Escape WEB3D 密室逃脱 设计文档

陈昕婕 15302010057 刘雨馨 15302010043

耿同欣 15302010048 何宇雯 15302010042

零、项目环境

项目基于 nodejs v8.1+，包括但不限于 express,threejs,socket.io 等库的使用。

首先项目基于 webpack，在本地运行时我使用了 webpack 的 development 模式并且启动了 source-line 便于 debug。在上线时我改成了 production 模式，在 public 目录底下生成[name]_bundle.js 等静态文件。并将整份静态文件上传至服务器。

其次我讲服务器 server 的代码也上传至服务器，通过 npm 安装 forever 库，在后台持久运行 server 程序，并且将 error 信息，运行信息等输出到相应 log 文件里。

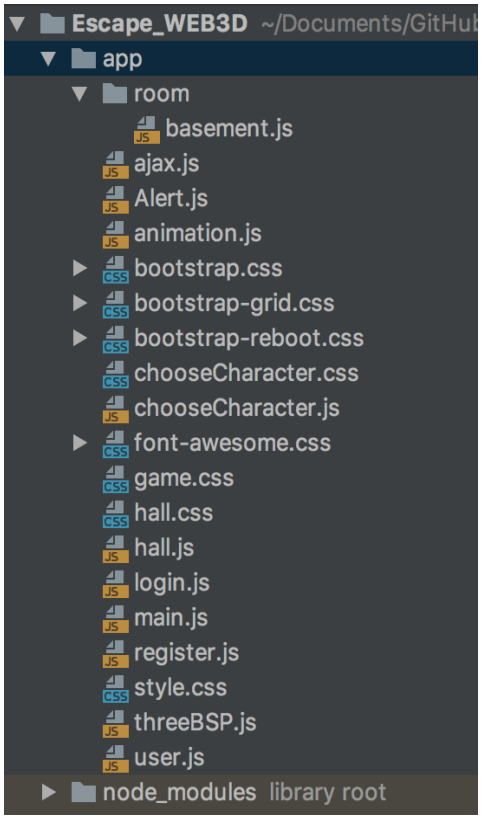
接着配置好 neo4j，只需要通过浏览器访问 7474 端口开启即可。

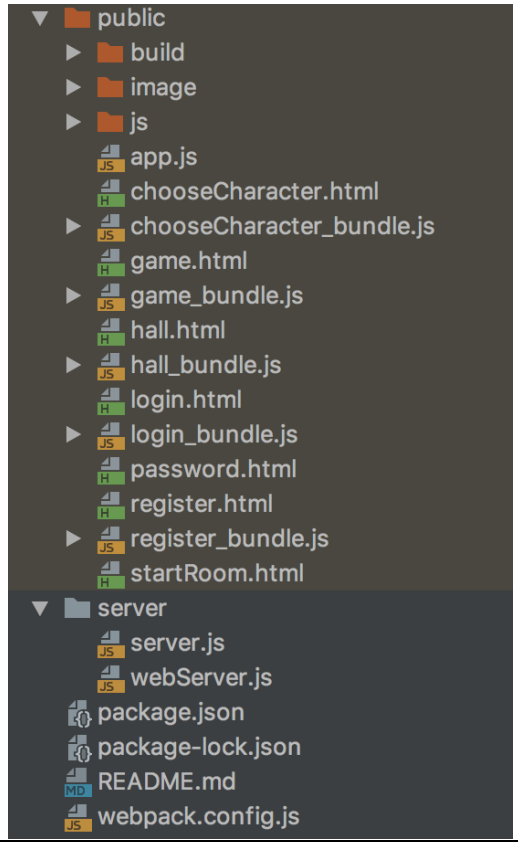
由于亚马逊的流量控制，在配置的过程中还麻烦了戴老师开启了我们需要的端口。至此完成了密室逃脱项目的线上部署。

服务器基本配置：Ubuntu 系统，Nodejs 环境、NEO4J 图数据库

网址：http://52.83.171.185:9753

一、项目组织以及其中每个文件的说明





app	client 端相关 css 文件和 js 文件存放目录
room	
basement.js	密室逃脱地下室模型加载
Alert.js	用户名长度限制
animation.js	场景加载/场景交互/socket.io 客户端
chooseCharacter.css	选择人物页面的布局
game.css	游戏页面的布局
hall.css	游戏大厅页面的布局
style.css	登录注册页面的布局
user.js	人物类（人物模型、人物动画、人物信息更新）
public	client 端 html 文件和 webpack 打包后的 js 文件存放目录

app.js	处理网页请求的服务器
chooseCharacter.html	选择人物页面
chooseCharacter_bundle.js	选择人物页面经过 webpack 打包后的 js 文件
game.html	游戏页面
game_bundle.js	游戏页面经过 webpack 打包后的 js 文件
hall.html	游戏大厅页面
hall_bundle.js	游戏大厅页面经过 webpack 打包后的 js 文件
login.js	登录页面
login_bundle.js	登录页面经过 webpack 打包后的 js 文件
password.html	修改密码页面
register.html	注册页面
register_bundle.js	注册页面经过 webpack 打包后的 js 文件
startRoom.html	创建新的游戏房间页面
server	服务器目录
server.js	socket.io 服务器
webserver.js	服务器端总入口
webpack.config.js	webpack 配置文件

二、关键功能实现细节

public/app.js

1、登录、注册、选择角色、新建房间、加入房间、登出跳转页面：用到node.js 中的 express 框架：

app.get('/', function (req, res) {res.render(__dirname + "/" + "login.html");})

express 接收到"/"路由，转入登录页面。

app.get('/login', function (req, res)

express 接收到"/login"路由，接收前一页面表单数据，连接数据库，进行注册操作。

app.post('/register', function (req, res)

express 接收到"/register"路由，接收前一页面表单数据，连接数据库，进行注册操作。

选择角色、新建房间、加入房间、登出功能同样运用 express 框架接收前页面 http 路由请求的方式转入该功能。

2、neo4j 数据库操作

2.1 注册、登录

注册：

首先'match(n:User) where n.name={name} return n'，判断图数据库中是否已存在该用户名，若不存在，则'CREATE (somebody:User { name: {name}, password: {password}, gender:0}) RETURN somebody'，把该用户的用户名和密码写入 neo4j 图数据库。

登录：

'match(a:User) where a.name={name} and a.password={password} return

a.gender'，在图数据库中查询是否有对应用户信息。

2.2 新建房间

首先'match(n:Room) where n.id={id} return n'，查询该房间名是否已经存在，若不存在，'CREATE (room:Room { id: {id}, num:1, user1:{user1}，user2:null, user3:null, status:0}) RETURN room'，创建新房间（标签为 Room），设置房间名（属性 id），目前人数（属性 num）设为 1，并把房间创建者的名字存储为属性 user1，房间状态为准备中（属性 status：0 表示准备中，1 表示已开始游戏）

2.3 加入房间

首先'match(room:Room) where room.id={id} return room.num'，确认对应房间存在且在准备中，然后'match(room:Room) where room.id={id} and room.user1 is null return room.num'，然后依次查询 user1，user2，user3 是否为空，第一个查询到为空的把对应属性值设置为当前用户名，如 'match(room:Room) where room.id={id} set room.num={num},room.user1={user1}',

server/server.js：服务器端使用 socket.io 与客户端进行交互

1、新建房间

服务器端包含两个全局变量，map 对象 map 和 gameuser。

io.on('connection', function (socket) 服务端监测玩家连接， socket.on('room', function (data) 服务器监测到客户端新建房间发出的 room 接口，在该函数中初始化变量 map[roomId]和 gameuser[roomId]，在其中存放游戏开始后房间

内要存储的信息，在 map[roomId]中存放游戏信息，例如钥匙是否被拿走，isKey: true，拿走钥匙的玩家 candleUser: null，蜡烛是否被拿走，拿走蜡烛的玩家，输入密码的玩家等。在 gameuser[roomId]中存放玩家信息。

2、加入房间

服务器端监测客户端 join 接口，socket.on('join', function (data)，在函数中将用户信息存入 gameuser[roomid][username]，socket.join(data.room)用户加入该房间，socket.broadcast.to(data.room).emit('connection', joindata);在该房间其余客户端广播该玩家位置信息，socket.emit('connection', gameuser[roomid][k]);向本服务端发送其他玩家信息。

在 socket.on('join', function (data)该函数中进行游戏过程中其他与客户端交互的操作。

3、开始游戏

```
if (number == 3 && map[roomid].isStart === false) {
        map[roomid].isStart = true;
        io.to(data.room).emit('start', map[roomid].isStart);
}
```

判断加入房间人数为 3，以及 map[roomid].isStart 为 false 后，将 isStart 设为 true，发送给房间所有客户端 start 接口和 isStart 字段。

4、更新玩家位置

socket.on('update', (data)

服务器端监测客户端 update 接口，接收玩家位置信息，将位置信息广播给房间其余客户端。

5、捡起钥匙/捡起蜡烛/输入密码操作

服务器端监测客户端 key 接口，接收捡起钥匙的玩家用户名，判断 map[roomid].isKey === true && map[roomid].keyUser == null 后更新 map[roomid].isKey = false;map[roomid].keyUser = data;再将捡起钥匙的玩家信息广播给房间内所有客户端。

捡起蜡烛/输入密码操作原理与捡起钥匙相同。

6、聊天操作

服务器端监测客户端 chat 接口，接收该玩家发送的聊天信息，服务器端发送给房间内其余客户端该玩家用户名及聊天内容。判断信息为"compliance will be rewarded" 且游戏开始后，服务器端广播给房间内所有客户端 hint 接口，返回提示内容的字段。

7、开门成功逃脱操作

服务器端监测客户端 door 接口，接收开门的玩家用户名，判断密码锁已被打开 map[roomid].isCode === true 以及 data === map[roomid].keyUser 判断开门的玩家是否为捡起钥匙的玩家以避免玩家修改 js 代码，随后服务器端广播给房间内所有客户端 win 接口。

8、玩家退出游戏

socket.on('disconnect', function ())

进行数据库操作

'match(n:Room) where n.id={id} return n.user1,n.user2,n.user3,n.num'，查询房间中各个用户名和房间人数，然后依次判断当前用户是 user1/user2/user3 中的哪个，把对应属性设为 null，然后 num 属性减一，如'match(room:Room)

where room.id={id} set room.user1=null, room.num={num}'（{num}中的num 是减一后的值）。

当 num 减一后为 0 时，删除数据库中的房间信息，'match(room:Room) where room.id={id} delete room'。

THREEJS 部分

1. 模型加载

静态的比如书本书桌等模型为 obj 格式+mtl 样式文件，动态的人物模型为 fbx 格式，人物可进行基本的行走动画。这里实现的原理涉及到人物的骨骼模型 bone 的节点动画。

2. 碰撞检测

采用 threejs 封装好的 raycaster 类，由中心物体向四周发射射线，对不可穿透物体检测，如果射线长度小于阈值（我设定的 40），即判断碰撞。对用户每一次前进都进行碰撞检测，一旦发现碰撞则回滚上一步行走，实现人物无法走进模型。

3. 物体拾取

同样是使用了 threejs 的 raycaster 类。从相机像鼠标点击的点发射射线，如果射线遇到指定物体，则判断点到了指定物体，然后启动对应执行函数（比如捡起蜡烛，弹出密码框，开门等）。

其中涉及到屏幕坐标和 3D 坐标系的转换，这其中的原理主要还是坐标系的矩阵变化。

4. 物体补间动画

用到了 threejs 的衍生库 tween.js 实现指定物体的始末状态，自动补全中间动画。当用户成功开门的时候，门有一个旋转 90 度的动画，代码中指定了始末状态的 rotation.y 的值，根据库在渲染时调用 TWEEN.update()实现动画的更新。

具体代码：

```
new TWEEN.Tween(pickObject[4].rotation).to({z: Math.PI /
2},2000).easing(TWEEN.Easing.Elastic.Out).start();
```

5. 物体裁剪

墙等模型使用了 cubeGeometry 类，并通过 texture 贴图。使用了 THREEBSP.js 的衍生库实现对立方体模型的裁剪，实现墙中间挖了个门。

具体代码：

```
    var cut = new ThreeBSP(door);
    var resultBSP = cutWall.subtract(cut);
    var result = resultBSP.toMesh(wallMat);
    result.material.flatshading = THREE.FlatShading;
    result.geometry.computeFaceNormals();  //重新计算几何体侧面法向量
    result.geometry.computeVertexNormals();
    result.material.needsUpdate = true;  //更新纹理
    result.geometry.buffersNeedUpdate = true;
    result.geometry.uvsNeedUpdate = true;
```

6. 权威服务器的实现

为了防止一些多用户并发的情况（例如两个用户同时捡起钥匙造成的冲突），将

部分游戏逻辑归至服务器维护（例如密码锁状态，钥匙所属状态，蜡烛所属状态，游戏状态等），客户端只负责将"捡起钥匙"请求发给服务器，由服务器判断最终捡起钥匙的用户并且广播告知同房间所有玩家。

三．项目中使用的其他技术要点

1. Webpack 的使用

整个客户端基于 npm 的 webpack 模块搭建，编写程序时使用 development 模式并且 source-inline 配置进行程序调试。

使用 webpack，由一个总入口引入多个 js 文件，最终生成 bundle.js 文件进行加载，在配置上能够加快文件的加载速度。

2. Neo4j 图数据库

使用图数据库，使得用户以及用户对应的房间信息更加直观。

3. TWEEN.JS

实现 3D 模型中的补间动画加载。

4. THREE.JS

使用基于 WEB GL 的库，使得 3D 开发更加便捷。

5. THREEBSP.JS

实现模型的编辑，裁剪等。

6. Bootstrap 框架

几乎所有网页的控件布局等使用了 bootstrap 的框架类，使得 UI 美观适配性好。

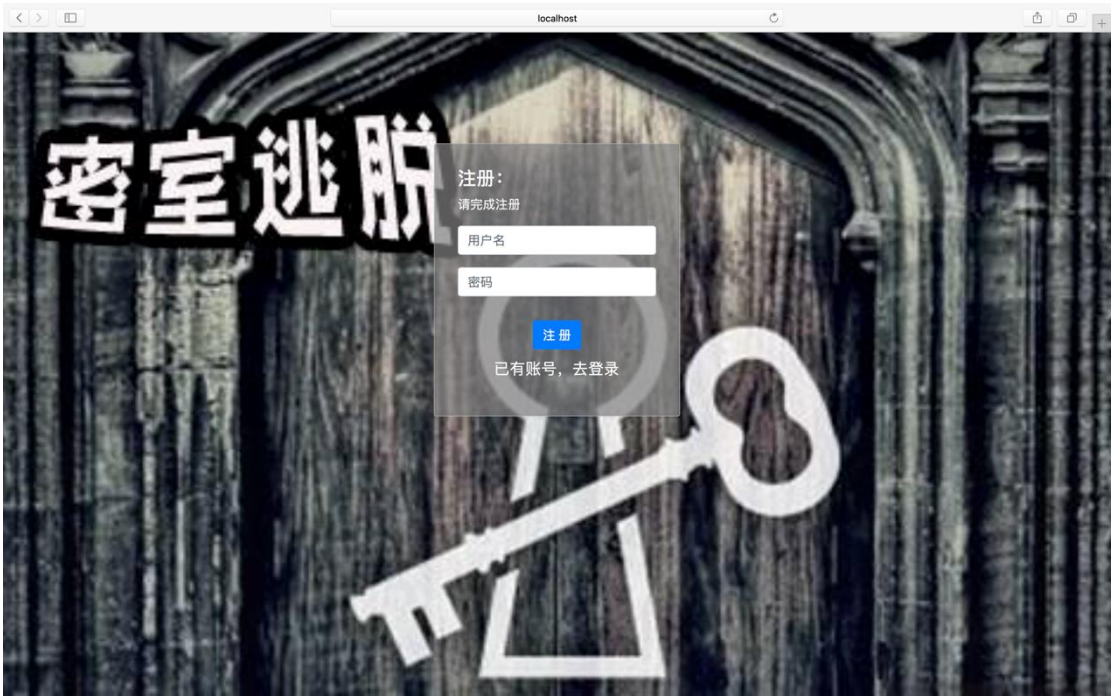
7. 路由控制

在网页加载和跳转的过程中，我们使用了 express 的 app.use()和 app.get()方法

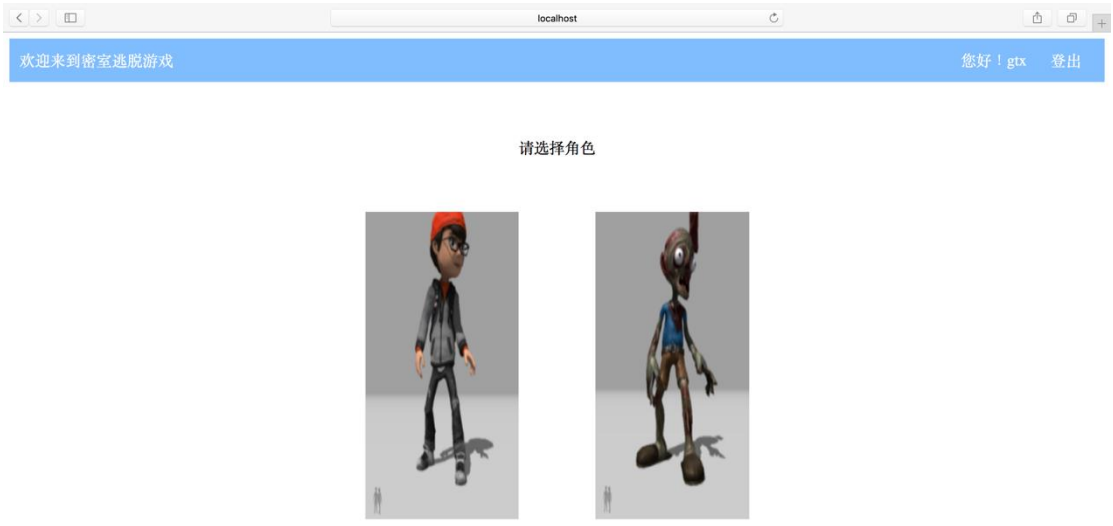
实现了路由控制。通过接收请求访问到 webpack 的静态目录。

系统操作说明

1. 在浏览器打开网址 http://52.83.171.185，跳转入 注册/登录 入口页面。

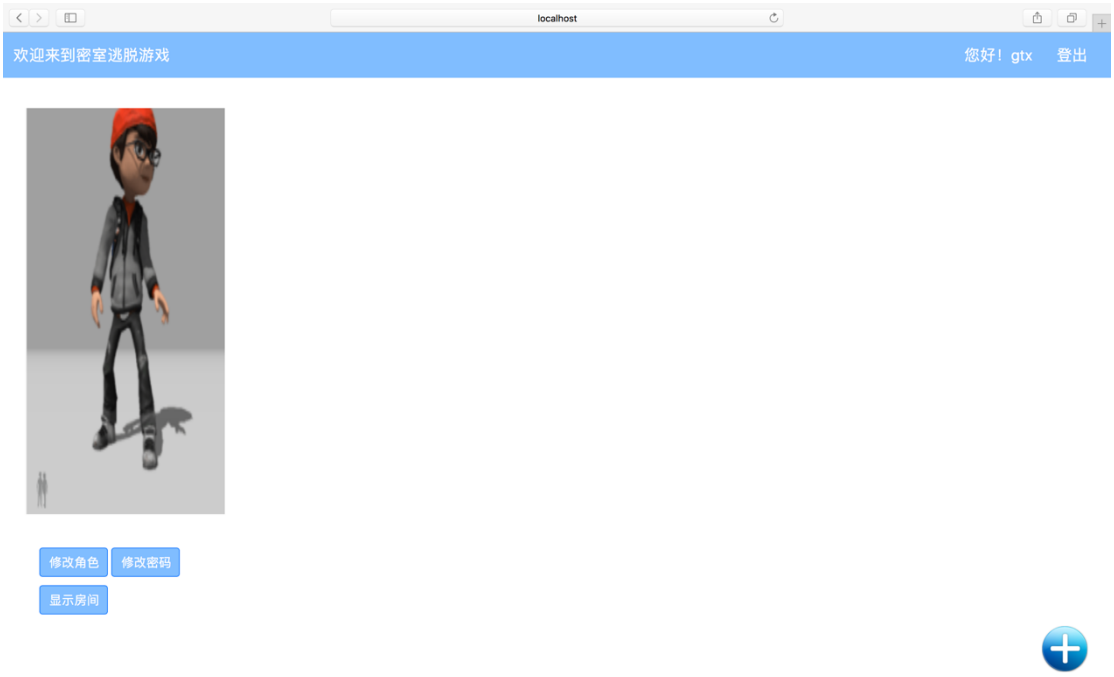


2. 输入用户名和密码，完成注册/登录



3. 注册后进入选择角色页面（如上图），点击图片选择角色，进入游戏大厅页

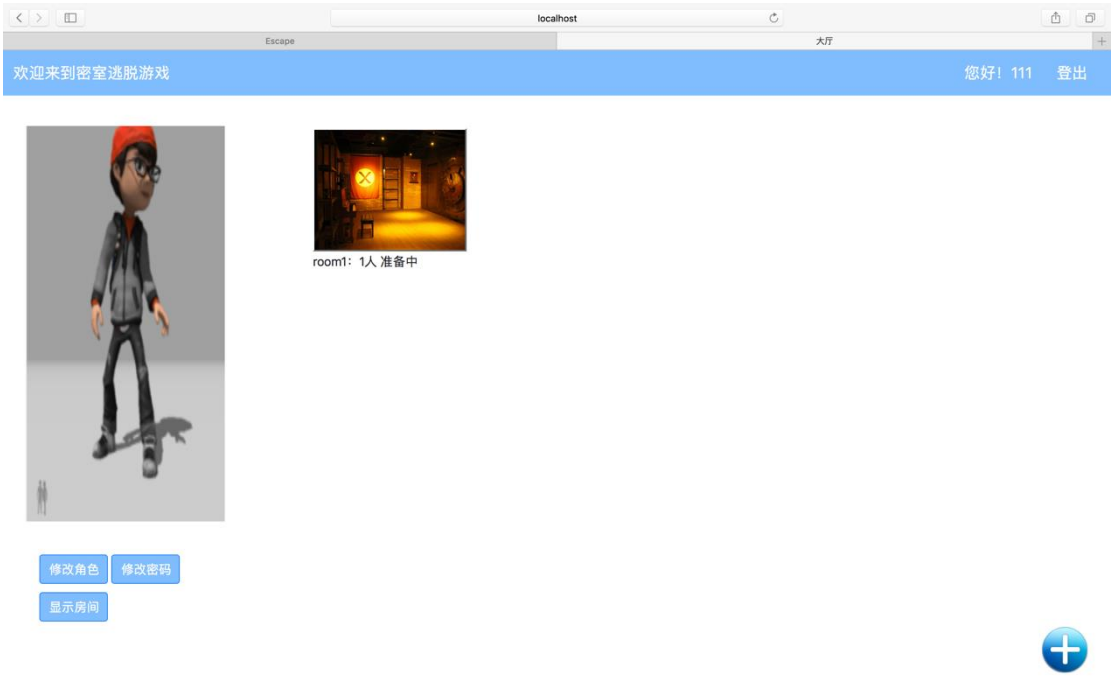
面（如下图）



4. 点击右下角的加号图片可以创建新房间（见 5），点击左下角“显示房间”按钮可以查看已经创建的房间（见 6），点击“修改角色”按钮可以重新选择角色（见 7），点击“修改密码”按钮可以修改登录密码（见 8）

5. 创建新房间页面如下图，输入房间名并点击“设置”按钮，进入游戏页面

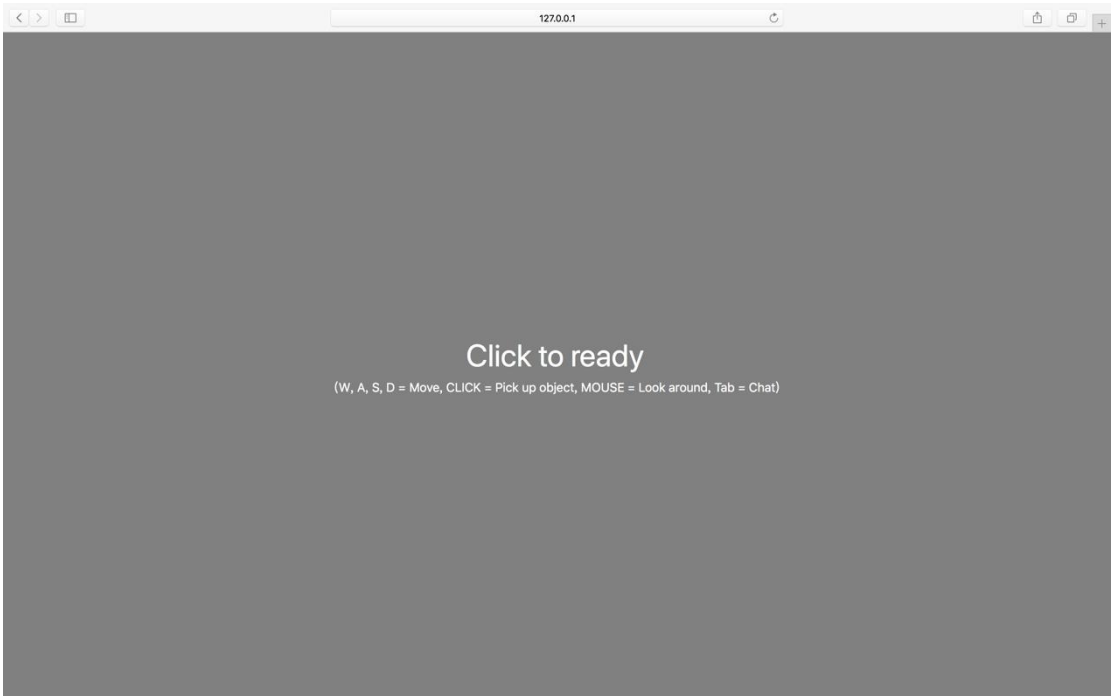
6. 点击“显示房间”按钮后，效果如下图，可以选择“准备中”的房间进入，跳转到游戏页面（见 9）



7. 点击“修改角色”按钮后，重新进入选择角色页面，之后操作同 3
8. 点击“修改密码”按钮后，进入修改密码页面（如下图），输入新密码并点击“修改”按钮

9. 游戏页面

如下图所示，单击鼠标进入游戏



进入房间即进入游戏等待状态，等待状态可以在房间里任意走动，但是无法和任何物体交互。当房间的等待玩家满 3 人时系统将提示游戏开始，此时可以愉快的游戏并且和场景，其他玩家互动。

游戏操作指南：

WASD 和上下左右方向键移动人物位置，其中 AD 为左右平移，WS 为向前向后平移。

鼠标移动进行扭头方向切换，抬头低头视角控制。

鼠标点击物体可拾取物体或者和物体交互（打开密码锁或者打开密室门）。

Tab 键切出聊天框。

游戏通关攻略

1. 玩家走到书柜后面，书柜后面藏了一把钥匙，可拾取。
2. 桌面上的蜡烛道具可拾取。
3. 玩家走到书柜前，在靠左的书柜的最底下一层最靠右的书是一本日记，点击可阅读日记。
4. 日记使用了隐写术，仅有蜡烛道具的玩家可以阅读到具体的文字，其他玩家只能读到空白的信纸。
5. 日记讲述了游戏背景，透露出了暗号"compliance will be rewarded"，按 tab 切出聊天框，输入暗号，系统会返回 code=1783。
6. 走到门边上密码锁，点击密码锁弹出输入框，在里面输入正确的密码 1783，显示 opened。
7. 当密码锁成功破解后，拿到钥匙的玩家点击门，旋转门会打开，即密室逃脱成功。

团队分工

本次代码采用 github 共享（链接：github.com/Point178/Escape_WEB3D），小组主要按照入口程序 / 3D 设计 / 服务器 socket.io / 客户端 socket.io / 部署服务器 进行分工。

耿同欣 （30%）

分工：

1. 玩家登录注册功能
2. 加入房间功能

3. Socket.io 服务端
4. 找 web3D 模型
5. PBL 的管理

心得：

了解了 neo4j 图数据库的相关知识；学会使用 socket.io 进行客户端与服务器之间的通信，并实现分组广播

刘雨馨（30%）

分工：

1. 玩家登录注册功能
2. 加入房间功能
3. Socket.io 服务端

心得：

在这次高级 web 的项目中，我学习了如何使用 node.js 中的 express 框架，学习了使用 socket.io 进行服务器端与客户端的对接和交互。

陈昕婕（39%）

分工：

1. 所有 Threejs3D 部分（包括但不限于场景的布置，交互（模型调整，碰撞检测，物体拾取交互，游戏逻辑等））
2. Socket.io 客户端
3. 服务器的部署
4. Webpack 配置
5. 小组分工协调

6. 密室游戏故事线设计

心得：学习了 threejs 库，包括 threebsp 等的衍生库的使用，在实践中对 3D 模型的世界坐标，加载原理，矩阵计算，碰撞检测等的射线发射原理，透视关系等的有了具体的了解。学会 socket.io 客户端的收发使用。同时学习了 webpack 的配置，为整个项目的运行部署搭好了基础的框架，体会到了大型项目的技术栈流程。

何宇雯：（1%）

1. 查找部分模型。