臂胳膊旋转角速度
        angleoflarmb, //左小臂胳膊旋转角速度
        angleofrarma, //右大臂胳膊旋转角速度
        angleofrarmb, //右小臂胳膊旋转角速度
        angleofllegb, //左大腿胳膊旋转角速度
        angleofrlegb, //左小腿胳膊旋转角速度
        angleofllega, //右大腿胳膊旋转角速度
        angleofrlega = 0;//右小腿胳膊旋转角速度
    float la, ra, ll, rl = 0;
    float a = 0.2,
        b = 0.1,
        angle = 0;
```

```
15.     const float angle1 = 0.2;
16.     const float angle2 = 0.1;
17.     void initAngle();
18.     void InitGL(int Width, int Height);
19.     void DrawBody();
20.     void DrawShoulder();
21.     void DrawHip();
22.     void DrawArmA();
23.     void DrawArmB();
24.     void DrawLegA();
25.     void DrawLegB();
26.     void DrawHead();
27.     void DrawNeck();
28. };
```

在数据之后我们给出了一些函数，这些函数可以帮助我们绘制出火柴人的身体结构。包括头、身体、肩膀、胯、双臂、双腿、脖子。为了简单绘制，我们将人体的各个部位简化为长方体，并用不同的颜色区分各个部分。

在本次试验中我们大量使用的 glut 函数有两个，下面我们简单介绍一下：

1. Void glTranslatef(GLfloat x, GLfloat xy GLfloat z)

   函数功能：沿 X 轴正方向平移 x 个单位，

   沿 Y 轴正方向平移 y 个单位，

   沿 Z 轴正方向平移 z 个单位

2. Void glRotataf(Glfloat angle,GLfloat x, GLfloat xy GLfloat z)

   函数功能：以点(0,0,0)到点(x,y,z)为轴，旋转 angle 角度；旋转的正方向为做(0,0,0)到(x,y,z)的向量，用右手握住这条向量，大拇指指向向量的正方向，四指环绕的方向就是旋转的方向；

这样的话，我们画小人的过程就是通过以上两个函数将绘制原点移动到相应位置，然后执行绘制当前肢体的操作，这里需要注意的是，由于默认火柴人的肢体为刚体，在旋转位移过程中不会发生形变，所以绘制身体部位的 drawBody()、DrawShoulder()等函数内部是不做任何改动的，在绘制过程中改动的知识它绘制的位置。

画出火柴人

这一部分我们就要来介绍火柴人在行走过程中的绘制了，我们还是从代码讲起。首先看我们第一部分代码:

```
1.  //本次绘图最基础的开始坐标原点
2.  glTranslatef(0.0f, 0.0f, 0.0f);//平移矩阵(x,y,z)
3.  glRotatef(people.angle, 0, 1, 0);//旋转矩阵
4.  glPushMatrix();//保存当前模型的视图矩阵
```

我们来看一下以上代码的功能，以上代码的主要功能是确定火柴人绘制位置的整体坐标，你可以通过修改第 2 行 glTranslatef(x,y,z)的三个参数来控制火柴人在三维空间中的位移，你

可以通过第 3 行的 glRotatef(angle,x,y,z)来控制火柴人在空间中的旋转，因为这里我们选定了(0,1,0)为我们的正方向，所以本实验中我们的火柴人只能前后左右旋转，第一个参数 people.angle 为控制旋转角度，这个参数在我们之后的使用“wasd”控制火柴人方向中会用到。

接下来我们开始绘制人物，首先我们先画出手臂，其中 void DrawArmA()表示的是画出小臂，void DrawArmB()是表示画出大臂：

```
1.  glTranslatef(0.0f, 0.0f, 0.0f);
2.  //glTranslatef(25.0, 0.0, 0.0);
3.  glRotatef(-90, 0, 1, 0);
4.  glPushMatrix();
5.  glTranslatef(0.0f, 1.2f, 0.0f);
6.  glPushMatrix();
7.  glTranslatef(0.8, 0.0, 0.0);
8.  glTranslatef(0.0, -0.4, 0.0);
9.  glRotatef(people.angleoflarmb, 1, 0, 0);
10. glPushMatrix();
11. glTranslatef(0.0, -0.8, 0.0);
12. glTranslatef(0.0, -0.3, 0.0);
13. glRotatef(people.angleoflarma, 1, 0, 0);
14. people.DrawArmA();
15. glPopMatrix();
16. people.DrawArmB();
17. glPopMatrix();
18. //画出右臂
19. glPushMatrix();
20. glTranslatef(-0.8, 0.0, 0.0);
21. glTranslatef(0.0, -0.4, 0.0);
22. glRotatef(people.angleofrarmb, 1, 0, 0);
23. glPushMatrix();
24. glTranslatef(0.0, -0.8, 0.0);
25. glTranslatef(0.0, -0.3, 0.0);
26. glRotatef(people.angleofrarma, 1, 0, 0);
27. people.DrawArmA();
28. glPopMatrix();
29. people.DrawArmB();
30. glPopMatrix();
```

接下来我们画出头、脖子和肩膀：

```
1.  //return shoulder
2.  glPushMatrix();
3.  glTranslatef(0.0, 0.3, 0.0);
4.  glPushMatrix();
5.  glTranslatef(0.0, 0.7, 0.0);
6.  people.DrawHead();
```

```
glPopMatrix();
people.DrawNeck();
glPopMatrix();
people.DrawShoulder();
glPopMatrix();
```

接下来我们来完成双腿的绘制工作：

```
glPushMatrix();
glTranslatef(0.0f, -1.2f, 0.0f);
glPushMatrix();
glTranslatef(0.4, 0.0, 0.0);
glTranslatef(0.0, -0.4, 0.0);
glRotatef(people.angleofllegb, 1, 0, 0);
glPushMatrix();
glTranslatef(0.0, -1.5, 0.0);
glRotatef(people.angleofllega, 1, 0, 0);
people.DrawLegA();
glPopMatrix();
people.DrawLegB();
glPopMatrix();

glPushMatrix();
glTranslatef(-0.4, 0.0, 0.0);
glTranslatef(0.0, -0.4, 0.0);
glRotatef(people.angleofrlegb, 1, 0, 0);
glPushMatrix();
glTranslatef(0.0, -1.5, 0.0);
glRotatef(people.angleofrlega, 1, 0, 0);
people.DrawLegA();
glPopMatrix();
people.DrawLegB();
```

最后是绘制出胯和身体：

```
people.DrawHip();
glPopMatrix();
people.DrawBody();
glPopMatrix();
glPopMatrix();
```

至此，我们已经完成了对小人的绘制，但是为了下一次的绘制，我们需要记录下本次作出改变的地方，首先我们要记录当前火柴人的双臂双腿的已旋转角度，在之前的火柴人的类定义中，我们可以看到有几个标识符的定义：a,b,la,ra,ll,lg，这些标识符从前到后的依次表示大臂旋转角度、大腿旋转角度、左小臂旋转角度、右小臂旋转角度、左小腿旋转角度、右小

腿旋转角度。

```
people.angleoflarma += people.la;
people.angleofrarma += people.ra;
people.angleoflarmb += people.a;
people.angleofrarmb -= people.a;

people.angleofllega += people.ll;
people.angleofrlega += people.rl;
people.angleofllegb -= people.b;
people.angleofrlegb += people.b;
```

之后，我们知道人的关节是有旋转角度的，不可能旋转出 360 度，而且在普通走路中，我们手臂最大旋转应该是 45-50 度左右，所以每次更新旋转角度之后我们要判断一下当前是否已经到了最大角度，旋转到最大角度之后就要改变旋转方向复位。

```
if (50 > people.angleoflarmb && people.angleoflarmb > 0 && people.a > 0) {
    people.a = people.angle1;
    people.b = people.angle2;
    people.la = -people.angle1 * 0.75;
    people.ra = -people.angle1;
    people.ll = 0.75*people.angle2;
    people.rl = people.angle2 * 1.5;
}
if (people.angleoflarmb >= 50 && people.a > 0) {
    people.a = -people.angle1;
    people.b = -people.angle2;
}
if (people.angleoflarmb > 0 && people.a < 0) {
    people.a = -people.angle1;
    people.b = -people.angle2;
    people.la = people.angle1 * 0.75;
    people.ra = people.angle1;
    people.ll = -people.angle2 * 0.75;
    people.rl = -people.angle2 * 1.5;
}
if (-
50 < people.angleoflarmb && people.angleoflarmb <= 0 && people.a < 0) {
    people.a = -people.angle1;
    people.b = -people.angle2;
    people.la = -people.angle1;
    people.ra = -people.angle1 * 0.75;
    people.ll = people.angle2 * 1.5;
    people.rl = people.angle2 * 0.75;
}
```

```
if (people.angleoflarmb <= -50 && people.a < 0) {
    people.a = people.angle1;
    people.b = people.angle2;
}
if (people.angleoflarmb < 0 && people.a>0) {
    people.a = people.angle1;
    people.b = people.angle2;
    people.la = people.angle1;
    people.ra = people.angle1 * 0.75;
    people.ll = -people.angle2 * 1.5;
    people.rl = -people.angle2 * 0.75;
}
```

实现 WASD 控制火柴人方向转变

到目前为止，我们已经完成了火柴人的绘制和控制行走的工作，现在火柴人已经可以在原地完成行走的动作了，接下来我们要添加键盘控制事件来控制火柴人完成转向的工作。

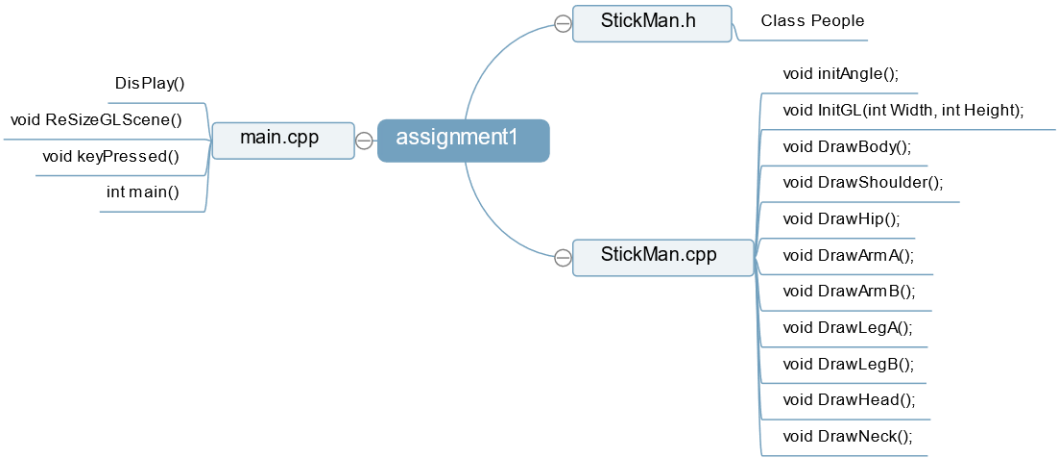
```
void keyPressed(unsigned char key, int x, int y) {
    switch (key) {
    case 'w':
        people.angle = 270;
        break;
    case 'a':
        people.angle = 0;
        break;
    case 's':
        people.angle = 90;
        break;
    case 'd':
        people.angle = 180;
        break;
    default:
        break;
    }
    people.initAngle();
}
```

然后我们在 main 函数中绑定键盘事件“

```
glutKeyboardFunc(&keyPressed);
//绑定键盘事件
```

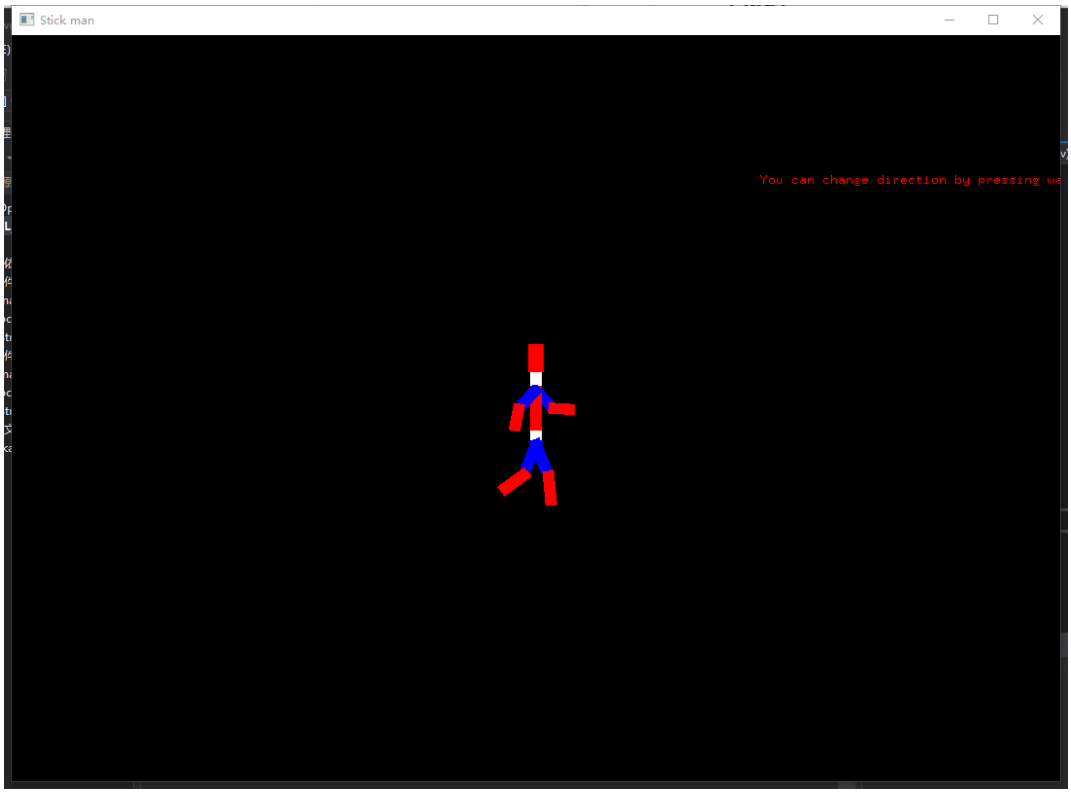
至此本实验完成。

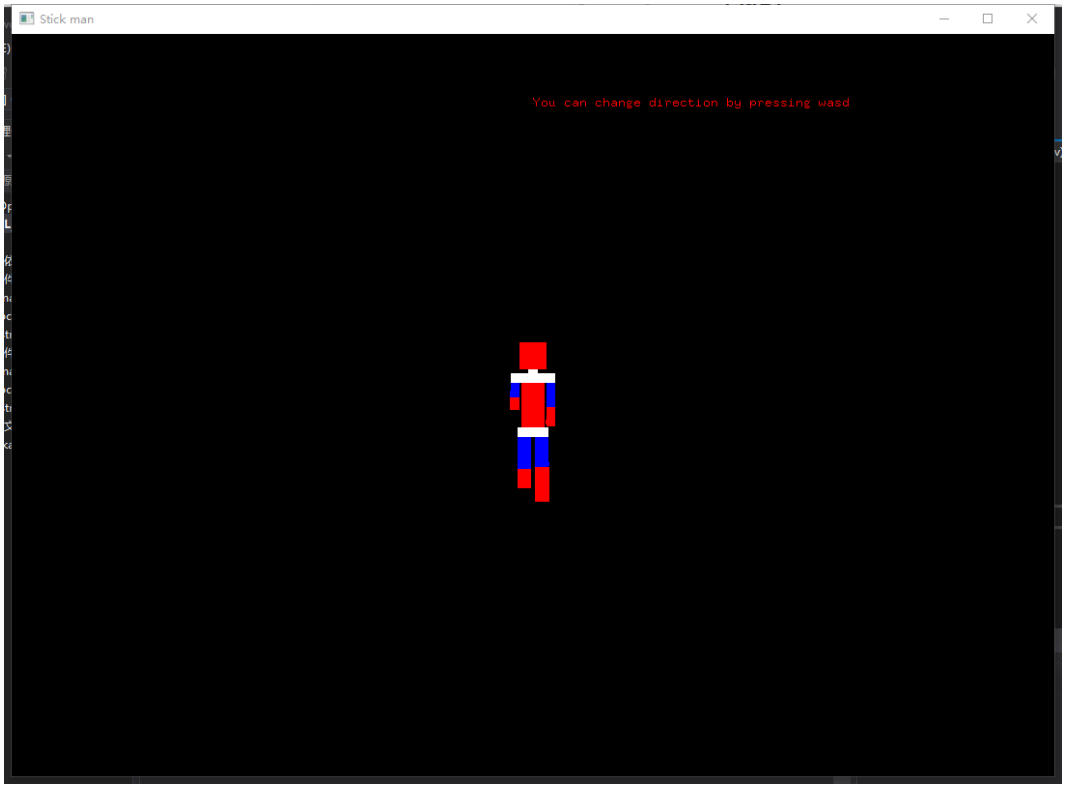
实验代码结构



实验结果和收获

结果展示





反思和收获

通过本实验，我们了解了图形学绘图的层级结构，尤其是在前一个变换矩阵的基础上进行本次修改，这一点的集中表现是在手臂和腿的绘制上，你需要在大臂变换矩阵的基础

上推导小臂变换矩阵，然后绘制大臂小臂。还是很好地。只不过不知道为什么，我没有发现我的初始化出了什么问题，不能设置相机的视角。

实验源代码

https://github.com/yuemos/Computer-Graphics.git