中山大学软件学院 **2011** 级软件工程专业(2013 秋季学期)

《数字媒体技术》 期末大作业

(考试形式：大作业　　时间：3 周)

警示

《中山大学授予学士学位工作细则》第六条

考试作弊不授予学士学位

方向：数字媒体技术　　姓名：黄孝楠　　学号：11331136

出卷：孙 伟　　审核：

大作业基本要求：

完成一个可交互的虚拟场景展示系统。系统中场景可以选择和中山大学相关的学习和生活场景。场景由图像背景和三维物体对象组成，交互时用鼠标点击三维物体对象的某个组成元素，弹出视频播放窗口，播放一段自己录制并配音的视频。拖动鼠标，可以观看不同角度的场景。

大作业设计与开发要求

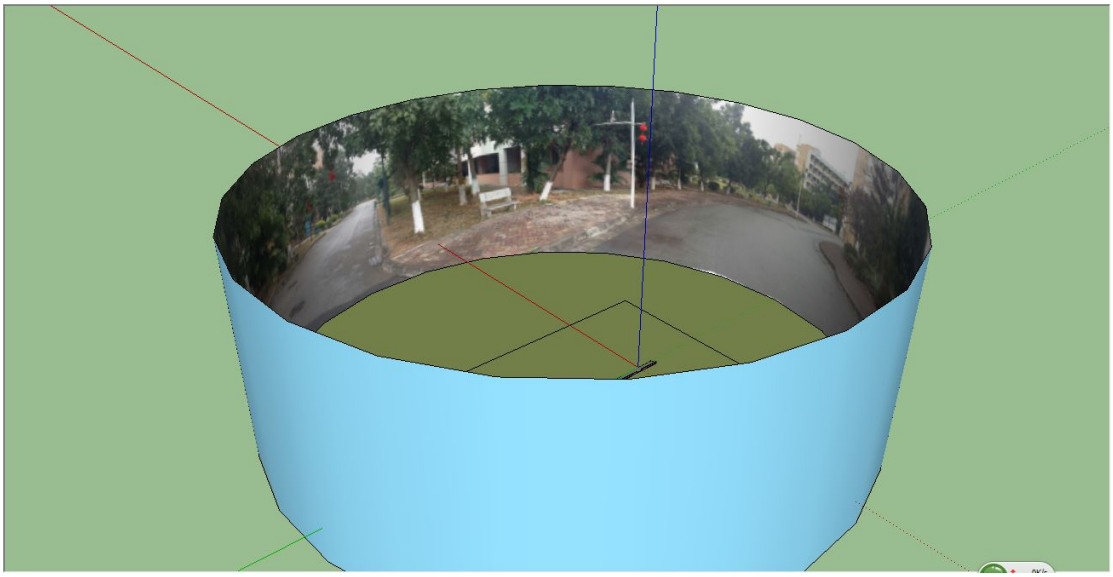
1） 背景图像；要求为至少 180 度可浏览拼接图像，可参考平时作业中的拼接方法或工具进行拼接。考核点为图像处理技术；分值 20 分。

学校里拍了一些照片:

用 photomerge 进行了拼接，图片在 image 文件夹里面，结果如下

获得全景图以后，我对全景图进行了一些调整。
要将全景图当成背景，我利用了一个比较大的圆柱体：

如图：



全景图在 demo 里面的效果：

2） 三维物体对象；要求使用 3D 建模工具建模。不能太简单，要体现一定的技巧和工作量。考核点为计算机图形和 3D 技术；分值 20 分。

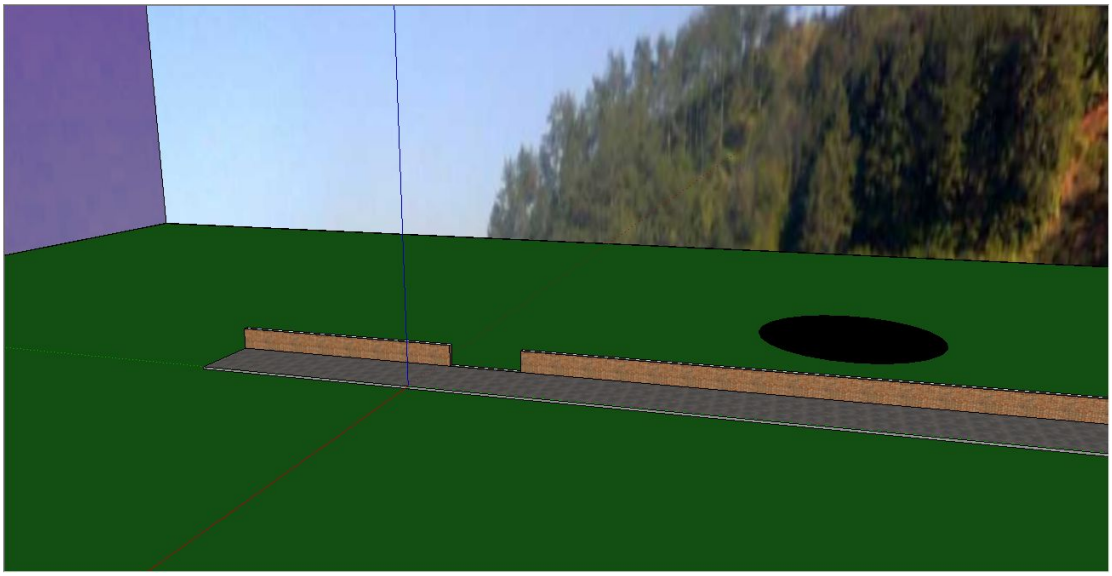
我使用了 SketchUp 进行基本建模和贴图，然后用 blender 和 threejs 进行 json 文件的转换（因为 threejs 提供的插件不够完善，我对转换后的 json 模型文件进行了一些修正）。

模型在 model 里面，格式为 json。

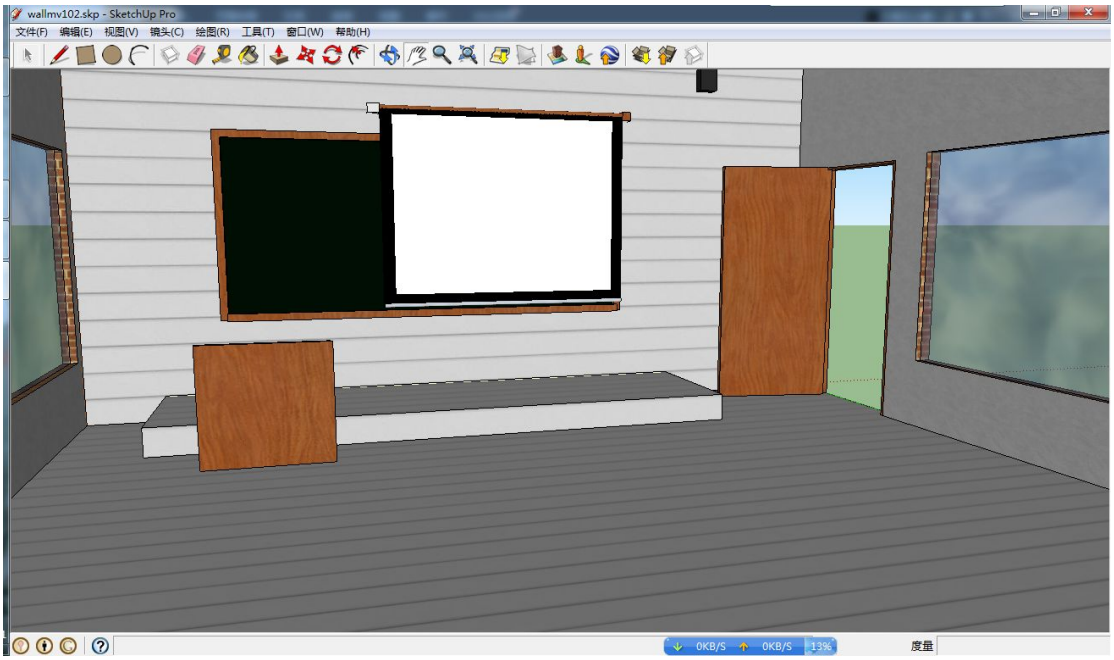
因为交互系统模型比较多，还涉及到交互行为，所以我将整体的模型都分成了比较多的 part，然后在 web 端进行整合和调整，可以比较好的提高渲染性能和质量。

Sketchup 建模：

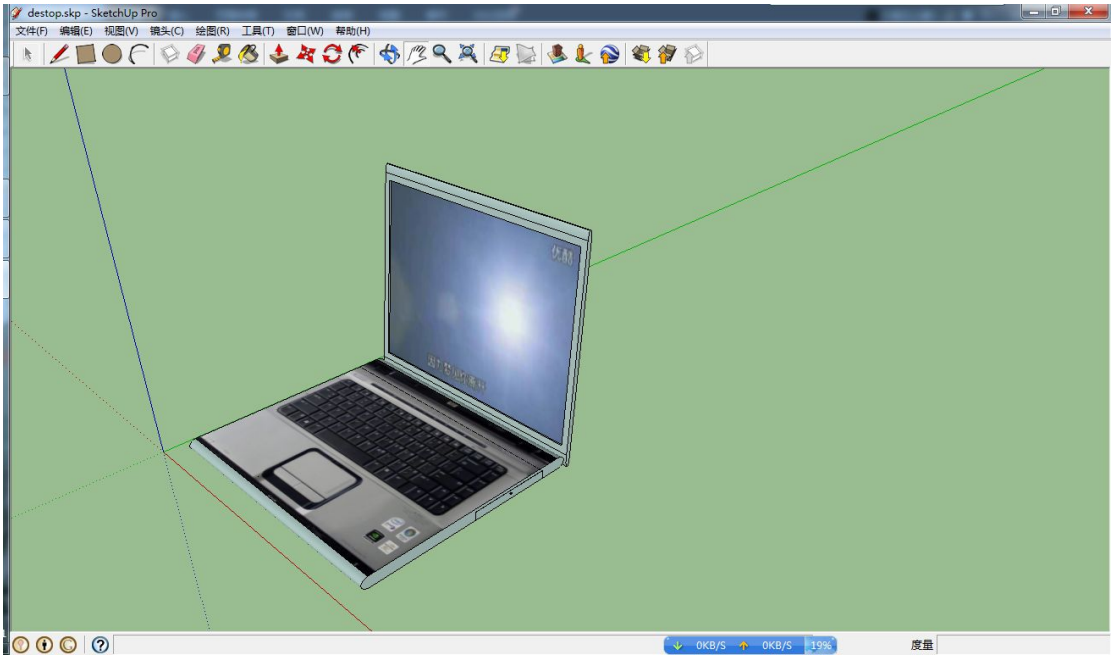
走廊：



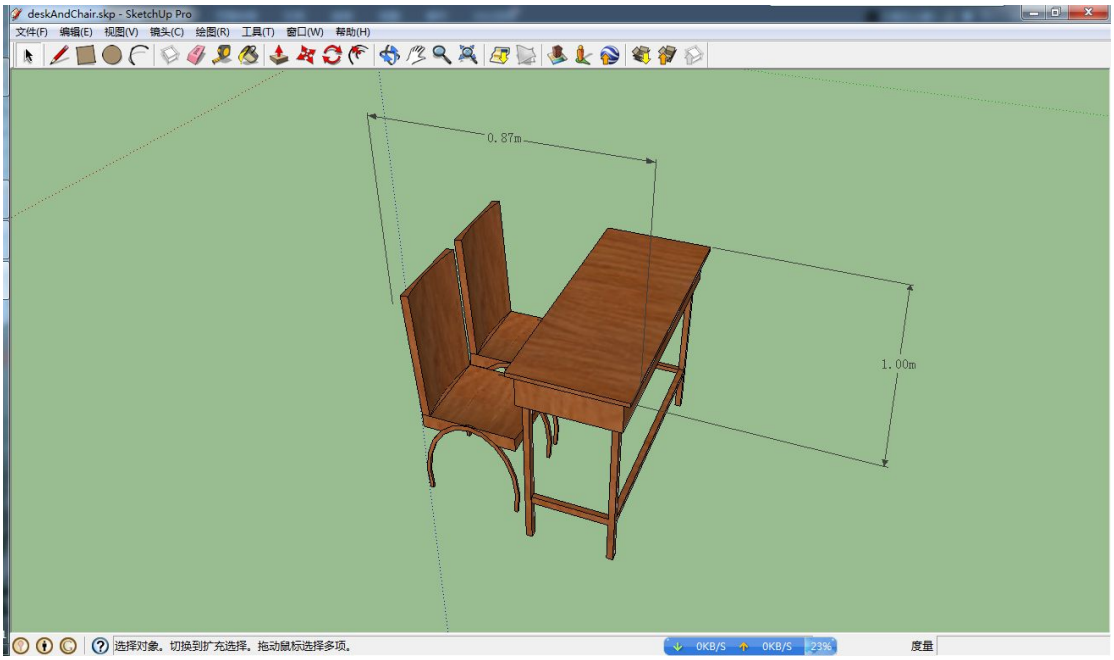
教室：



笔记本电脑：



桌椅：



因为需要进行 web 平台展示，所以我使用的 sketchup 的模板单位和 web 端的全部都是以真实世界的单位和尺寸，以便在集成元素的时候可以统一。

3） 自己录制并配音的视频；要求自己录制视频和声音，内容要有设计理念与场景相匹配。将编辑好的视频用某种编码格式存储。考核点为视频、音频处理技术；分值 20 分。

时间仓促，没时间录制配音视频，我在优酷上下了个视频，然后工具剪辑成一小段（1.65M，不会消耗太多服务器资源），

我尝试过将视频作为纹理贴到一个面上，可是水平有限，threejs 文档缺乏，尝试多次都没成功，所以只好将视频放到了 html 里面浮动。

视频为 MP4 格式，采用了 html5 规范的 <video> 标签进行展示，优点是无需任何插件，缺点是对老旧浏览器的兼容性不够好。

4） 用 siverlight、flash、openGL（不限于此）等工具集成 1）--3）等数字媒体元素，形成可交互的虚拟场景展示系统。考核点为数字媒体集成技术；分值 40 分。

工具：three. js，webgl（JavaScript 和 OpenGL ES 2.0 的结合）。Jquery（方便地输出各种调试信息）

我先在本地用 firefox 进行开发调试，然后将 demo 部署到了 SAE 上。

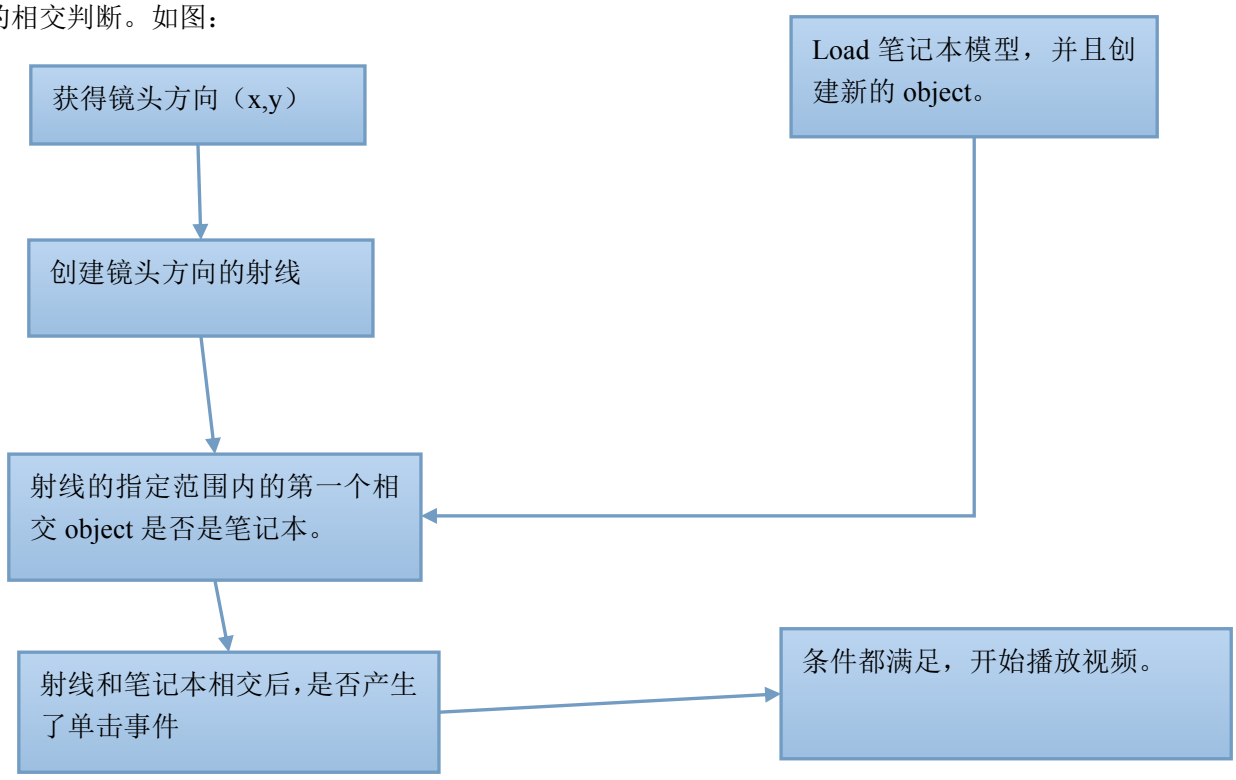
考虑到网页加载的速度问题，我将整个场景的模型分成了各个部分，比如我将桌子这个重复出现的模型只进行了一次建模，然后在 js 里面进行了多次调用，达到展现多个桌子，加快加载速度。

核心系统构建：

首先将 camera 和一个 object 进行绑定，进行镜头控制：

交互系统构建：

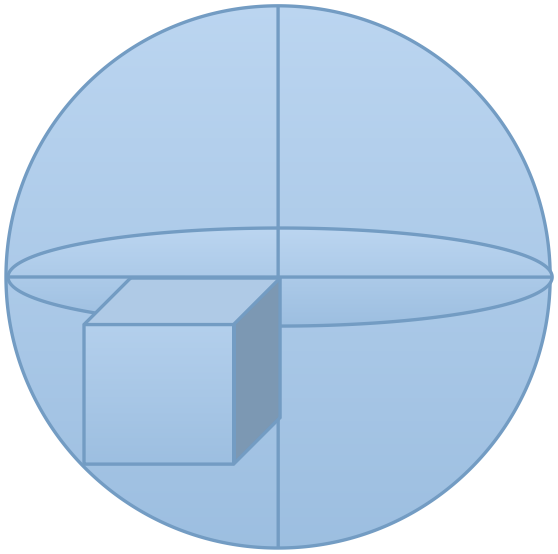
首先是要获取镜头的方向，方向分为水平方向和垂直方向。获得方向后进行射线和 object 的相交判断。如图：



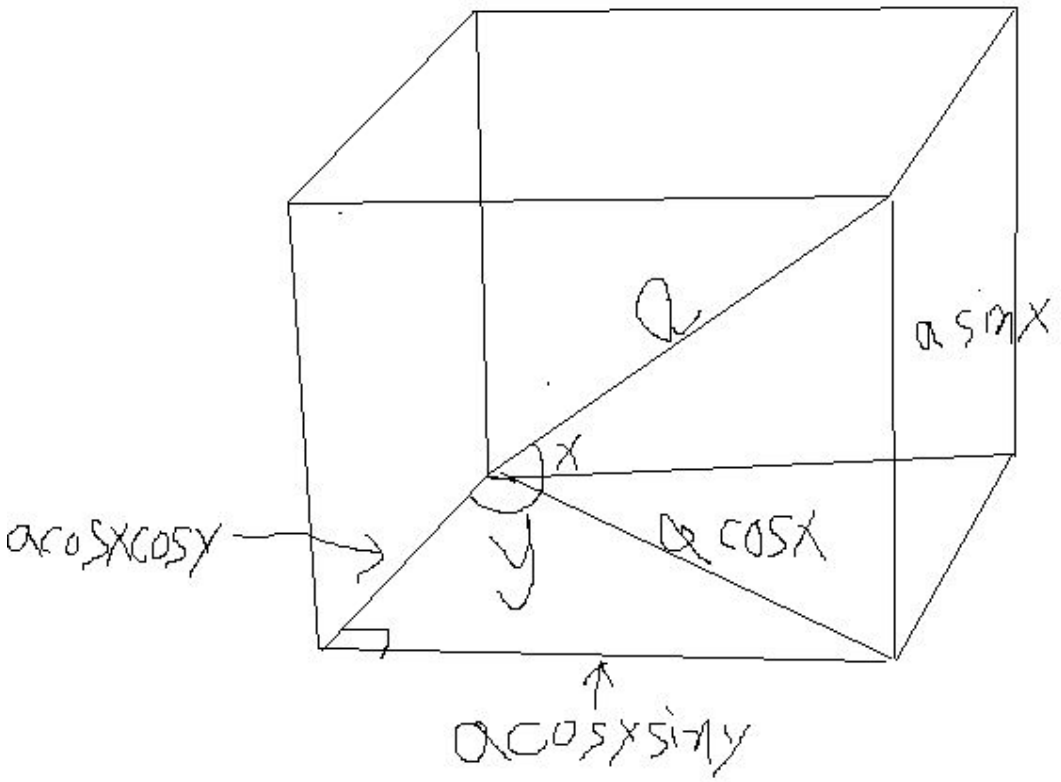
其中获得镜头方向是比较困难的一件事情，threejs 里面对方向的构建比较繁琐：
Direction 是一个三维坐标来表示的，需要自己从角度转换。

direction — The direction vector that gives direction to the ray.

这是镜头方向模型



然后简化成两个三角形：



通过坐标换算，将两个角度换算成一个 direction 坐标，然后进行射线构建：

Raycaster

This class makes raycasting easier. Raycasting is used for picking and more.

Constructor

Raycaster(origin, direction, near, far) {

origin — The origin vector where the ray casts from.
direction — The direction vector that gives direction to the ray.
near — All results returned are further away then near. Near can't be negative. Default value is 0.
far — All results returned are closer then far. Far can't be lower then near . Default value is Infinity.

This creates a new raycaster object.

这样，将镜头作为 origin，刚刚获得的 vector 方向作为 direction，然后设置交互的有效距离 near 和 far，就可以建立一个 pickup 的射线，这个射线会随着 object 的渲染而实时更新，这样才能达到效果：
下面这个可以将笔记本 object 作为 object，然后用这个射线的方法进行拾取判断。

.intersectObject(object, recursive)

object — The object to check for intersection with the ray.
recursive — If set, it also checks all descendants. Otherwise it only checks intersecton with the object.

checks all intersection between the ray and the object with or without the descendants.

假如这个 intersectObject 检查到了笔记本 object，那么就进行单击事件的监听：
onclick="playVideo();
这样，上述所有的条件都检查通过，那么在下一次渲染的时候（0.03S 左右后），我就可以开始 play video 了。这样一个初步的交互系统就构建完毕了。

物理系统的构建：

物理系统的碰撞等，我进行了简单的距离判断，结合上述的镜头角度进行这个系统的构建：
下面选取前方的物体碰撞进行构建说明：
首先利用上面的方法，
1 获得镜头方向的判断
2 构建射线
3 获得与射线相交的物体
4.获得这个物体和 origin 的最短距离
5.比较最短距离是否已经达到碰撞的距离
6.达到的话就禁用前进
7.未达到就可以继续前进

下面是具体的是否可以前进的检查函数：

```
    this.isForwardObject = function () {
        var co = -Math.cos(yawObject.rotation.y);
        var si = -Math.sin(yawObject.rotation.y);
        ray.ray.direction.set(si, 0,co);//对正前方进行判断
/*      $("#info6").text("x:" + si);
        $("#info7").text("z:" + co);
*/

//      $("#info3").text("(x:" + si + ",z:" + co + ")");
        ray.ray.origin.copy(yawObject.position);
        //ray.ray.origin.z -= 10;
        var intersections = ray.intersectObjects(this.objects);

        if (intersections.length > 0) {
            var distance = intersections[0].distance;
            if (distance > 0 && distance < 1) {
                isForwardObject = true;
                canMoveForward = false;
                moveForward = false;
                return;
            }
        }
        ray.ray.direction.set(si, -1,co);//对正前方 脚下-45度进行判断
        intersections = ray.intersectObjects(this.objects);
        if (intersections.length > 0) {
            distance = intersections[0].distance;
            if (distance > 0 && distance < 1.6) {
                isForwardObject = true;
                canMoveForward = false;
                moveForward = false;
                return;
            }
        }



        isForwardObject = false;
        canMoveForward = true;

    };
```

同理构建 8 个方向的检查（左右方向会更加复杂一些）：
下方向检查：

```
    this.isOnObject = function () {
        ray.ray.direction.set(0, -1, 0);
        ray.ray.origin.copy(yawObject.position);
        ray.ray.origin.y -= 10;
        var intersections = ray.intersectObjects(this.objects);
        if (intersections.length > 0) {
            var distance = intersections[0].distance;
            if (distance > 0 && distance < 1.7) {
                isOnObject = true;
                canJump = true;
                return;
            }
        }
        isOnObject = false;
        canJump = false;

    };
```

后方向检查：

```
this.isBackwardObject = function () {
    var backRotation = yawObject.rotation.y + Math.PI;//偏移180度
    var co = -Math.cos(backRotation);
    var si = -Math.sin(backRotation);
    ray.ray.direction.set(si, 0,co);
    ray.ray.origin.copy(yawObject.position);
    var intersections = ray.intersectObjects(this.objects);

    if (intersections.length > 0) {
        var distance = intersections[0].distance;
        if (distance > 0 && distance < 1) {
            isBackwardObject = true;
            canMoveBackward = false;
            moveBackward = false;

            return;
        }
    }
    ray.ray.direction.set(si, -1,co);
    intersections = ray.intersectObjects(this.objects);

    if (intersections.length > 0) {
        distance = intersections[0].distance;
        if (distance > 0 && distance < 1.7) {
            isBackwardObject = true;
            canMoveBackward = false;
            moveBackward = false;

            return;
        }
    }
    isBackwardObject = false;
    canMoveBackward = true;

};
```

左方向检测：

```
this.isLeftObject = function () {
    var backRotation = yawObject.rotation.y + Math.PI / 2;
    var co = -Math.cos(backRotation);
    var si = -Math.sin(backRotation);
    ray.ray.direction.set(si, 0,co);
    ray.ray.origin.copy(yawObject.position);
    //ray.ray.origin.z -= 10;
    var intersections = ray.intersectObjects(this.objects);

    if (intersections.length > 0) {
        var distance = intersections[0].distance;
        if (distance > 0 && distance < 1) {
            isLeftObject = true;
            canMoveLeft = false;
            moveLeft = false;
            return;
        }
    }
    ray.ray.direction.set(si, -1,co);
    intersections = ray.intersectObjects(this.objects);
    if (intersections.length > 0) {
        distance = intersections[0].distance;
        if (distance > 0 && distance < 1.7) {
            isLeftObject = true;
            canMoveLeft = false;
            moveLeft = false;
            return;
        }
    }
    isLeftObject = false;
    canMoveLeft = true;

};
```

右方向检测：

```
this.isRightObject = function () {
    var backRotation = yawObject.rotation.y - Math.PI / 2;
    var co = -Math.cos(backRotation);
    var si = -Math.sin(backRotation);
    ray.ray.direction.set(si, 0,co);
    ray.ray.origin.copy(yawObject.position);
    //ray.ray.origin.z -= 10;
    var intersections = ray.intersectObjects(this.objects);

    if (intersections.length > 0) {
        var distance = intersections[0].distance;
        if (distance > 0 && distance < 1) {
            isRightObject = true;
            canMoveRight = false;
            moveRight = false;
            return;
        }
    }
    ray.ray.direction.set(si,-1,co);
    intersections = ray.intersectObjects(this.objects);
    if (intersections.length > 0) {
        distance = intersections[0].distance;
        if (distance > 0 && distance < 1.7) {
            isRightObject = true;
            canMoveRight = false;
            moveRight = false;
            return;
        }
    }
    isRightObject = false;
    canMoveRight = true;

};
```

上方检查：

```
this.isUnderObject = function () {

    ray.ray.direction.set(0, 1,0);
    ray.ray.origin.copy(yawObject.position);
    //ray.ray.origin.z -= 10;
    var intersections = ray.intersectObjects(this.objects);

    if (intersections.length > 0) {
        var distance = intersections[0].distance;
        if (distance > 0 && distance < 0.5) {
            velocity.y = -0.6;
            isUnderObject = true;
            return;
        }
    }
    isUnderObject = false;

};
this.getDesktop = function(desktop){
    this.desktop = desktop;
};
```

这样构建了 8 方向的简单物理碰撞逻辑，就可以避免穿墙等等的发生了。

最终的 demo：

Demo 操作说明：
单击进入场景
esc 退出
w/向上:前进;
s/向下:后退
a/向左:往左走
d/向右:往右走
鼠标控制视野

交互：
靠近笔记本，准星单击笔记本电脑，会弹出视频，再次单击笔记本关闭。

核心代码是（js 文件夹下）：
PointerLockControls.js（camera 控制）
smmain.js（scenejs 加载）

demo 演示：
http://ving.sinaapp.com/

如果网站无法打开，用 firefox 本地运行 index.html 即可。
本地用 chrome 运行 demo 会遇到跨域问题，请不要使用。
Demo 运行截图：


说明：
单击进入场景
esc 退出
w/向上:前进;
s/向下:后退
a/向左:往左走
d/向右:往右走
鼠标控制视野
靠近笔记本，准星单击笔记本电脑，会弹出视频，
再次单击笔记本关闭(在chrome下有效)
FPS:13
fps比较低，配置低于渲染要求，画面卡顿是正常的


说明：
单击进入场景
esc 退出
w/向上:前进;
s/向下:后退
a/向左:往左走
d/向右:往右走
鼠标控制视野
靠近笔记本，准星单击笔记本电脑，会弹出视频，
再次单击笔记本关闭(在chrome下有效)
FPS:29
fps比较低，配置低于渲染要求，画面卡顿是正常的
sinaapp.com 现以全屏显示。
随时可以按 ESC 键退出。


《数字媒体技术》期末大
ving.sinaapp.com/with-texture/digital-video-final-project.html
说明：
单击进入场景
esc 退出
w/向上:前进;
s/向下:后退
a/向左:往左走
d/向右:往右走
鼠标控制视野
靠近笔记本，准星单击笔记本电脑，会弹出视频，
再次单击笔记本关闭(在chrome下有效)
FPS:14
fps比较低，配置低于渲染要求，画面卡顿是正常的
优酷
0:40
0KB/S
0KB/S
66%


《数字媒体技术》期末
ving.sinaapp.com/with-texture/digital-video-final-project.html
说明：
单击进入场景
esc 退出
w/向上:前进;
s/向下:后退
a/向左:往左走
d/向右:往右走
鼠标控制视野
靠近笔记本，准星单击笔记本电脑，会弹出视频，
再次单击笔记本关闭(在chrome下有效)
FPS:13
fps比较低，配置低于渲染要求，画面卡顿是正常的
优酷
随往事慢慢飘散
0KB/S
0KB/S
64%

参考资料：

http://threejs.org/

http://www.w3school.com.cn/index.html