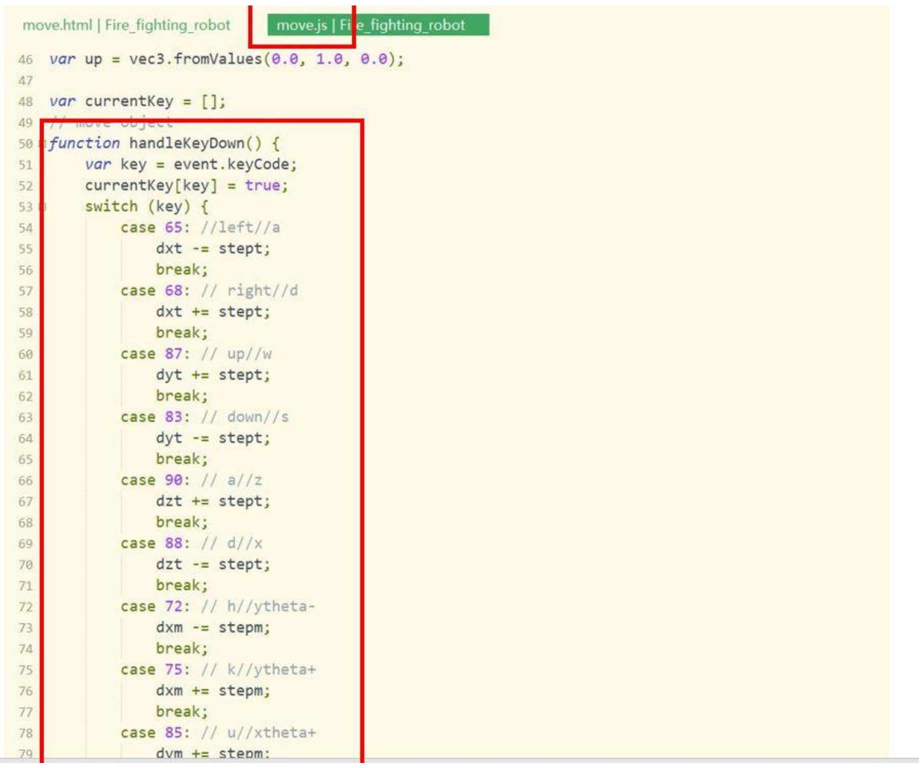
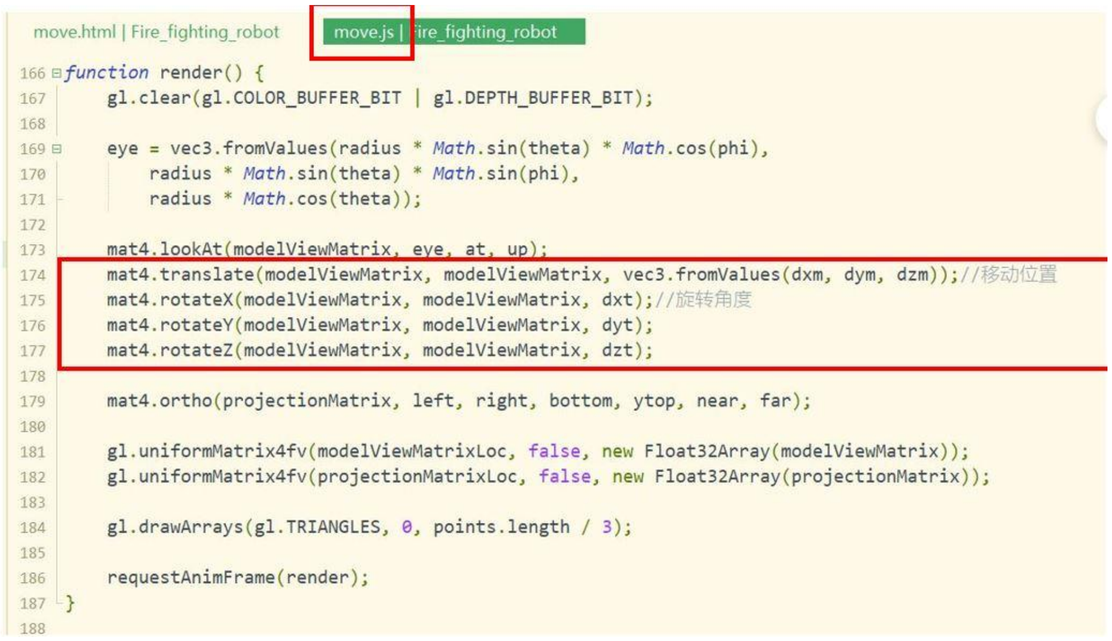
1. 机器人运动

（1）在 move.js 中通过按键事件侦听实现键盘与机器人的交互。


move.html | Fire_fighting_robot
move.js | Fire_fighting_robot

```
var up = vec3.fromValues(0.0, 1.0, 0.0);

var currentKey = [];
// move object
function handleKeyDown() {
    var key = event.keyCode;
    currentKey[key] = true;
    switch (key) {
        case 65: //left//a
            dxt -= stept;
            break;
        case 68: // right//d
            dxt += stept;
            break;
        case 87: // up//w
            dyt += stept;
            break;
        case 83: // down//s
            dyt -= stept;
            break;
        case 90: // a//z
            dzt += stept;
            break;
        case 88: // d//x
            dzt -= stept;
            break;
        case 72: // h//ytheta-
            dxm -= stepm;
            break;
        case 75: // k//ytheta+
            dxm += stepm;
            break;
        case 85: // u//xtheta+
            dym += stepm;
```

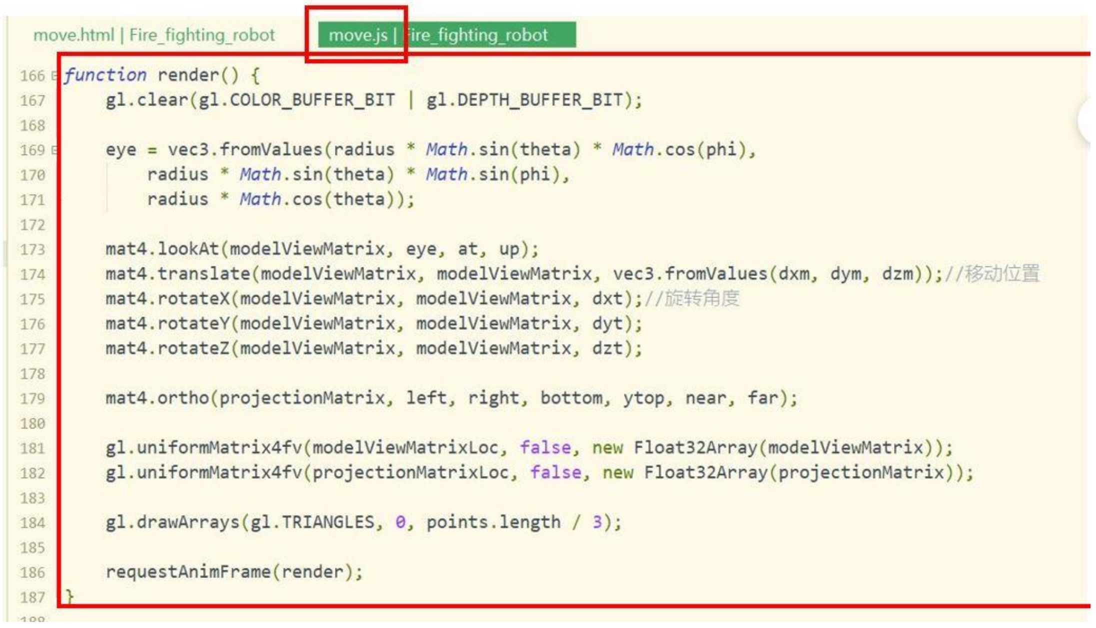
（2）在 move.js 中通过 mat4.translate 和 mat4.rotateX（mat4.rotateY，Mat4.rotateZ）分别实现平移和旋转。


move.html | Fire_fighting_robot
move.js | Fire_fighting_robot

```
function render() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    eye = vec3.fromValues(radius * Math.sin(theta) * Math.cos(phi),
        radius * Math.sin(theta) * Math.sin(phi),
        radius * Math.cos(theta));

    mat4.lookAt(modelViewMatrix, eye, at, up);
    mat4.translate(modelViewMatrix, modelViewMatrix, vec3.fromValues(dxm, dym, dzm));//移动位置
    mat4.rotateX(modelViewMatrix, modelViewMatrix, dxt);//旋转角度
    mat4.rotateY(modelViewMatrix, modelViewMatrix, dyt);
    mat4.rotateZ(modelViewMatrix, modelViewMatrix, dzt);

    mat4.ortho(projectionMatrix, left, right, bottom, ytop, near, far);

    gl.uniformMatrix4fv(modelViewMatrixLoc, false, new Float32Array(modelViewMatrix));
    gl.uniformMatrix4fv(projectionMatrixLoc, false, new Float32Array(projectionMatrix));

    gl.drawArrays(gl.TRIANGLES, 0, points.length / 3);

    requestAnimFrame(render);
}
```

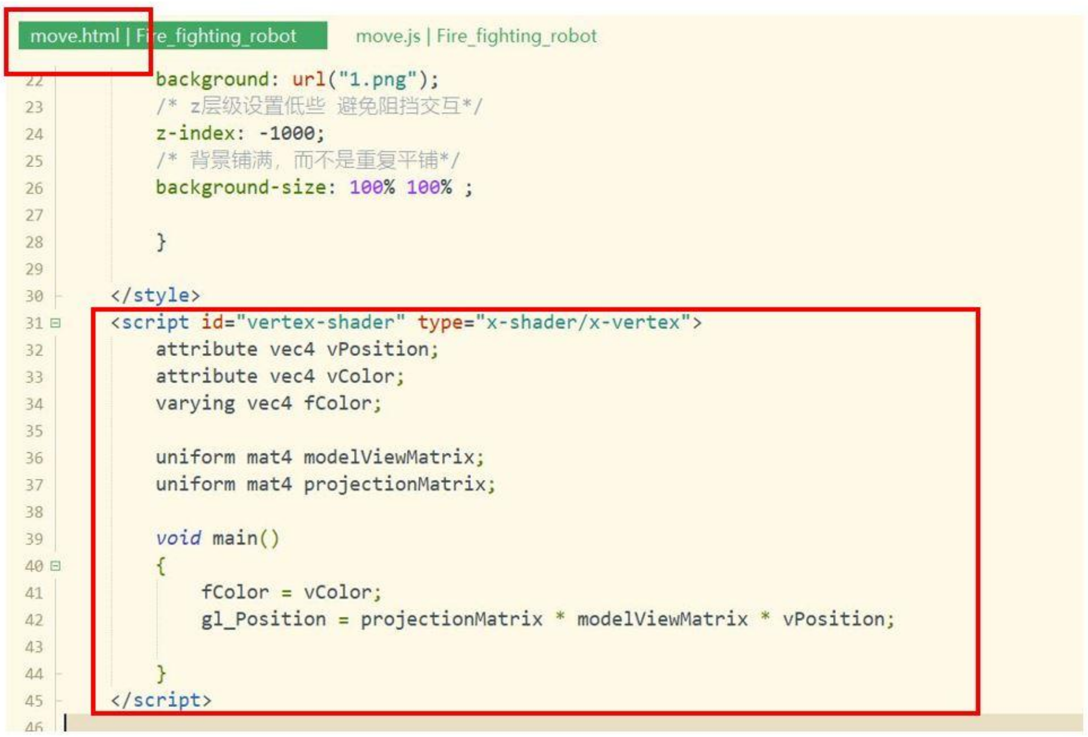
2. 投影

（1）在 move.js 中实现投影矩阵代码。


move.html | Fire_fighting_robot
move.js | Fire_fighting_robot

```
function render() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    eye = vec3.fromValues(radius * Math.sin(theta) * Math.cos(phi),
        radius * Math.sin(theta) * Math.sin(phi),
        radius * Math.cos(theta));

    mat4.lookAt(modelViewMatrix, eye, at, up);
    mat4.translate(modelViewMatrix, modelViewMatrix, vec3.fromValues(dxm, dym, dzm));//移动位置
    mat4.rotateX(modelViewMatrix, modelViewMatrix, dxt);//旋转角度
    mat4.rotateY(modelViewMatrix, modelViewMatrix, dyt);
    mat4.rotateZ(modelViewMatrix, modelViewMatrix, dzt);

    mat4.ortho(projectionMatrix, left, right, bottom, ytop, near, far);

    gl.uniformMatrix4fv(modelViewMatrixLoc, false, new Float32Array(modelViewMatrix));
    gl.uniformMatrix4fv(projectionMatrixLoc, false, new Float32Array(projectionMatrix));

    gl.drawArrays(gl.TRIANGLES, 0, points.length / 3);

    requestAnimFrame(render);
}
```

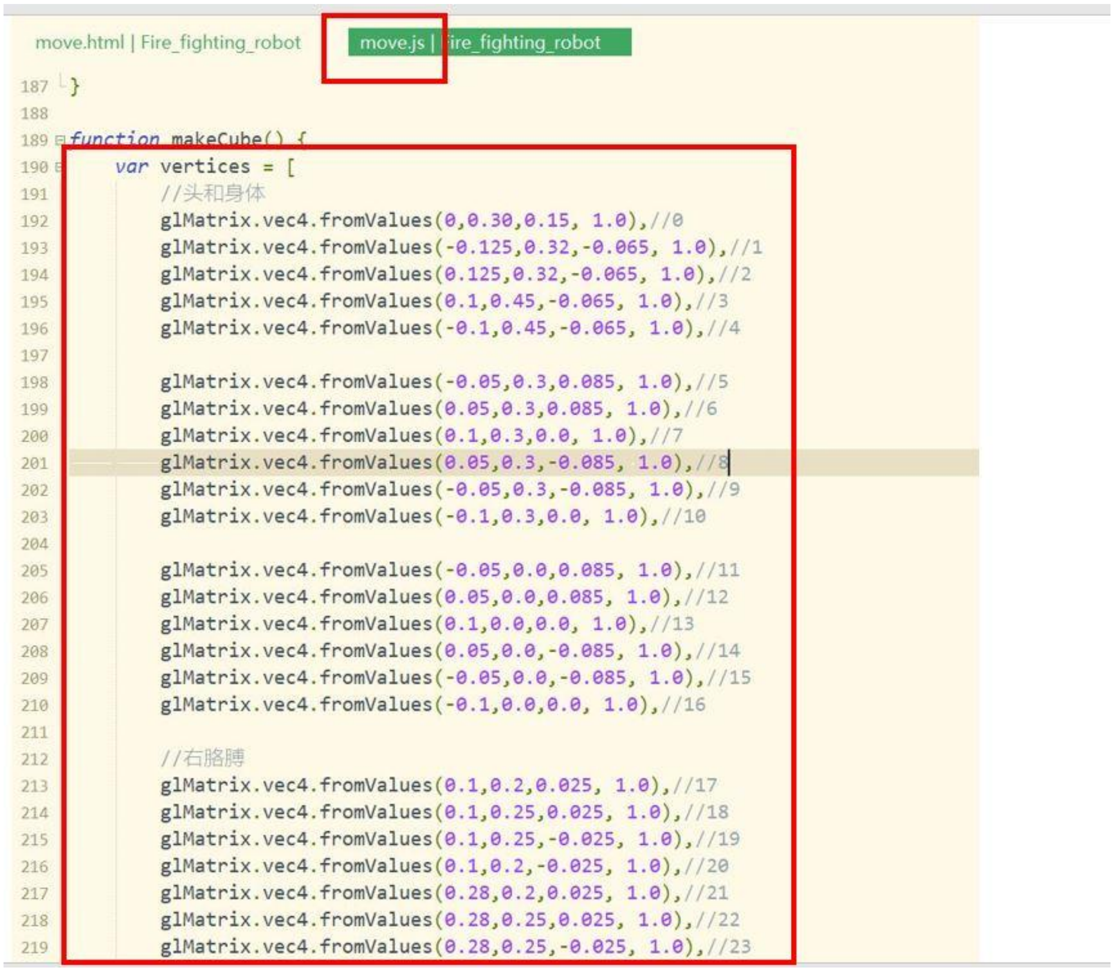
（2）在 move.html 实现投影矩阵顶点着色器


move.html | Fire_fighting_robot
move.js | Fire_fighting_robot

```
        background: url("1.png");
        /* z层级设置低些 避免阻挡交互*/
        z-index: -1000;
        /* 背景铺满，而不是重复平铺*/
        background-size: 100% 100% ;

        }

    </style>
    <script id="vertex-shader" type="x-shader/x-vertex">
        attribute vec4 vPosition;
        attribute vec4 vColor;
        varying vec4 fColor;

        uniform mat4 modelViewMatrix;
        uniform mat4 projectionMatrix;

        void main()
        {
            fColor = vColor;
            gl_Position = projectionMatrix * modelViewMatrix * vPosition;

        }
    </script>
```

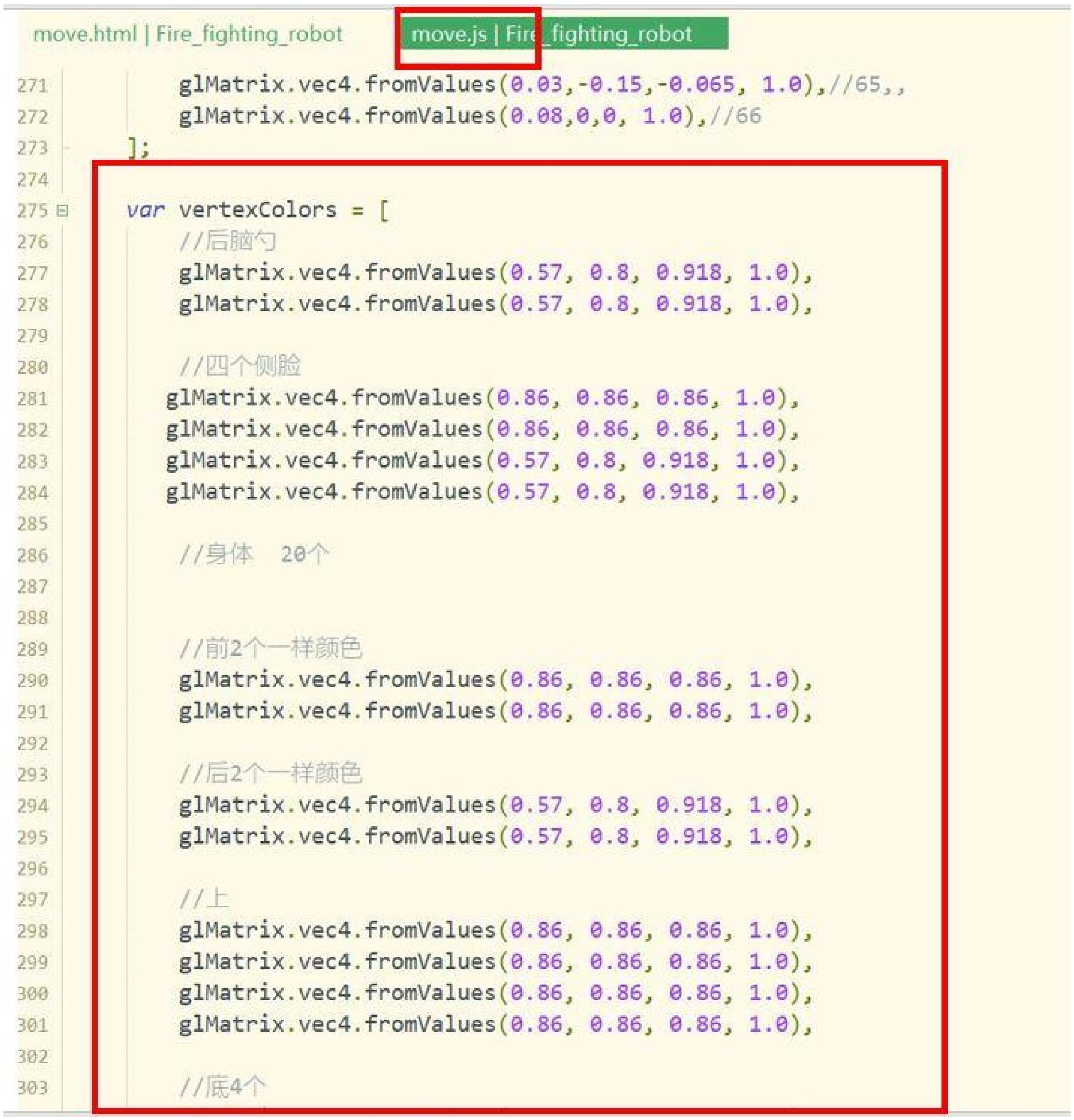
3. 构建机器人

（1）设置顶点


move.html | Fire_fighting_robot
move.js | Fire_fighting_robot

```
}

function makeCube() {
    var vertices = [
        //头和身体
        glMatrix.vec4.fromValues(0,0.30,0.15, 1.0),//0
        glMatrix.vec4.fromValues(-0.125,0.32,-0.065, 1.0),//1
        glMatrix.vec4.fromValues(0.125,0.32,-0.065, 1.0),//2
        glMatrix.vec4.fromValues(0.1,0.45,-0.065, 1.0),//3
        glMatrix.vec4.fromValues(-0.1,0.45,-0.065, 1.0),//4

        glMatrix.vec4.fromValues(-0.05,0.3,0.085, 1.0),//5
        glMatrix.vec4.fromValues(0.05,0.3,0.085, 1.0),//6
        glMatrix.vec4.fromValues(0.1,0.3,0.0, 1.0),//7
        glMatrix.vec4.fromValues(0.05,0.3,-0.085, 1.0),//8
        glMatrix.vec4.fromValues(-0.05,0.3,-0.085, 1.0),//9
        glMatrix.vec4.fromValues(-0.1,0.3,0.0, 1.0),//10

        glMatrix.vec4.fromValues(-0.05,0.0,0.085, 1.0),//11
        glMatrix.vec4.fromValues(0.05,0.0,0.085, 1.0),//12
        glMatrix.vec4.fromValues(0.1,0.0,0.0, 1.0),//13
        glMatrix.vec4.fromValues(0.05,0.0,-0.085, 1.0),//14
        glMatrix.vec4.fromValues(-0.05,0.0,-0.085, 1.0),//15
        glMatrix.vec4.fromValues(-0.1,0.0,0.0, 1.0),//16

        //右胳膊
        glMatrix.vec4.fromValues(0.1,0.2,0.025, 1.0),//17
        glMatrix.vec4.fromValues(0.1,0.25,0.025, 1.0),//18
        glMatrix.vec4.fromValues(0.1,0.25,-0.025, 1.0),//19
        glMatrix.vec4.fromValues(0.1,0.2,-0.025, 1.0),//20
        glMatrix.vec4.fromValues(0.28,0.2,0.025, 1.0),//21
        glMatrix.vec4.fromValues(0.28,0.25,0.025, 1.0),//22
        glMatrix.vec4.fromValues(0.28,0.25,-0.025, 1.0),//23
```

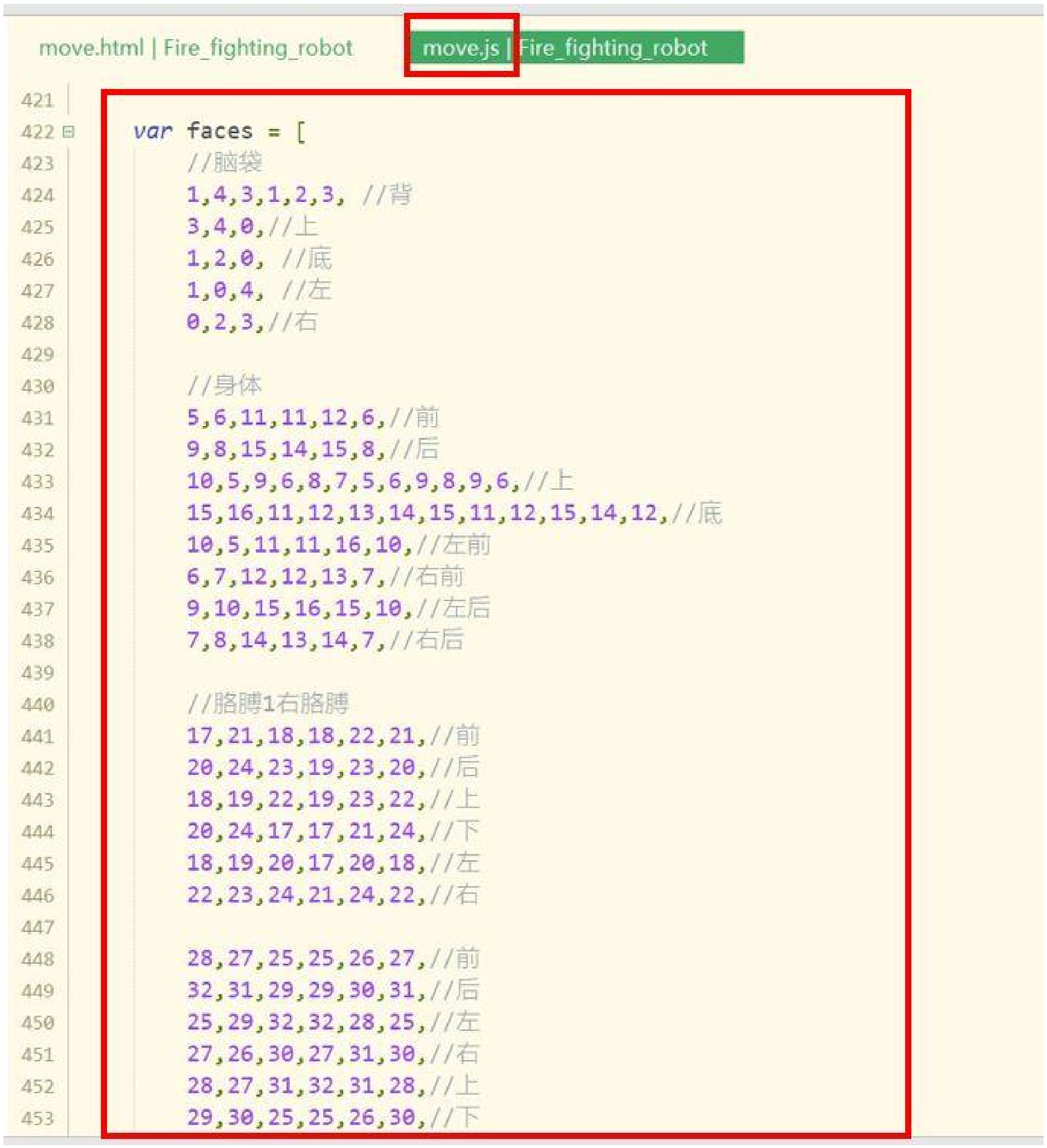
（2）设置顶点颜色



move.html | Fire_fighting_robot
move.js | Fire_fighting_robot

```
        glMatrix.vec4.fromValues(0.03,-0.15,-0.065, 1.0),//65,,
        glMatrix.vec4.fromValues(0.08,0,0, 1.0),//66
    ];

    var vertexColors = [
        //后脑勺
        glMatrix.vec4.fromValues(0.57, 0.8, 0.918, 1.0),
        glMatrix.vec4.fromValues(0.57, 0.8, 0.918, 1.0),

        //四个侧脸
       glMatrix.vec4.fromValues(0.86, 0.86, 0.86, 1.0),
       glMatrix.vec4.fromValues(0.86, 0.86, 0.86, 1.0),
       glMatrix.vec4.fromValues(0.57, 0.8, 0.918, 1.0),
       glMatrix.vec4.fromValues(0.57, 0.8, 0.918, 1.0),

        //身体  20个


        //前2个一样颜色
        glMatrix.vec4.fromValues(0.86, 0.86, 0.86, 1.0),
        glMatrix.vec4.fromValues(0.86, 0.86, 0.86, 1.0),

        //后2个一样颜色
        glMatrix.vec4.fromValues(0.57, 0.8, 0.918, 1.0),
        glMatrix.vec4.fromValues(0.57, 0.8, 0.918, 1.0),

        //上
        glMatrix.vec4.fromValues(0.86, 0.86, 0.86, 1.0),
        glMatrix.vec4.fromValues(0.86, 0.86, 0.86, 1.0),
        glMatrix.vec4.fromValues(0.86, 0.86, 0.86, 1.0),
        glMatrix.vec4.fromValues(0.86, 0.86, 0.86, 1.0),

        //底4个
```

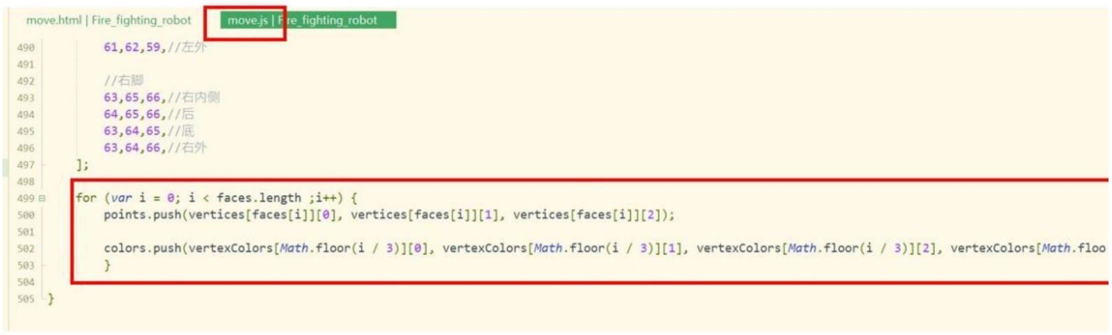
（3）定义机器人各面



move.html | Fire_fighting_robot   move.js | Fire_fighting_robot

```
var faces = [
    //脑袋
    1,4,3,1,2,3, //背
    3,4,0,//上
    1,2,0, //底
    1,0,4, //左
    0,2,3,//右

    //身体
    5,6,11,11,12,6,//前
    9,8,15,14,15,8,//后
    10,5,9,6,8,7,5,6,9,8,9,6,//上
    15,16,11,12,13,14,15,11,12,15,14,12,//底
    10,5,11,11,16,10,//左前
    6,7,12,12,13,7,//右前
    9,10,15,16,15,10,//左后
    7,8,14,13,14,7,//右后

    //胳膊1右胳膊
    17,21,18,18,22,21,//前
    20,24,23,19,23,20,//后
    18,19,22,19,23,22,//上
    20,24,17,17,21,24,//下
    18,19,20,17,20,18,//左
    22,23,24,21,24,22,//右

    28,27,25,25,26,27,//前
    32,31,29,29,30,31,//后
    25,29,32,32,28,25,//左
    27,26,30,27,31,30,//右
    28,27,31,32,31,28,//上
    29,30,25,25,26,30,//下
```

（4）顶点和颜色输入

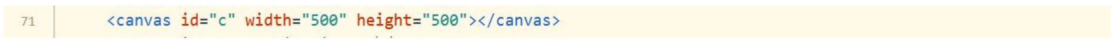


move.html | Fire_fighting_robot   move.js | Fire_fighting_robot

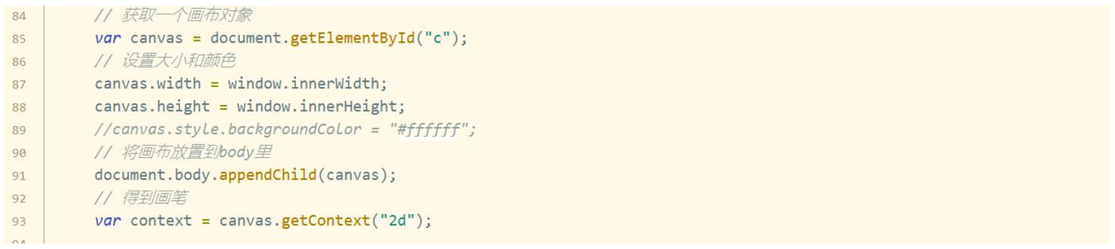
```
        61,62,59,//左外

        //右脚
        63,65,66,//右内侧
        64,65,66,//后
        63,64,65,//底
        63,64,66,//右外
    ];

    for (var i = 0; i < faces.length ;i++) {
        points.push(vertices[faces[i]][0], vertices[faces[i]][1], vertices[faces[i]][2]);

        colors.push(vertexColors[Math.floor(i / 3)][0], vertexColors[Math.floor(i / 3)][1], vertexColors[Math.floor(i / 3)][2], vertexColors[Math.floo
        }

}
```

4. 喷水喷干粉

（1）获取一个画布对象，并且设置其大小（以下所有步骤的代码都位于 **move.html** 中）

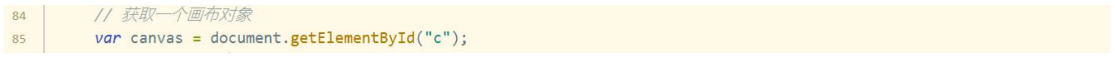


```
<canvas id="c" width="500" height="500"></canvas>
```

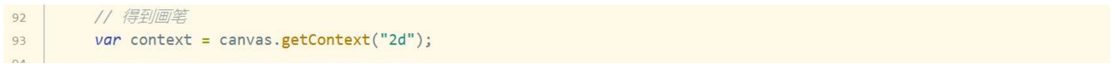


```
// 获取一个画布对象
var canvas = document.getElementById("c");
// 设置大小和颜色
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
//canvas.style.backgroundColor = "#ffffff";
// 将画布放置到body里
document.body.appendChild(canvas);
// 得到画笔
var context = canvas.getContext("2d");
```

（2）获取绘制上下文

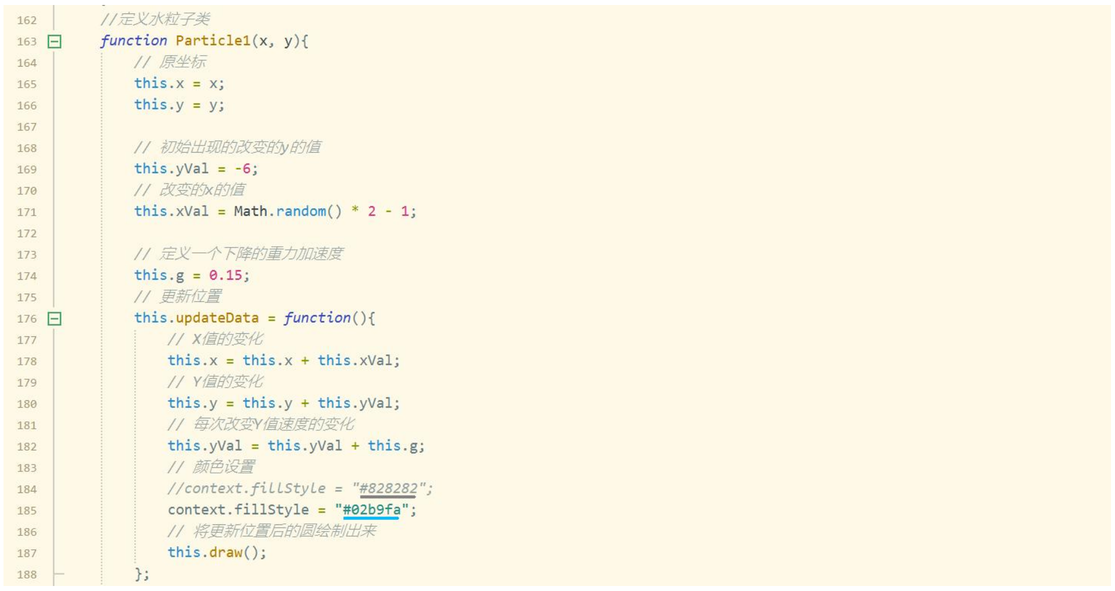


```
// 获取一个画布对象
var canvas = document.getElementById("c");
```

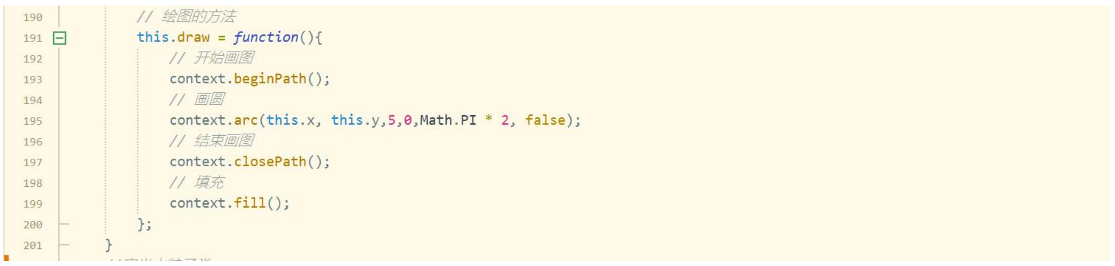


```
// 得到画笔
var context = canvas.getContext("2d");
```

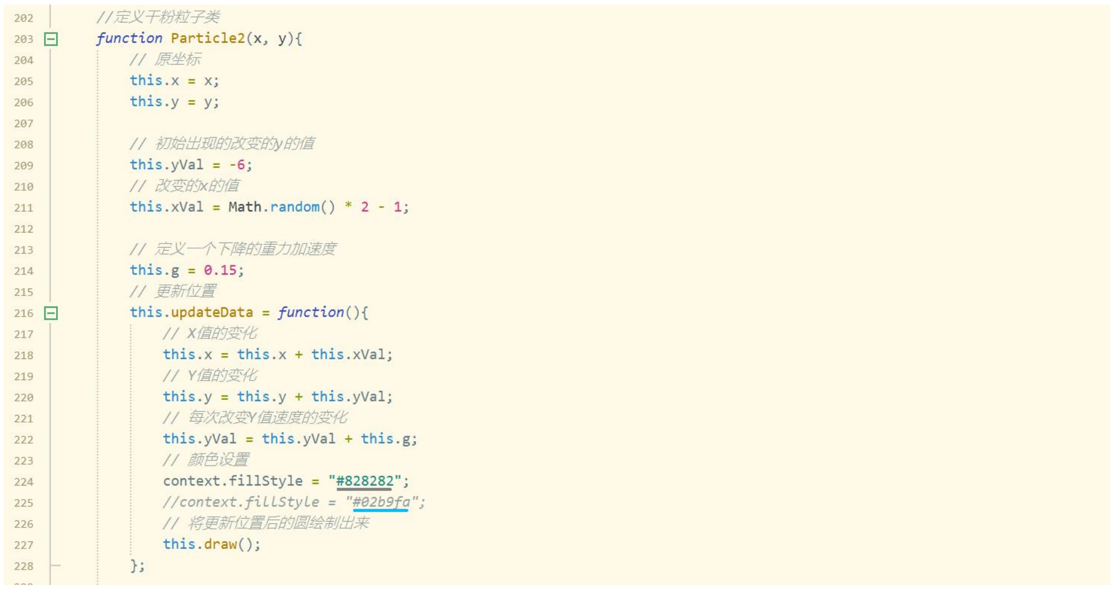
（3）定义粒子类（水粒子、干粉粒子）



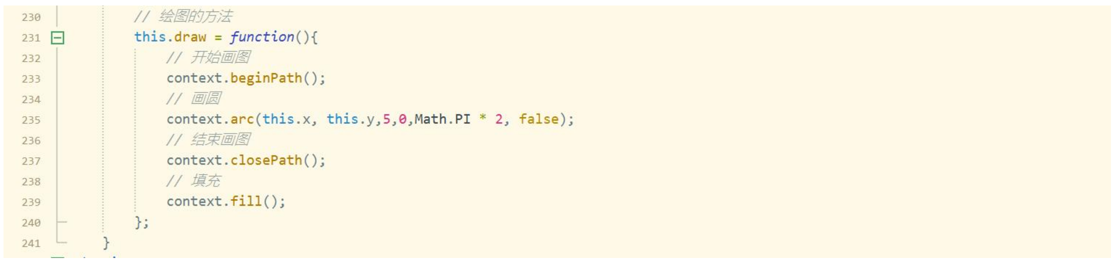
```
//定义水粒子类
function Particle1(x, y){
    // 原坐标
    this.x = x;
    this.y = y;

    // 初始出现的改变的y的值
    this.yVal = -6;
    // 改变的x的值
    this.xVal = Math.random() * 2 - 1;

    // 定义一个下降的重力加速度
    this.g = 0.15;
    // 更新位置
    this.updateData = function(){
        // X值的变化
        this.x = this.x + this.xVal;
        // Y值的变化
        this.y = this.y + this.yVal;
        // 每次改变Y值速度的变化
        this.yVal = this.yVal + this.g;
        // 颜色设置
        //context.fillStyle = "#828282";
        context.fillStyle = "#02b9fa";
        // 将更新位置后的圆绘制出来
        this.draw();
    };
```



```
    // 绘图的方法
    this.draw = function(){
        // 开始画图
        context.beginPath();
        // 画圆
        context.arc(this.x, this.y,5,0,Math.PI * 2, false);
        // 结束画图
        context.closePath();
        // 填充
        context.fill();
    };
}
```

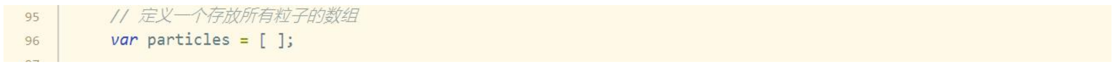


```
//定义干粉粒子类
function Particle2(x, y){
    // 原坐标
    this.x = x;
    this.y = y;

    // 初始出现的改变的y的值
    this.yVal = -6;
    // 改变的x的值
    this.xVal = Math.random() * 2 - 1;

    // 定义一个下降的重力加速度
    this.g = 0.15;
    // 更新位置
    this.updateData = function(){
        // X值的变化
        this.x = this.x + this.xVal;
        // Y值的变化
        this.y = this.y + this.yVal;
        // 每次改变Y值速度的变化
        this.yVal = this.yVal + this.g;
        // 颜色设置
        context.fillStyle = "#828282";
        //context.fillStyle = "#02b9fa";
        // 将更新位置后的圆绘制出来
        this.draw();
    };
```

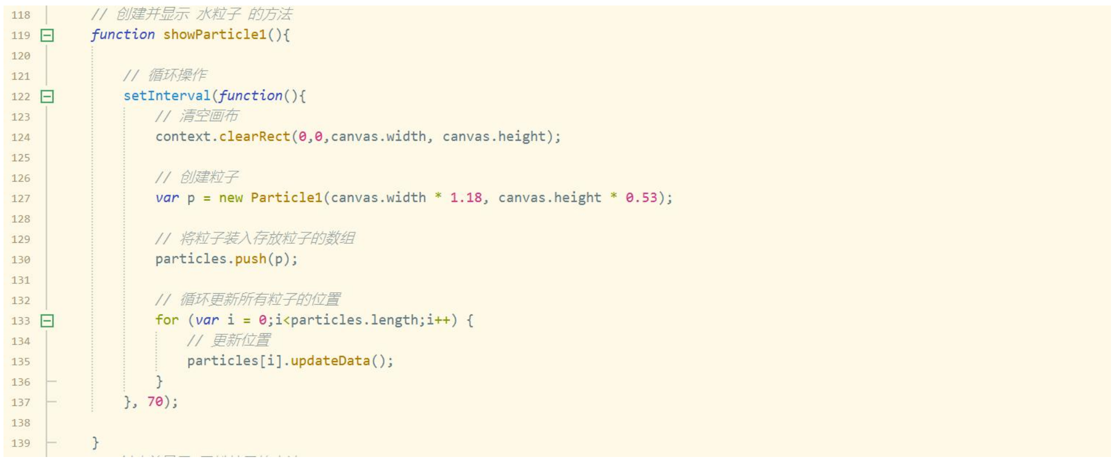


```
    // 绘图的方法
    this.draw = function(){
        // 开始画图
        context.beginPath();
        // 画圆
        context.arc(this.x, this.y,5,0,Math.PI * 2, false);
        // 结束画图
        context.closePath();
        // 填充
        context.fill();
    };
}
```

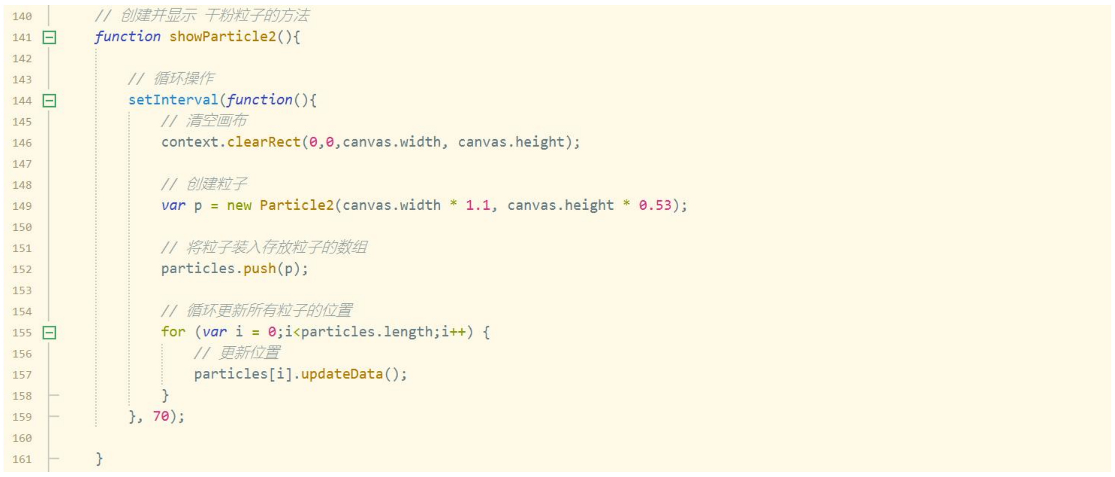
（4）创建并显示水粒子与干粉粒子的方法



```
// 定义一个存放所有粒子的数组
var particles = [ ];
```

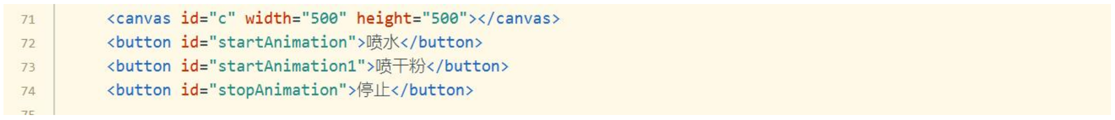


```
// 创建并显示 水粒子 的方法
function showParticle1(){

    // 循环操作
    setInterval(function(){
        // 清空画布
        context.clearRect(0,0,canvas.width, canvas.height);

        // 创建粒子
        var p = new Particle1(canvas.width * 1.18, canvas.height * 0.53);

        // 将粒子装入存放粒子的数组
        particles.push(p);

        // 循环更新所有粒子的位置
        for (var i = 0;i<particles.length;i++) {
            // 更新位置
            particles[i].updateData();
        }
    }, 70);

}
```

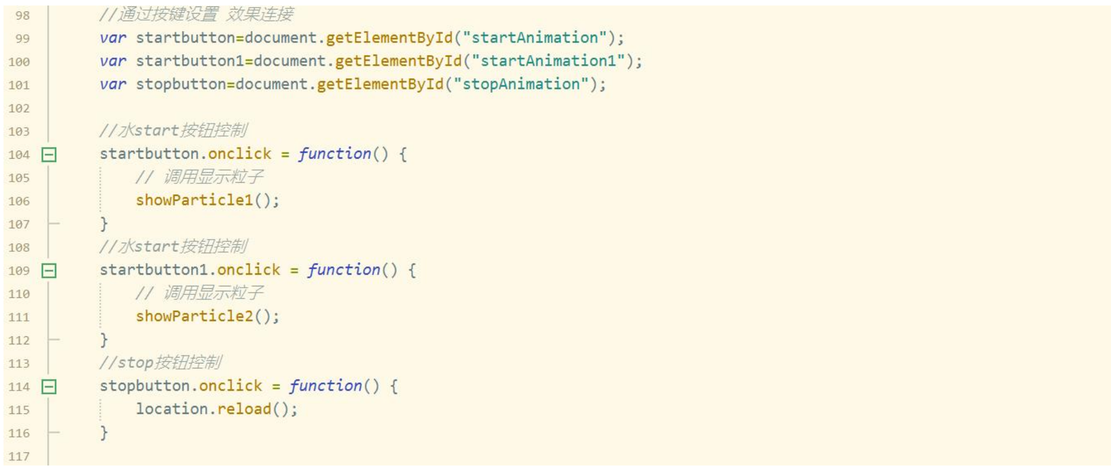


```
// 创建并显示 干粉粒子的方法
function showParticle2(){

    // 循环操作
    setInterval(function(){
        // 清空画布
        context.clearRect(0,0,canvas.width, canvas.height);

        // 创建粒子
        var p = new Particle2(canvas.width * 1.1, canvas.height * 0.53);

        // 将粒子装入存放粒子的数组
        particles.push(p);

        // 循环更新所有粒子的位置
        for (var i = 0;i<particles.length;i++) {
            // 更新位置
            particles[i].updateData();
        }
    }, 70);

}
```

（5）通过按钮事件侦听实现喷洒不同粒子的效果



```
<canvas id="c" width="500" height="500"></canvas>
<button id="startAnimation">喷水</button>
<button id="startAnimation1">喷干粉</button>
<button id="stopAnimation">停止</button>
```



```
//通过按键设置 效果连接
var startbutton=document.getElementById("startAnimation");
var startbutton1=document.getElementById("startAnimation1");
var stopbutton=document.getElementById("stopAnimation");

//水start按钮控制
startbutton.onclick = function() {
    // 调用显示粒子
    showParticle1();
}
//水start按钮控制
startbutton1.onclick = function() {
    // 调用显示粒子
    showParticle2();
}
//stop按钮控制
stopbutton.onclick = function() {
    location.reload();
}

```