2022.11.8 计算机图形学——课程设计阶段汇报

1. 系统完整功能描述
   - 机器人身体构造
   - 机器人移动（上下左右前后）
   - 机器人按键实现正投影
   - 机器人喷水
   - 机器人喷干粉
2. 各模块流程、结构具体实现，关键函数、变量等说明
   - ✓ 机器人身体构造模块

     模块流程:

     顶点坐标设置、顶点颜色设置、面片顶点设置、顶点颜色输入、渲染。

     结构具体实现:

```
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
```

```
    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),
```

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

makeCube()函数部分实现机器人顶点位置、顶点颜色、面片顶点的设置;

```
    //顶点
    vBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, vBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(points), gl.STATIC_DRAW);

    vPosition = gl.getAttribLocation(program, "vPosition");
    gl.vertexAttribPointer(vPosition, 3, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);
```

顶点的位置坐标存入缓存;

```
144     //颜色
145     cBuffer = gl.createBuffer();
146     gl.bindBuffer(gl.ARRAY_BUFFER, cBuffer);
147     gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colors), gl.STATIC_DRAW);
148
149     vColor = gl.getAttribLocation(program, "vColor");
150     gl.vertexAttribPointer(vColor, 4, gl.FLOAT, false, 0, 0);
151     gl.enableVertexAttribArray(vColor);
```

顶点的颜色信息存入缓存;

```
183
184     gl.drawArrays(gl.TRIANGLES, 0, points.length / 3);
185
```

在 render()函数中将机器人渲染到三维画布上。

关键函数:

makeCube()函数，用于设置机器人顶点坐标、顶点颜色参数、面片顶点组成。

initSphere()函数，用于画布获取，顶点、颜色输入缓存区，调用 render()函数渲染。

render()函数，用于将机器人渲染到三维画布中。

变量说明：

常量 vec3、vec4、mat4 ——glMatrix 库定义了名为 vec3、vec4 的矢量，用于处理 3、4 个数字的向量；定义了 mat4 的数组，用于处理 4*4 的矩阵。

匿名类型 canvas——用于存放绘画的画布对象。

gl——用于创建 WebGL。

points 数组、colors 数组作为容器，用于存放顶点坐标与颜色参数。

vBuffer、cBuffer 用于做缓存，存入顶点与颜色的参数。

vPosition、vColor 用于获取着色器中的对应变量，并向其传递数据。

✓ 干冰喷水效果实现模块

模块流程:

按钮控制设置，喷水效果位置设置、颜色设置、顶点设置、动画设置。

结构具体实现:

效果展示位置设置

```
7       #c{
8           width:240px;
9           height:240px;
10          position: absolute;
11          top:200px;
12          left:200px;
13          border-radius: 50% 50% 50% 50%
14      }
```

按钮控制设置

```
121             var startbutton=document.getElementById("startAnimation");
122             var stopbutton=document.getElementById("stopAnimation");
123             //stop按钮控制
124             stopbutton.onclick = function() {
125                 c1.style.display = 'none';
126             }
127             //start按钮控制
128             startbutton.onclick = function() {
129                 c1.style.display = 'block';
130             }
```

动画设置

```
142     // 动画开始
143     //初始化
144     var anim = {
145     init: function () {
146         //获取画布
147         var canvas = document.getElementById('c');
148         //定义画布的长宽
149         canvas.width = window.innerWidth;
150         canvas.height = window.innerHeight;
151         //创建2d空间 用this相当于全局变量
152         this.c = canvas.getContext('2d');
153         this.letters = "O";
154         this.gravity = 0.1;
155         this.maxParticles = 100;
156         this.cw = canvas.width,
157             this.ch = canvas.height,
158             this.particles = []
159     },
```

执行动画设置

```
161     //执行动画
162     render: function () {
163         this.fadeCanvas();
164         //添加粒子的属性
165         var particle = {
166             //粒子显示的位置
167             x: anim.cw / 2,
168             y: anim.ch,
169             //x,y轴速度值计算
170             xSpeed: (Math.random() * 20) - 10,
171             ySpeed: (Math.random() * 20) - 10,
172             size: 1,
173             //
174             character: anim.letters[Math.floor(Math.random() * anim.letters.length)],
175             //粒子颜色设定
176             //灰色:
177             //color: [180, 0, 50, .2]
178             //蓝色:
179             color: [196, 100, 49, .2]
180
181         }
```

粒子设置

```
187     //绘制水滴
188     //可以更改粒子为渐变色
189     drawParticles: function () {
190         var paritcleCount = this.particles.length;
191         var c = this.c;
192         for (var i = 0; i < paritcleCount; i++) {
193             var particle = this.particles[i];
194             var h = particle.color[0],
195                 s = particle.color[1] + '%',
196                 l = particle.color[2] + '%',
197                 a = particle.color[3];
198             var hsla = 'hsla(' + h + ',' + s + ',' + l + ',' + a + ')';
199             c.font = "10px sans-serif";
200             c.fillStyle = hsla;
201             c.fillText(particle.character, particle.x, particle.y);
```

由于渐变效果不是很理想，未采用渐变效果

```
203             //粒子速度
204             particle.x += particle.xSpeed;
205             particle.y += particle.ySpeed;
206
207             //粒子大小
208             particle.size *= 0.96;
209             //粒子位置
210             particle.y *= 0.96;
211
212             //粒子渐变色效果
213
214             //particle.color[2] *= 0.995;
215             //particle.color[0] += 0.98;
216             //if (particle.color[0] > 253) {
217             //    particle.color[2] = 100;
218             //   particle.color[3] = 1;
219             //}
```

关键函数:

render()函数，用于将机器人渲染到三维画布中。

fadecanvas()函数，用于清除画布。

Init()函数，实现喷水、干冰效果。

tidyParticles()函数，用于判断粒子边界值是否处于画布中。

drawParticles()函数，用于绘制水滴。

变量说明：

Canvas.width 定义了画布的宽度

Canvas.height 定义了画布的高度

X 定义了粒子在 x 轴显示位置

Y 定义了粒子在 y 轴显示位置

xSpeed 定义了粒子在 x 轴方向的速度

ySpeed 定义了粒子在 y 轴方向的速度

hsla 定义了粒子的颜色，可以通过更改粒子累加数值来改变颜色渐变程度

particle.size 粒子的大小设置

✓ 机器人相机模块

模块流程：

设置机器人原始坐标；
按键触发，生成机器人对应坐标进行调整的数值；
相机定位；
依据新的角度与位移参数进行模型视图矩阵的更新　旋转矩阵、位移矩阵；
选定镜头，设置投影矩阵，正投影；
裁剪，设定有效成像空间(Viewing volume)；
渲染。

结构具体实现：

```
190 function makeCube() {
191     var vertices = [
192         //头和身体
193         glMatrix.vec4.fromValues(0,0.30,0.15, 1.0),//0
194         glMatrix.vec4.fromValues(-0.125,0.32,-0.065, 1.0),//1
```

makeCube()函数，设置机器人原始坐标；

```
113 function handleKeyUp() {
114     currentKey[event.keyCode] = false;
115 }
116
```

```
50 function handleKeyDown() {
51     var key = event.keyCode;
52     currentKey[key] = true;
53     switch (key) {
54         case 65: //left//a
55             dxt -= stept;
56             break;
57         case 68: // right//d
58             dxt += stept;
59             break;
```

按键触发，生成机器人对应坐标进行调整的数值；

```
169     eye = vec3.fromValues(radius * Math.sin(theta) * Math.cos(phi),
170         radius * Math.sin(theta) * Math.sin(phi),
171         radius * Math.cos(theta));
172
173     mat4.lookAt(modelViewMatrix, eye, at, up);
```

相机定位；

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

依据新的角度与位移参数进行模型视图矩阵的更新　旋转矩阵、位移矩阵；

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

选定镜头，设置投影矩阵，正投影；
裁剪，设定有效成像空间(Viewing volume)；

```
184     gl.drawArrays(gl.TRIANGLES, 0, points.length / 3);
185
186     requestAnimFrame(render);
```

在画布中渲染机器人；

关键函数：

handleKeyUp()函数与 handleKeyDown()函数，用于捕捉按键触发的信息，生成机器人对应坐标进行调整的数值；

requestAnimFrame()函数，用于通知浏览器重采用动画（即进行调整后的机器人的重绘）。

变量说明：

projectionMatrix 用于存放投射矩阵。
modelViewMatrix 用于存放模型视图矩阵。
modelViewMatrixLoc, projectionMatrixLoc;用于存放 shader 变量。
near、far、left、right、ytop、bottom 用于构建正交投影矩阵。
radius、theta、phi 用于设置视点矢量。
dxt、dyt、dzt 用于存放各个坐标方向的移动角度。
dxm、dym、dzm 用于存放各个坐标方向的移动距离。
stept、stepm 用于存放移动角度的步伐与移动距离的步伐。
eye、at、up 用于存放视点、视线、视方向的矢量。
currentKey 数组用于存放按钮点击事件的监听。

3. 所采用实现工具，开发环境，主要工具库等

（1）实现工具

Hbuilder，vscode

（2）开发环境

编译器采用：Hbuilder，浏览器采用：Chorme 浏览器，

（3）主要工具库

webgl-utils.js，objloader.js，initShaders.js，gl-matrix.js

4. 目前进度

实现了机器人，移动，喷水，喷干粉，投影。