**“基于Webgl和Websocket的网络3D互动平台”开发文档**

13302010044 朱天乐

1. **项目概述**

本项目使用了webgl为渲染工具，socket.io、socket.io-client作为网络通信工具，nodejs作为服务端，实现了一个多人在线3D互动平台。

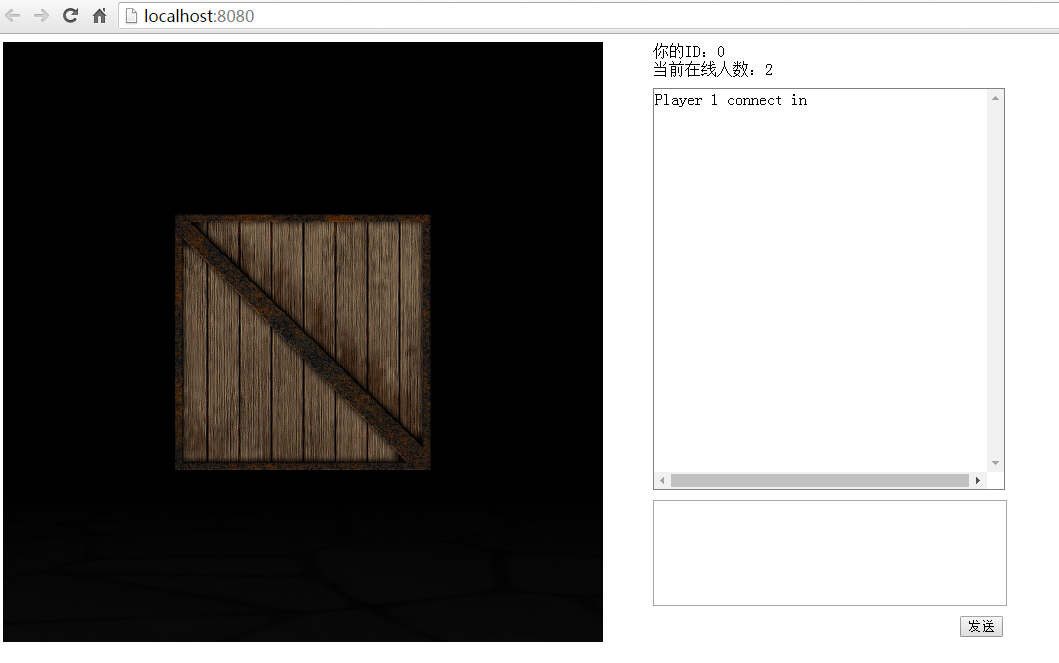
1. **使用说明**

要运行该项目，需要按照以下流程：（1）下载安装nodejs；（2）在项目目录下，在命令行输入“node app.js”；（3）使用浏览器访问“localhost:8080”。

1. **功能介绍**



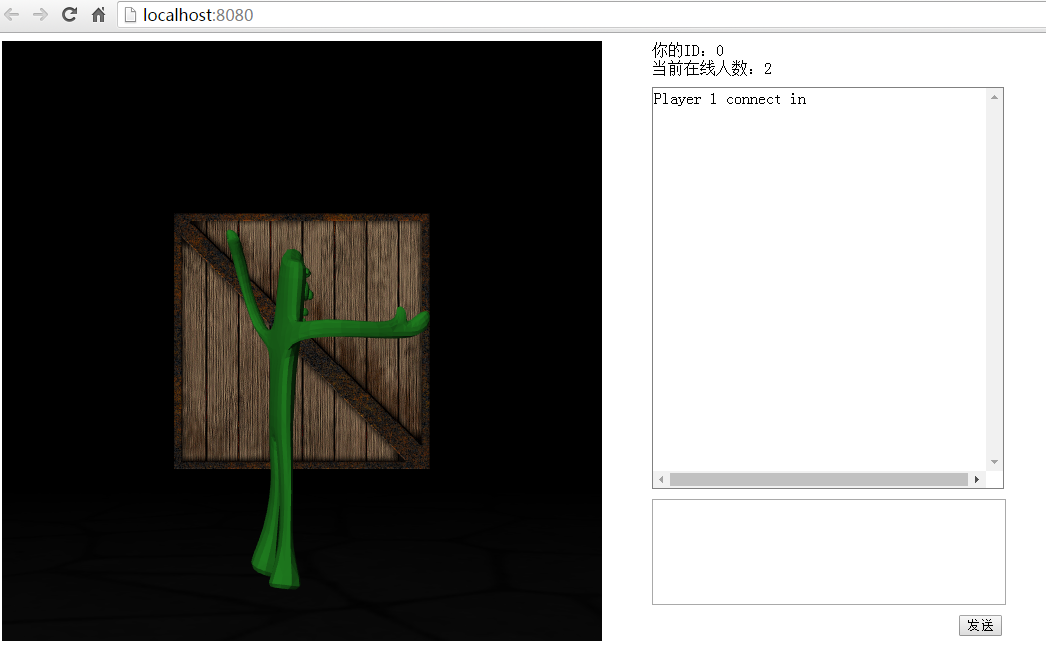
当连接到localhost:8080后，将访问上图所示页面。左边是一个使用webgl渲染的canvas，显示3D交互场景，玩家可以使用前后左右四个键进行场景漫游，WASD改变视角（复用了上学期图形学pj3的代码）；右边是聊天室，最上面显示的是服务器给该客户端设置的唯一ID，以及当前聊天室所有在线人数，下面是聊天窗口。



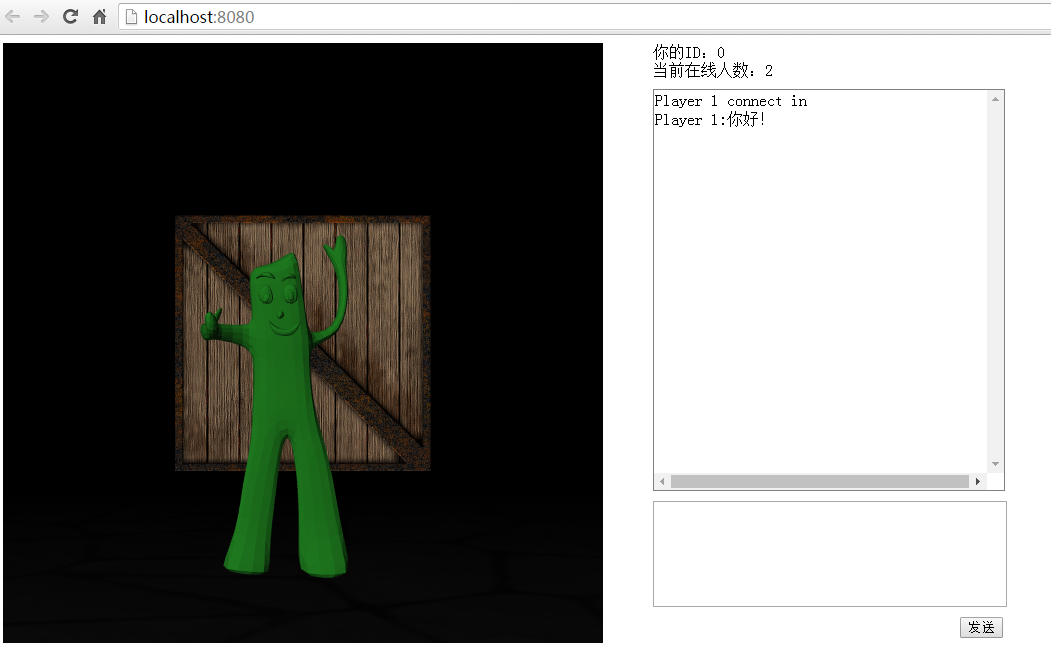
当有新玩家进入时，会在聊天窗口显示“Player N connect in”的提示信息，同时在线人数也会变化。



可以看到其他玩家移动了他的位置。



也能看到其他玩家旋转视角。



可以和其他玩家聊天。



其他玩家关闭浏览器后，可以看到聊天窗口显示了“Player N disconnected”的信息，同时该玩家的模型也从场景中消失了。

1. **客户端实现**
   1. **Webgl部分**

该部分复用了上学期图形学的project代码，在此基础上新增了三个函数：AddNewPlayer(id)、DeletePlayer(id)、OnPlayerPositionChange(id,position,lookat)。他们的用途分别是新建玩家id为id的模型，删除玩家id为id的模型，将玩家id为id的模型放置到position的位置，其朝向为lookat。

* 1. **Websocket部分**

该部分使用了socket.io-client包。在窗口打开后，连接到“localhost:8080”。连接成功后，设置了六个事件回调函数：

socket.on(‘SetPlayerNO’)：接收从服务器发来的玩家id，并将该id设置为自己的id，并显示到网页上。该函数在刚连接的时候由服务器调用；

socket.on(‘SetOtherPlayers’)：接收从服务器发来的一个int[20]数组，该数组中值为1的下标为在线的玩家id，值为0的下标为不在线的玩家id。使用AddNewPlayer(id)为所有值为1的下标新建模型添加到场景中。同时，计算出在线人数并更新。该函数在刚连接的时候由服务器调用；

socket.on('NewPlayerIn')：接收从服务器发来的玩家id，并添加该id的模型到场景中，同时添加在线人数并更新。该函数在本客户端连接后，任何新玩家连接到服务器后由服务器调用；

socket.on('OnPlayerPositionChange')：接收从服务器发来的一个JSON对象{id:int, position:{x:float, y:float, z:float}, lookat:{x:float, y:float, z:float}}，表明id玩家的模型位置移动到了position，朝向为lookat。之后调用OnPlayerPositionChange(id,position,lookat)来调整场景中模型的位置和朝向；

socket.on('PlayerOut')：接收从服务器发来的一个id，表明id玩家离线。之后调用DeletePlayer(id)将该玩家的模型从场景中删除。同时减少在线人数并更新；

socket.on(‘ReceiveMsg’)：接受从服务器发来的一个JSON对象，里面包含信息来自的玩家id以及信息内容。之后调用AddMsg把信息显示到网页上；

此外，连接成功后，还将每隔100毫秒调用OnPositionChange()函数，将一个包含id、position和lookat的JSON对象发送给服务器，表示自己的位置移动。

1. **服务端实现**

该部分使用了socket.io包。服务端维护一个长度为20的int数组‘online‘，当某个下标的值设置为1时，表示该id已经使用（在线）。

当有客户端连接时，服务器从online数组中选择一个值为0的下标，并使用socket.emit(‘SetPlayerNO’)调用客户端上的回调函数将该值设置为客户端的id。之后，使用socket.emit(‘SetOtherPlayers’)将online数组发送给客户端，让客户端初始化场景。然后，使用socket.broadcast.emit(‘NewPlayerIn’)向所有连接的客户端发广播，调用回调函数处理新玩家加入的事件。

此外，还在该连接上创建三个事件回调函数：

socket.on('onPositionChange')：接收从客户端发来的JSON对象，并广播将’OnPlayerPositionChange’事件和该JSON对象发送到所有客户端；

socket.on('disconnect')：当一个客户端断开时，广播将’PlayerOut‘事件和该玩家的id发送到所有客户端；

socket.on(‘OnSendMsg’)：接收从客户端发来的JSON对象，并广播到所有客户端。