人体模型设计说明

小组成员：王郝杰 201812213502001 石君海 201812213502033

一、设计目的：

模仿人体动作，实现人体模型的动作仿真，对理解人体四肢及头部动作提供帮助

二、功能分析

本模型共有 12 个控制键，分别对应身体，头部，手臂，腿部以及各个关节的旋转。模型通过鼠标进行控制，每个对应部位可以从-180 度旋转到 180 度，实现了人的大部分肢体运动。通过构建立方体模块代替人的身体，各模块间转动实现肢体的运动。

三、运行环境

该模型属于 3D 模型的开发。建立模型的脚本语言有 HTML5、JavaScript、CSS，系统的平台是 Windows 10，开发工具是 Hbuilder，测试的浏览器有谷歌、 IE、 Hbuilder 自带的浏览器。

四、设计思路

我们小组根据自身能力选择构建此模型。通过构建大小不一的立方体拼接模拟人身体与四肢，但总体来说做出的模型简单且不够逼真。我们总的设计思路是要保证模型能够 通过旋转各个立方体以实现人体四肢的大部分运动

五、设计说明

1.首先，基于 HTML5 创建人体仿真模型（简易模型）；

2.创建模型之后我们所要考虑的就是如何对模型进行渲染上色,以及做一些必要的几何操作（转动、移动等），对于模型的渲染，我们采用的是基于 WebGL 的设计方法，主要涉及 JavaScript,HTML5。

3.人体模型的渲染主要分为六大块,包括界面设计,定义着色器,模型导入,几何变换,模型上色，效果渲染。

3.1 界面设计：

主要是对交互界面进行设计，添加交互控制，几何变换操作键等，其次要为后面的模型渲染创建一张画布，而后对模型的所有操作都会在这张画布上进行。

3.2 定义着色器：

这是用 WebGL 渲染模型最普通也是最重要的一步，着色器包括顶点着色器和面片着色器,这两个着色器将通过处理之后的一些模型信息的变量,对模型进行渲染,所以必不可缺。着色器可以在 html 中定义也可在外部 js 文件中进行定义，这里我们对着色器进行了打包，作为外部 js 文件导入使用。

3.2 模型创建：

这一步主要是利用建立的顶点信息实现人体仿真模型的创建，简单模拟人体形态。

3.3 几何变换：

对模型的旋转操作，方便对模型的观察，只需要利用相应的变换矩阵进行实现。

3.4 模型上色:

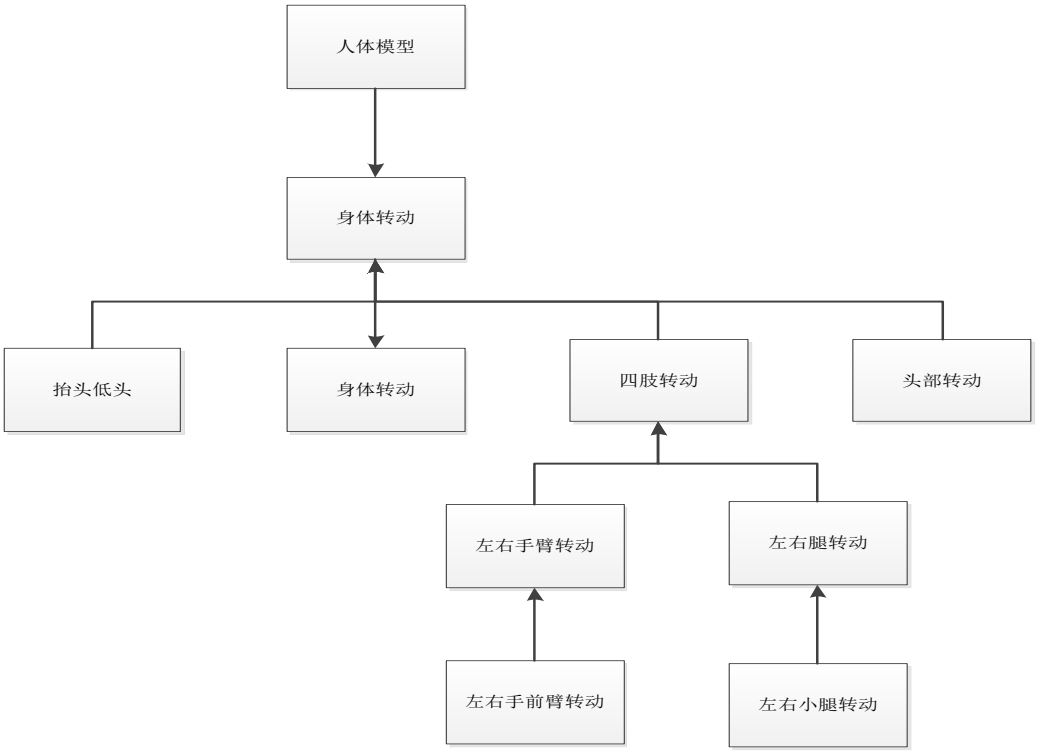
利用前面定义的着色器通过传递模型的颜色信息对模型进行上色处理。

3.5 效果渲染：

这一块主要包括灯光效果，渲染方式（实体、线框），投影方式（正交投影、透视投影以及相机的参数设置等等，这些操作都是为了使渲染出来的模型尽可能贴近真实。

六、功能模块设计

模型共有 11 个功能，分别为身体转动，转动头部，抬头低头，左右手臂与前臂活动，左右腿与小腿活动。



功能框架图

七、算法

旋转:

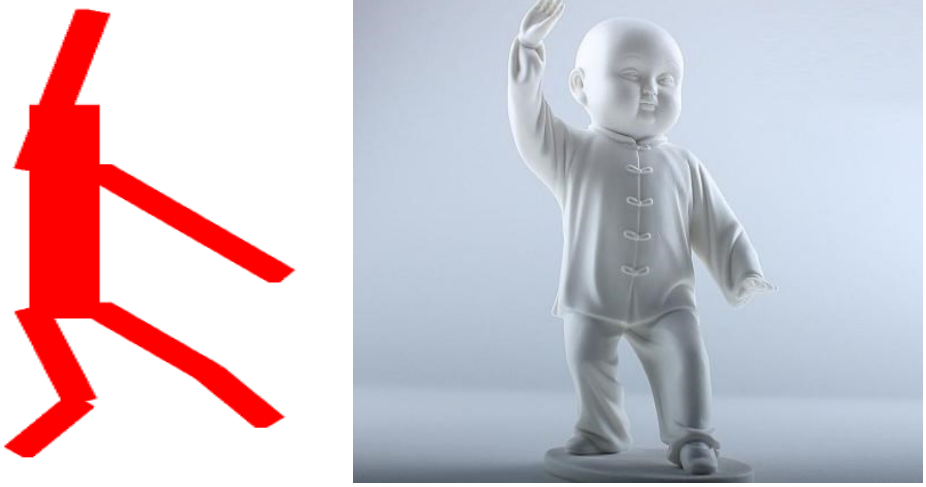
绕z轴旋转矩阵

$$R = R_Z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

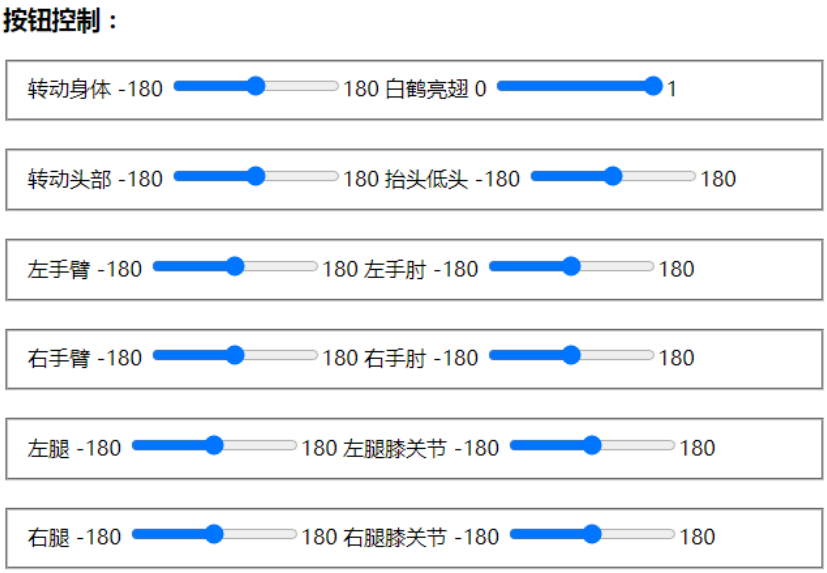
$$R = R_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8. 模型效果图



模型“白鹤亮翅”展示与对比图

九、模型交互界面展示



10. 主要的核心代码

```
//交互代码
document.getElementById("slider0").onchange = function() {
    theta[torsoId ] = event.srcElement.value;
    initNodes(torsoId);
};
document.getElementById("slider1").onchange = function() {
    theta[head1Id] = event.srcElement.value;
    initNodes(head1Id);
};

document.getElementById("slider2").onchange = function() {
     theta[leftUpperArmId] = event.srcElement.value;
     initNodes(leftUpperArmId);
};
document.getElementById("slider3").onchange = function() {
     theta[leftLowerArmId] =  event.srcElement.value;
     initNodes(leftLowerArmId);
};

document.getElementById("slider4").onchange = function() {
    theta[rightUpperArmId] = event.srcElement.value;
    initNodes(rightUpperArmId);
};
```

```
document.getElementById("slider5").onchange = function() {
     theta[rightLowerArmId] =  event.srcElement.value;
     initNodes(rightLowerArmId);
};
document.getElementById("slider6").onchange = function() {
    theta[leftUpperLegId] = event.srcElement.value;
    initNodes(leftUpperLegId);
};
document.getElementById("slider7").onchange = function() {
     theta[leftLowerLegId] = event.srcElement.value;
     initNodes(leftLowerLegId);
};
document.getElementById("slider8").onchange = function() {
     theta[rightUpperLegId] =  event.srcElement.value;
     initNodes(rightUpperLegId);
};
document.getElementById("slider9").onchange = function() {
    theta[rightLowerLegId] = event.srcElement.value;
    initNodes(rightLowerLegId);
};
document.getElementById("slider10").onchange = function() {
     theta[head2Id] = event.srcElement.value;
     initNodes(head2Id);
};
```

```
//模型进行旋转的定点后调用
function traverse(Id) {

    if(Id == null) return;
    stack.push(modelViewMatrix);
    modelViewMatrix = mult(modelViewMatrix, figure[Id].transform);
    figure[Id].render();
    if(figure[Id].child != null) traverse(figure[Id].child);
     modelViewMatrix = stack.pop();
    if(figure[Id].sibling != null) traverse(figure[Id].sibling);
}


function initNodes(Id) {
    var m = mat4();
    switch(Id){
      case torsoId:
      m = rotate(theta[torsoId], 0, 1, 0 );
      figure[torsoId] = createNode( m, torso, null, headId );
      break;
      case headId:
      case head1Id:
      case head2Id:
      m = translate(0.0, torsoHeight+0.5*headHeight, 0.0);
      m = mult(m, rotate(theta[head1Id], 1, 0, 0))
      m = mult(m, rotate(theta[head2Id], 0, 1, 0));
      m = mult(m, translate(0.0, -0.5*headHeight, 0.0));
      figure[headId] = createNode( m, head, leftUpperArmId, null);
      break;
      case leftUpperArmId:
      m = translate(-(torsoWidth+upperArmWidth), 0.9*torsoHeight, 0.0);
      m = mult(m, rotate(theta[leftUpperArmId], 1, 0, 0));
      figure[leftUpperArmId] = createNode( m, leftUpperArm, rightUpperArmId, leftLowerArmId );
      break;
      case rightUpperArmId:
      m = translate(torsoWidth+upperArmWidth, 0.9*torsoHeight, 0.0);
      m = mult(m, rotate(theta[rightUpperArmId], 1, 0, 0));
      figure[rightUpperArmId] = createNode( m, rightUpperArm, leftUpperLegId, rightLowerArmId );
      break;
      case leftUpperLegId:
      m = translate(-(torsoWidth+upperLegWidth), 0.1*upperLegHeight, 0.0);
      m = mult(m , rotate(theta[leftUpperLegId], 1, 0, 0));
      figure[leftUpperLegId] = createNode( m, leftUpperLeg, rightUpperLegId, leftLowerLegId );
      break;
      case rightUpperLegId:
      m = translate(torsoWidth+upperLegWidth, 0.1*upperLegHeight, 0.0);
      m = mult(m, rotate(theta[rightUpperLegId], 1, 0, 0));
      figure[rightUpperLegId] = createNode( m, rightUpperLeg, null, rightLowerLegId );
      break;
      case leftLowerArmId:
      m = translate(0.0, upperArmHeight, 0.0);
      m = mult(m, rotate(theta[leftLowerArmId], 1, 0, 0));
      figure[leftLowerArmId] = createNode( m, leftLowerArm, null, null );
      break;
      case rightLowerArmId:
      m = translate(0.0, upperArmHeight, 0.0);
      m = mult(m, rotate(theta[rightLowerArmId], 1, 0, 0));
      figure[rightLowerArmId] = createNode( m, rightLowerArm, null, null );
      break;
      case leftLowerLegId:
      m = translate(0.0, upperLegHeight, 0.0);
      m = mult(m, rotate(theta[leftLowerLegId], 1, 0, 0));
      figure[leftLowerLegId] = createNode( m, leftLowerLeg, null, null );
      break;
      case rightLowerLegId:
      m = translate(0.0, upperLegHeight, 0.0);
      m = mult(m, rotate(theta[rightLowerLegId], 1, 0, 0));
      figure[rightLowerLegId] = createNode( m, rightLowerLeg, null, null );
      break;

    }
}
```

```
 //绘制身体各部分
function torso() {

    instanceMatrix = mult(modelViewMatrix, translate(0.0, 0.5*torsoHeight, 0.0) );
    instanceMatrix = mult(instanceMatrix, scale4( torsoWidth, torsoHeight, torsoWidth));
    gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(instanceMatrix));
    for(var i =0; i<6; i++) gl.drawArrays(gl.TRIANGLE_FAN, 4*i, 4);
 }

function head() {

    instanceMatrix = mult(modelViewMatrix, translate(0.0, 0.5 * headHeight, 0.0 ));
    instanceMatrix = mult(instanceMatrix, scale4(headWidth, headHeight, headWidth) );
    gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(instanceMatrix));
    for(var i =0; i<6; i++) gl.drawArrays(gl.TRIANGLE_FAN, 4*i, 4);
 }

function leftUpperArm() {

    instanceMatrix = mult(modelViewMatrix, translate(0.0, 0.5 * upperArmHeight, 0.0) );
    instanceMatrix = mult(instanceMatrix, scale4(upperArmWidth, upperArmHeight, upperArmWidth) );
    gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(instanceMatrix));
    for(var i =0; i<6; i++) gl.drawArrays(gl.TRIANGLE_FAN, 4*i, 4);
 }

function leftLowerArm() {
 }

function rightUpperArm() {
 }

function rightLowerArm() {
 }

function  leftUpperLeg() {
 }

function leftLowerLeg() {
 }

function rightUpperLeg() {
 }

function rightLowerLeg() {
 }
```

```
//白鹤亮翅
document.getElementById("move").onchange = function() {

    theta[torsoId] = event.srcElement.value*65;
    initNodes(torsoId);

    theta[leftUpperArmId] = event.srcElement.value*15;
    initNodes(leftUpperArmId);

    theta[leftUpperLegId] = event.srcElement.value*160;
    initNodes(leftUpperLegId);
    theta[leftLowerLegId] = event.srcElement.value*60;
    initNodes(leftLowerLegId);

    theta[rightUpperArmId] = event.srcElement.value*130;
    initNodes(rightUpperArmId);

    theta[rightUpperLegId] = event.srcElement.value*130;
    initNodes(rightUpperLegId);
    theta[rightLowerLegId] = event.srcElement.value*10;
    initNodes(rightLowerLegId);

    /*theta[head1Id] = event.srcElement.value;
    initNodes(head1Id);

    theta[head1Id] = event.srcElement.value;
    initNodes(head1Id);

    theta[head1Id] = event.srcElement.value;
    initNodes(head1Id);*/

};
```

十、不足

（1）模型搭建不是很好，只能简易的仿真人体形态，视觉效果不是很好；
（2）人体仿真模型功能设计不足，功能实现较少；

十一、小组分工及自评

小组成员	分工情况	自评分数
王郝杰	实现模型旋转、展示文件制作，动作设计及制作	95
石君海	实现交互、 建立模型、文档制作	94