“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

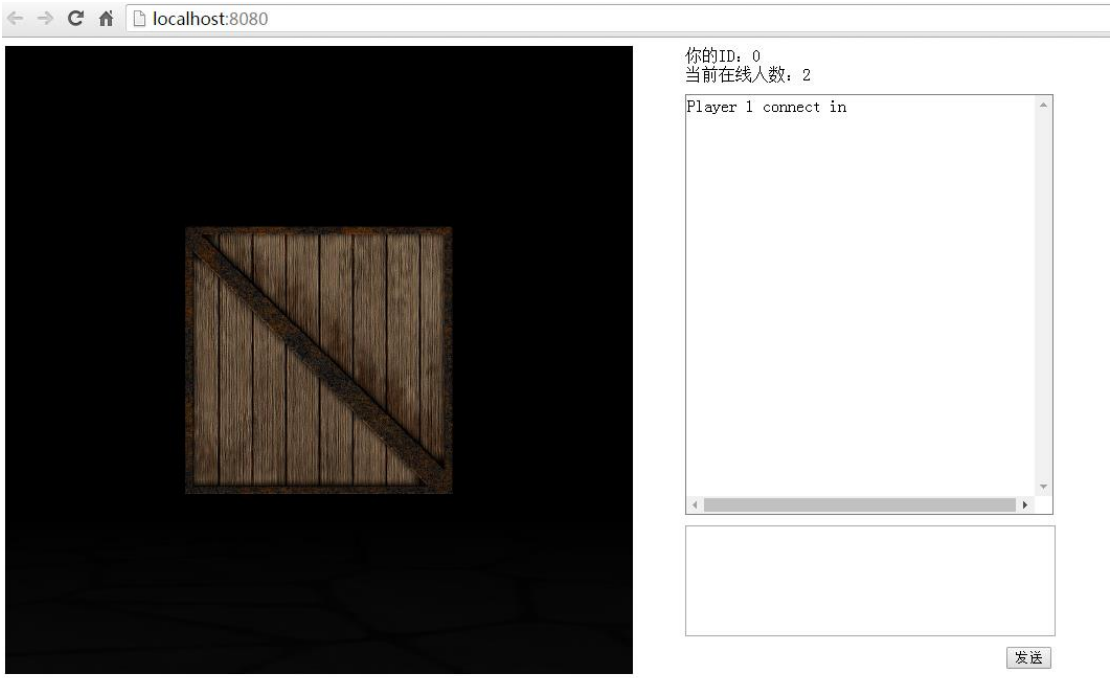
本项目使用了 webgl 为渲染工具，socket.io、socket.io-client 作为网络通信工具，nodejs 作为服务端，实现了一个多人在线 3D 互动平台。

2. 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”。

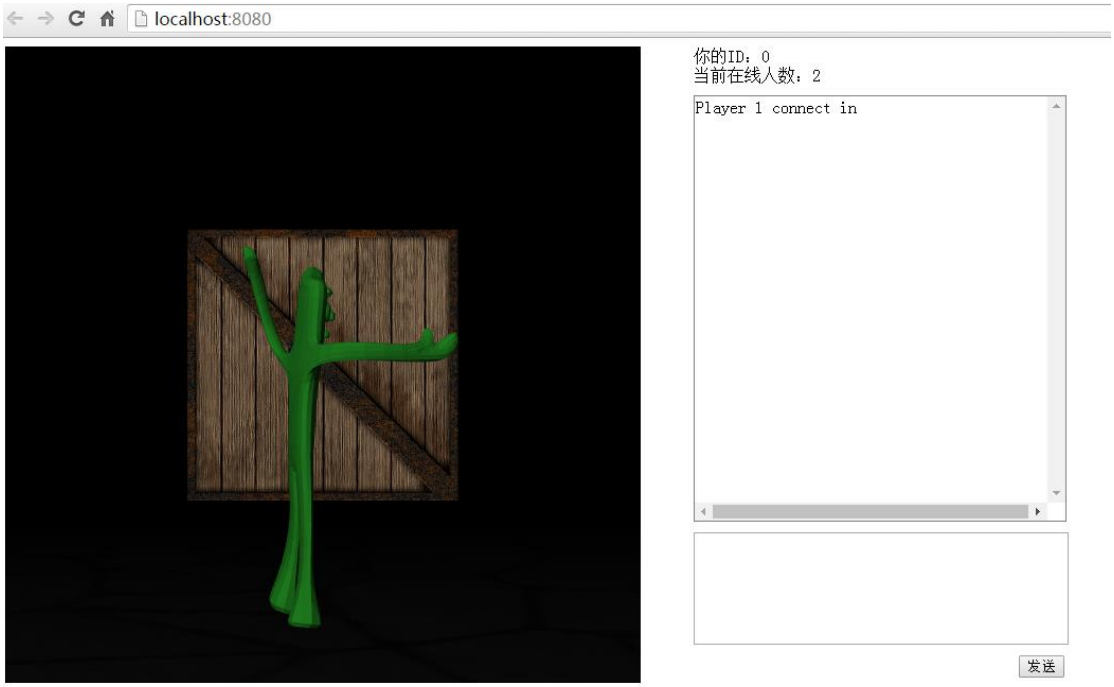
3. 功能介绍

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。

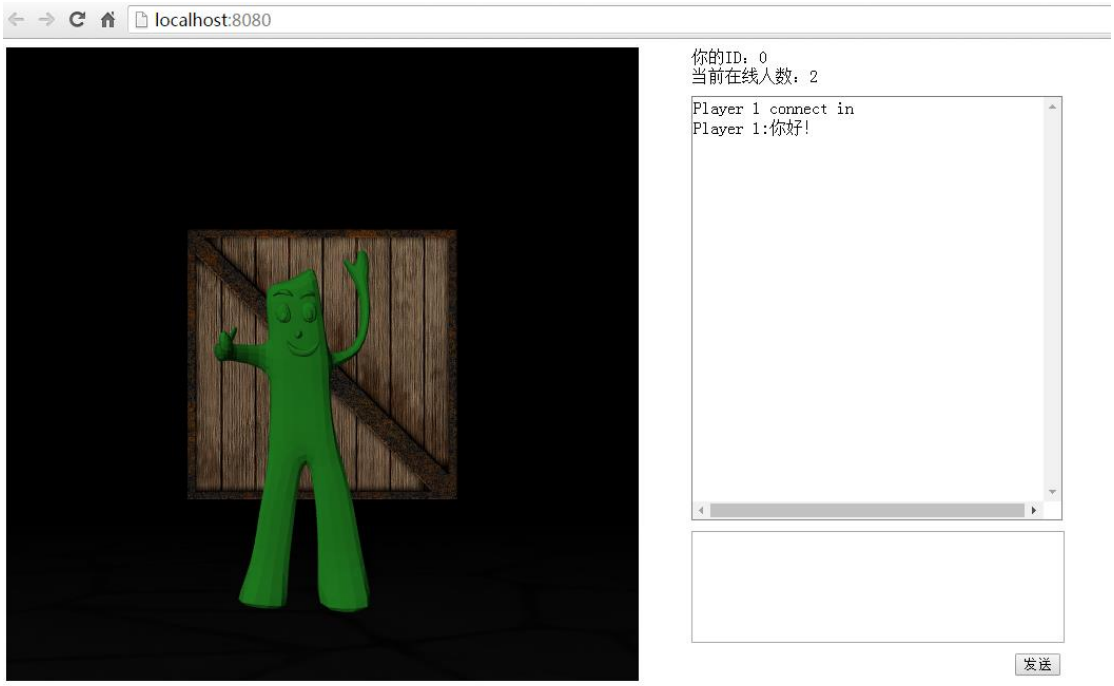


当有新玩家进入时，会在聊天窗口显示“Player N connect in”的提示信息，同时在线人数也会变化。

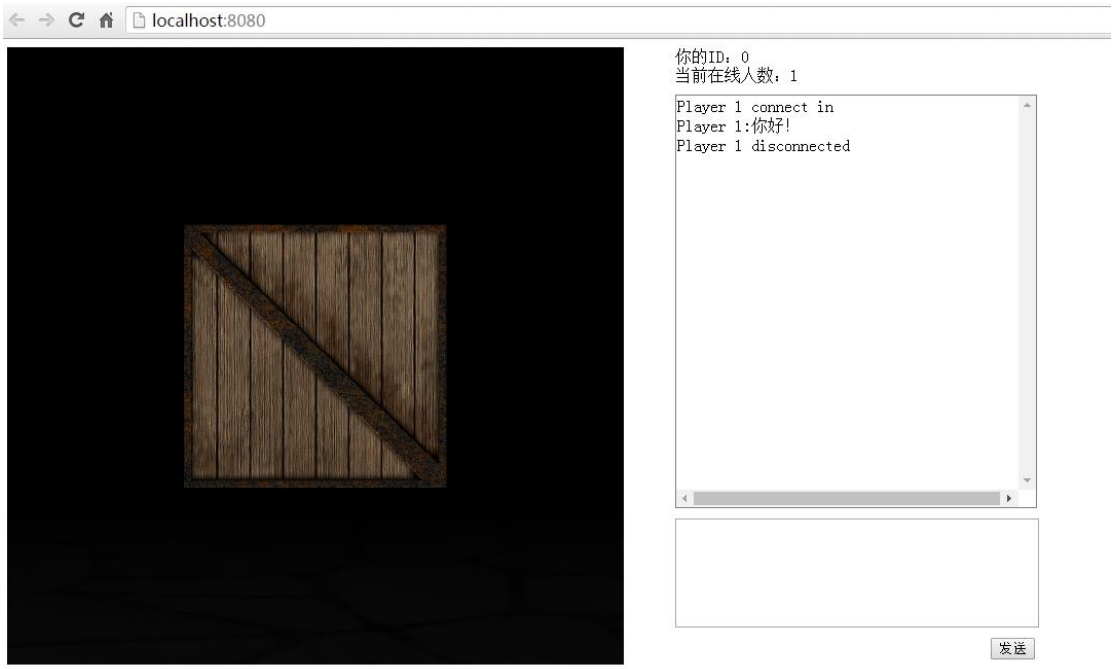
可以看到其他玩家移动了他的位置。



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。

4. 客户端实现

4.1 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。

4.2 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。

5. 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。