BlackJack 产品文档

使用说明

打开两个配置 Node.JS 环境的终端，进入 Server 目录，其中一个输入 node websocket.js，另一个输入 node http.js。然后用两个浏览器打开 127.0.0.1:360，进入游戏初始界面，在第一个页面点击 Start Game 按钮，输入第二页面右上角的 ID，在第二个页面确认，开始游戏。游戏初始发牌两张，可通过点击 Hit 按钮继续发牌，通过左侧 score 分数来决定是否继续要牌，当 score 点数超过 21 点时，游戏失败，显示”You Bust!”信息。要牌过程中随时可以通过点击 Stand 按钮结束要牌，进入判断胜负阶段。若庄家超过 21 点，则玩家胜出，显示”Maker Bust!You Win!”提示，否则按照庄家和玩家的点数大小，点数大者胜出。Restart 按钮出现，可点击重新开始游戏。

设计思想

- **主题框架**

游戏逻辑主体由 JavaScript 完成，客户端及服务器端由 Node.JS 实现。仅在设置页面大小和显示庄家暗牌两处语句使用了 jQuery。考虑到玩家和庄家两个对象，特采用了单例模式的设计思想来实现，将玩家分数等设为私有变量，从而一定程度上增强了数据的安全性。

游戏发牌采用一种完全随机算法，通过获得当前时间的毫秒数后让其分别与 4 和 13 取余而获得将发牌的花色以及点数，然后通过一个标记数组判断这张牌是否还在牌堆中，若不在则重新获取当前毫秒数生成一张牌，若在则发出该张牌并标记该牌已被发出。

- **游戏同步**

为了实现两个客户端的通信，使用了 HTML5 的 websocket 组件来实现客户端，利用 Node.JS 的 ws 模块建立 websocket 服务器作为消息中转，使用 ws 协议实现通信。为了保证客户端与服务器端保持长连接，每隔 10 秒向服务器发送一个心跳包。

游戏页面打开后，客户端首先会检查本地 cookie 是否储存有 currentID，若未储存则意味着本次连接为第一次连接或 cookie 失效，此时向服务器请求一个新的客户端标识 ID。否则获取储存的 currentID，向服务器发送使用此 ID 连接进行通讯。

当服务器接收到新 ID 请求后，将会把本次连接加入客户端队列的尾端并返回新的 ID。当服务器接收到使用既定 ID 连接的请求后，将会根据 ID 把本次连接添加到队列的指定位置。

- **游戏逻辑**

游戏初始由服务器随机生成四张牌给两个玩家各发牌两张，每名玩家会接受到自己的两张牌和对手的第一张牌。客户端根据玩家手中牌和对家明牌统计分数并显示牌面和分数。然后两方各自进行游戏。

当玩家点击 hit 按钮时，玩家向服务器请求发牌并进入等待状态，服务器进行随机发牌后将该张牌相关数据传回玩家，玩家接收到新的牌后计算是否爆牌。若爆牌则向另一个玩家发送所有牌的数据并进入等待状态，否则恢复到游戏状态。

若玩家接收到对方牌的全部数据，则代表对手已完成游戏并进入等待状态。此时对方牌的相关数据将会被储存到客户端，待玩家结束游戏后，根据之前存储的对手数据和自身数据直接显示游戏结果，同时向对手发送游戏结束命令并发送己方所有牌的相关数据。对手接收到游记结束命令后根据对方发来的数据与己方牌的信息计算并结果，此时本轮游戏结束。