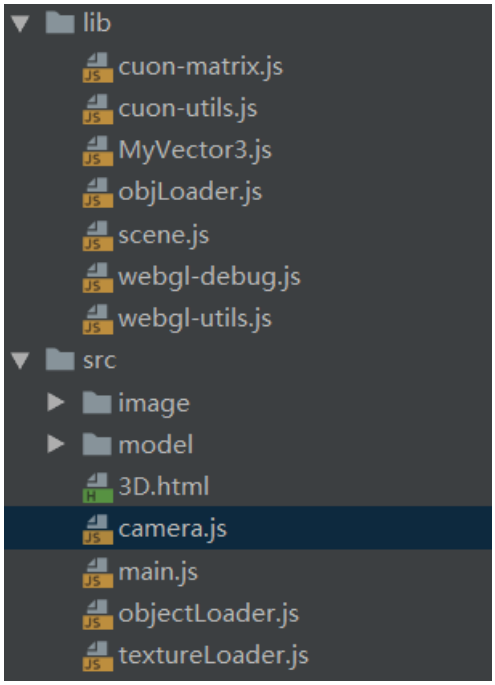
计算机图形学 pj3 说明文档

一、代码结构



lib 包：包含 webgl 的一些配置文件以及模型加载和配置信息
image 包：包含地板和箱子的纹理贴图
model 包：模型对象文件

objectLoader.js：根据传入的模型类对象的信息加载模型并设置各种属性及渲染

```
//模型加载类
class ObjectLoader {
    //构造方法
    constructor(o, config) {
        this.gl = config.gl;
        this.o = o;
    }

    init() {...}

    initShaders() {...}

    // 读取文件
    readOBJFile(fileName, scale, reverse) {...}

    onReadOBJFile(fileString, fileName, scale, reverse) {...}

    onReadComplete() {...}

    // render the object
    render(now) {...}
}
```

textureLoader.js：根据传入参数的不同分别渲染箱子和地板的纹理

```
class TextureLoader {
    constructor(o, config) {
        this.o = o;
        this.gl = config.gl;
        this.config = config;
    }

    init() {...}

    initShaders() {...}

    // 初始化纹理
    initTextures() {...}

    // 加载纹理
    loadTexture() {...}

    // 渲染纹理
    render(now) {...}
}
```

camera.js：负责相机的平移和旋转，通过键盘控制改变 state 数组中各个状态，再渲染

```
class Camera {
    static init() {...}

    // 得到mvpMatrix
    static getMvpMatrix() {...}

    // 相机的平移
    static move(x, y) {...}

    //相机的旋转
    static rotate(x, y) {...}
}
```

main.js：程序入口
首先调用对象加载类加载地板、箱子及其他模型，当加载的是鸟模型时让其旋转

```
//地板
let floorLoader = new TextureLoader(floorRes, config: {
    'gl': gl,
    'textureIndex': 0
}).init();
loaders.push(floorLoader);
//箱子
let boxLoader = new TextureLoader(boxRes, config: {
    'gl': gl,
    'textureIndex': 1
}).init();
loaders.push(boxLoader);
//其它模型
for (let object of ObjectList) {
    let loader = new ObjectLoader(object, config: {
        'gl': gl
    }).init();

 //当模型是鸟时让鸟旋转
if (object.objFilePath.indexOf('bird') > 0) {
    loader.nextTick = (now) => {
        let rotate = (now / 1000 * 120) % 360;
        let trans = (now / 500);
        loader.o.transform[1].content[0] = rotate;
        loader.o.transform[0].content[0] = Math.sin(trans) * 5;
        loader.o.transform[0].content[1] = Math.cos(trans) * 3 + 10;
        loader.o.transform[0].content[2] = Math.sin(trans) * 4 + 10;
    }
}
    loaders.push(loader);
}
```

初始化相机

```
static init() {
    Camera.fov = CameraPara.fov;
    Camera.near = CameraPara.near;
    Camera.far = CameraPara.far;
    Camera.at = new Vector3(CameraPara.at);
    Camera.eye = new Vector3(CameraPara.eye);
    Camera.up = new Vector3(CameraPara.up);

    Camera.state = {
        moveUp: 0, moveDown: 0, moveLeft: 0, moveRight: 0,
        rotateUp: 0, rotateDown: 0, rotateLeft: 0, rotateRight: 0,
        pointLight: false
    };
```

添加键盘点击事件改变 camera 类中 state 数组各状态的值

```
function addEvent() {
    let keys = new Map();
    keys.set('W', 'moveUp');
    keys.set('S', 'moveDown');
    keys.set('A', 'moveLeft');
    keys.set('D', 'moveRight');
    keys.set('I', 'rotateUp');
    keys.set('K', 'rotateDown');
    keys.set('J', 'rotateLeft');
    keys.set('L', 'rotateRight');
    keys.set('F', 'pointLight');

    //通过键盘点击改变各种状态
    keys.forEach( callbackfn: () => {
        window.addEventListener( type: 'keydown', listener: (e) => {
            let key = String.fromCharCode(e.which);
            if (!keys.get(key))
                return;
            if (key === 'F') {
                Camera.state[keys.get(key)] = !Camera.state[keys.get(key)];
            } else
                Camera.state[keys.get(key)] = 1;
        });

        window.addEventListener( type: 'keyup', listener: (e) => {
            let key = String.fromCharCode(e.which);
            if (!keys.get(key))
                return;
            if (key === 'F')
                return;
            Camera.state[keys.get(key)] = 0;
        });
```

最后渲染

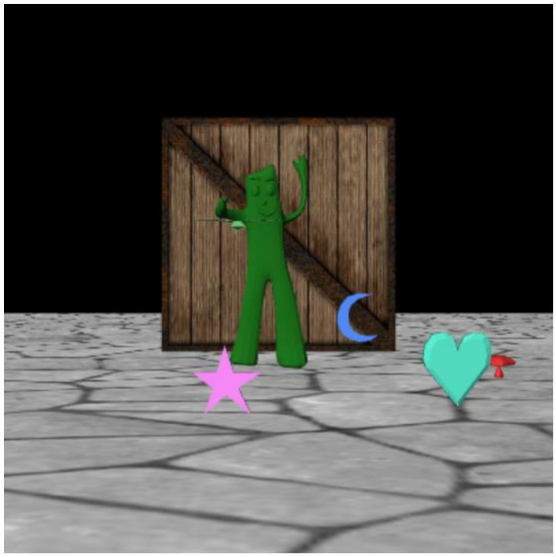
```
let tick = (now) => {

    gl.clearColor( red: 0, green: 0, blue: 0, alpha: 1);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);


    let time = now - last;
    last = now;

    let posX = (Camera.state.moveUp - Camera.state.moveDown) * MOVE_VELOCITY * time / 1000;
    let posY = (Camera.state.moveRight - Camera.state.moveLeft) * MOVE_VELOCITY * time / 1000;
    let rotX = (Camera.state.rotateUp - Camera.state.rotateDown) * ROT_VELOCITY * time / 1000 / 180 * Math.PI;
    let rotY = (Camera.state.rotateRight - Camera.state.rotateLeft) * ROT_VELOCITY * time / 1000 / 180 * Math.PI;
    if (posX || posY)
        Camera.move(posX, posY);
    if (rotX || rotY)
        Camera.rotate(rotX, rotY);

    for (let loader of loaders) {
        loader.render(now);
    }

    window.requestAnimationFrame(tick);
};
tick();
```

二、运行方式及效果

W、A、S、D 分别是向前、向左、向后、向右移动

I、K、J、L 分别是视角向上、向下、向左、向右

F 相机点光源的打开与关闭

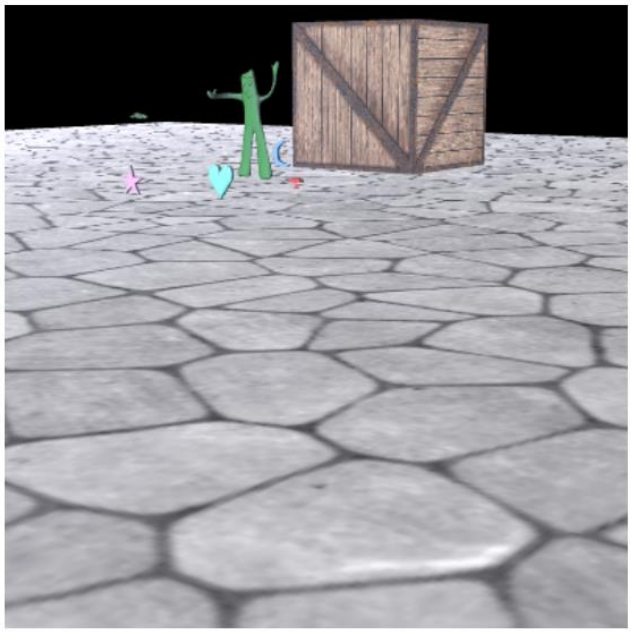


W、A、S、D分别是向前、向左、向后、向右移动

I、K、J、L分别是视角向上、向下、向左、向右

F相机点光源的打开与关闭

运行后截图



W、A、S、D分别是向前、向左、向后、向右移动

I、K、J、L分别是视角向上、向下、向左、向右

F相机点光源的打开与关闭

三、开发运行环境

运行：Chrome

开发：intelliJ

四、感想

这次 pj 学会了 3D 场景贴图及漫游。由于 Javascript 是弱类型的编程语言，所以在开发过程中出现了许多这方面的错误。这个 pj 很有趣，收获很多，希望这个课程越办越好。