扑克牌游戏项目设计说明

课程：面向对象程序设计
任课老师：陈奇
成员构成：

朱理真（组长） 3190101094 3190101094@zju.edu.cn

夏熙浩 3190100924 3190100924@zju.edu.cn

龙静毅 3190101088 3190101088@zju.edu.cn

苏哲 3180101065 3180101065@zju.edu.cn

完成时间：2021 年 7 月 11 号

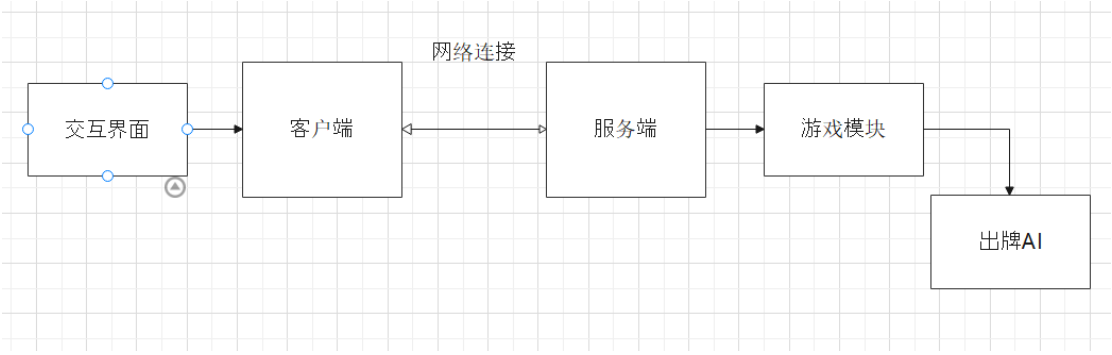
一、需求分析

目标环境：Windows10

目标任务：
使用 C++语言和面向对象知识，编写应用程序，实现扑克牌游戏，且支持：

1. 友好的图形用户界面
2. 实现三种扑克牌游戏（三人斗地主、四人斗地主和双扣）
3. 支持四玩家参与游戏（单机参与）
4. 支持 1-4 人参与游戏，不足人数由计算机扮演
5. 支持不同玩家通过网络联机参与游戏

二、总体设计



如上图，程序总体分为 5 个部分，4 个模块，其中客户端和服务端属于网络模块。用户通过

交互界面进行扑克牌游戏，同时各个客户端信息通过网络集中于游戏模块进行处理，并指示交互界面显示不同的内容，使得游戏逻辑与交互界面相当程度上独立，便于后续功能的拓展。同时，为支持低于给定人数玩家时的游戏模式，还实现了简单的扑克牌出牌 AI。

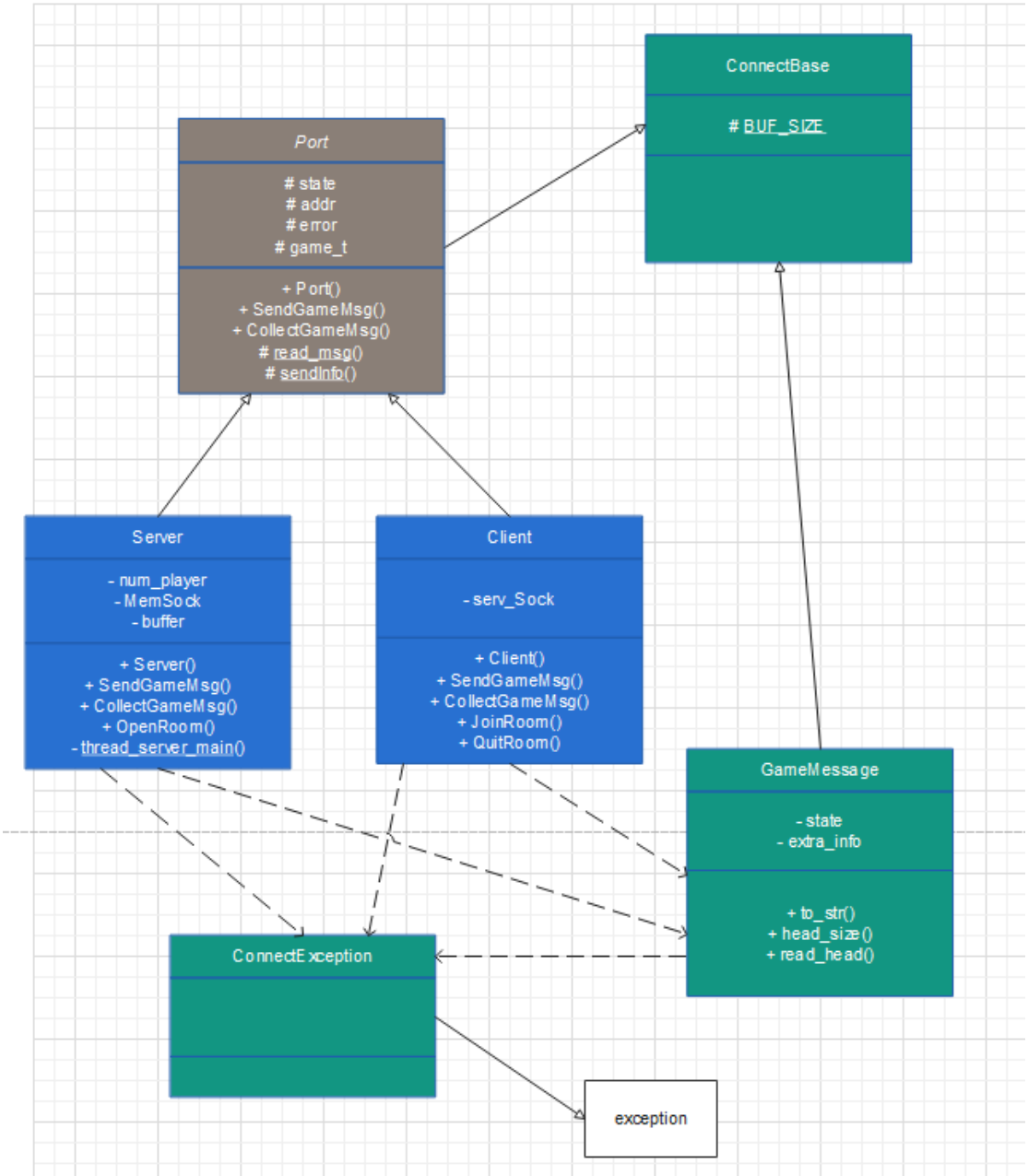
三、系统模块说明

1.网络模块

简单介绍

网络模块以 TCP/IP 协议为基础，通过调用 winsock2 库函数，利用异步 I/O 模型进行网络编程。封装实现了 Client 类和 Server 类，为其他模块提供了连接、交互基本信息和发生接收包的功能。

模块内关系



核心类介绍

ConnectBase 类：

```
static const int BUF_SIZE=1024;
```

网络模块的基本组件，提供最基础的属性。随时可以拓展，提供更多功能。

GameMessage 类：

```
int head_size() { return sizeof(state) + sizeof(int); }
int read_head(const char*);
int to_str(char*)const;
void setPackage(const Package& pkg);
Package getPackage();
static Package getPackage(char* src,int len);
const string& getName() { return extra_info; }
static string getName(char* src, int len) { return string(src, len); }
void addName(const string& name);
void setSignal(msg_t ms);
void setGameType(GameType t);
void setGameType(char* p);
void setInt(int integer);
void setInt(char* p);
```

最基础的网络包。包含发送接收所必要的长度、状态等信息，支持 Package 类、字符串、整数和网络状态信号的发送。可以提供更多拓展信息类型的发送。

ConnectException 类：

继承于 exception 类，用于网络部分的异常抛出。被多种异常类继承，表示不同的异常原因。

Port 类

```
msg_t read_msg(SOCKET s, GameMessage& gm, Package& pkg);

static void SendSignal(msg_t sig, SOCKET to);
static void SendName(const string& name, SOCKET to);
static void SendGameType(GameType gt, SOCKET to);
static void SendInt(int integer, SOCKET to);
static void read_buf(SOCKET s, char* dst, int size);
static void SendInfo(const GameMessage& gm, SOCKET clntsocket);
```

通信端口的抽象类。提供基础的设定套接字参数、封装并发送网络包和接收并解析网络包功能。

Server 类

```
void SendGameMsg(const Package &p);
Package CollectGameMsg(int sender=i_server);
const string& GetClientNameR(int k) const;
```

```
int OpenRoom(GameType gt,int humans,int robots);
```

作为服务端，继承于 Port 类，可以进行游戏房间的开放（即接收客户端）和网络包的收发。为不阻塞程序，使用异步 I/O 来处理来自多个客户端的信息，同时将其分流，各自缓存。其他模块通过 Server 接收信息时，实际是从缓存中取得的。

Client 类

```
Client();
const vector<string>& JoinRoom(
    const char* IP,
    const string& name,
    GameType& gameType,
    int&pos); // 接收或超时后才返回
int QuitRoom();

void SendGameMsg(const Package &p);
Package CollectGameMsg(int sender=i_server);
```

作为客户端，继承于 Port 类，提供加入房间，退出房间和网络包的收发功能。

2.游戏模块

简单介绍

为扑克牌游戏提供了游戏逻辑和游戏消息的发送和接收。实现了三种游戏，分别是三人斗地主、四人斗地主和双扣。

核心类介绍

Card 类

最基本的扑克游戏卡片，具有花色和阶数两种属性（大小王也包括在内）

CardSet 类

卡牌的集合，提供插入、删除、随机抽取和提供某种卡片的数量的功能。

```
CardSet(int n = 0); // returns a set of n full decks, containing 54n cards
CardSet(const string &s); // decodes a string encoded by CardSet::String()
int GetNum(int s, int r) const; // returns the number of Card(s, r)'s in the set
int GetNumOfCards() const; // returns the total number of cards in the set
Card GetCard(int k) const; // returns the kth card in the set sorted with ID
void Insert(const Card &c); // inserts a c into the set
CardSet Take(int k = 1); // randomly takes and subtracts k cards from the set
string String() const; // encodes the set to a string
operator bool() const; // returns whether the set is empty
CardSet operator+(const Card &s) const; // returns the sum of the set and c
CardSet operator-(const Card &c) const; // returns the difference of the set and c
bool operator==(const CardSet &s) const; // returns whether the set is exactly the
same as s
bool operator!=(const CardSet &s) const; // returns whether the set is different from
```

s

ci 类

用于记录轮到哪一个玩家

```
ci(int x = 0);
operator int() const;
ci<m> operator++();
ci<m> operator--(int);
ci<m> operator+=(int x);
ci<m> operator-=(int x);
ci<m> operator+(int x) const;
ci<m> operator-(int x) const;
```

Message 和及其聚合 MsgSeries 类

```
Message(MsgType t = m_empty, bool b = false); // returns a message with unspecified parameters
Message(const string &s); // decodes s to a message
MsgType GetType() const; // returns the type
bool IsRequest() const; // returns whether the message requires a reply
int GetPar(int k = 0) const; // returns the kth parameter, with k in [0, 8)
int GetPlayer() const; // returns the 7th parameter, which is usually a player index
int GetTime() const; // returns the 6th parameter, which is usually a time
CardSet GetCards() const; // returns the card set parameter
string GetExtension() const; // returns the extension, usually storing variable-length parameters e.g. text to display
const string &GetExtensionR() const; // returns the extension by reference
void SetType(MsgType t); // sets the type to t
void SetIsRequest(bool b);
void SetPar(int k, int v); // sets the kth parameter to val
void SetPlayer(int p);
void SetTime(int t);
void SetCards(const CardSet &s);
void SetExtension(const string &s);
string String() const; // encodes the message to a string
```

负责发送游戏信息

Package 类

```
Package(const Header &h, const string &d); // returns a package with h as header and d as payload data

Header GetHeader() const; // returns the header
string GetData() const; // returns the payload data
const string &GetDataR() const; // returns the payload data by reference
```

最基础的游戏发送包

Analysis 类

```
Analysis(int f = 0, int p = 0, int k = 0, int l = 0, int v = 0);
Analysis(const CardSet &s, Rule r); // analyzes s under r
```

提供出牌的模式分析功能

3.交互界面模块

交互界面模块主要继承于 wxWidgets 库中的类

MyButton 类:

实现按钮

MainMenu 类:

实现菜单

PageController 类:

实现页面控制

MainFrame 类:

主框架结构

wxImagePanel 类:

用于显示图片（扑克牌）

DeckPanel 类:

扑克牌基座

CenterBlockSizer 类:

实现中心控制

4.AI 模块

作为 game 的成员函数存在，实现以下几个函数:

```
int get_score(const CardSet& cards) { // bid 1 if score > 58 bid 1; bid 2 if score > 62 ; bid 3 if score >68 // pass if score < 53…

#define show_card(card_set) for (int i = 0; i < card_set.GetNumOfCards(); i++) {cerr << "(" << card_set.GetCard(i).GetSuit() << ", " <

// n 当前已经选择了多少张牌
// last_n 上一轮出了多少张牌
// first 现在从第几张牌开始选
void SearchBest(const Rule& rule, const CardSet& my_card, const Analysis& last, CardSet& to_dealt, int n, int first, int &max_score) {

#include <assert.h>

CardSet DouDizhuGame::playout_robot(ci<3> k) const {…

CardSet SirenDouDizhuGame::playout_robot(ci<4> k) const {…

CardSet ShuangkouGame::playout_robot(ci<4> k) const {…

int DouDizhuGame::bid_robot(ci<3> k) const {…

int SirenDouDizhuGame::bid_robot(ci<4> k) const {…
```

5.扑克牌游戏程序的界面

见附录【程序使用说明】

6.扑克牌游戏程序的存储模块

无存储模块

四、系统设计难点及解决

1.网络模块

难点：由于随时要接收消息，同时保持与其他模块的独立性，需要实现多线程，以异步的方式接收信息。

解决：使用 Windows 自带的进程函数，实现了一些数据的加锁和解锁；同时使用异步 I/O 模型进行操作。

难点：由于需要进行多局游戏，玩家需退出房间再重进，但是房主房间只会在所有玩家都退出时才会关闭，导致房间完全关闭前又有玩家进入，从而产生错误。

解决：玩家若再次进入房间，则先进入等待状态；等前一局的所有玩家都退出后，再将它们加入到房间成员中，并开始游戏。

难点：由于采用了多线程结构，传统的调试方法使用起来会有困难，从而降低调试效率

解决：定义了具有一定层次结构的异常类，并且捕获时能够输出信息。在所有可能出错的代码段设置不同的异常，使得异常抛出时，很容易确定发生地点，从而掌握发生原因，提高了调试效率。

2.游戏模块

难点：需要实现多种游戏类型，代码可能有大量的复用同时又有不同的结构，需要精心地设计继承关系

解决：使用模板类结构，使多种游戏类继承共同的游戏逻辑框架（比如扑克牌的数量，大小等），对不同的游戏规则实现不同的玩法处理。

难点：在处理游戏逻辑的同时，还要进行游戏控制信息的发送，以指示 UI 显示内容，这一步不可避免的导致封装性的减弱。

解决：需要通过双方积极有效的沟通交流，确定信号含义和发送接收顺序，使得游戏能够顺利进行。

3.交互界面模块

这次大程设计我使用的图形库是 wxWidgets，主要是考虑到 QT 我不太了解，听说它是通过嵌入式修改 C++ 语法来实现一些附加的功能的，我不太喜欢研究和课程无关的东西，所以就选择了 wxWidgets，后者是纯粹的 C++ 的库。当然，二者的学习成本最后看下来其实也差不多，毕竟这是我第一次使用 C++ 的第三方库，对很多套路还不是很熟悉，使用 C 的库的时候我们一般只是在调用它给出的函数或者宏，而 C++ 主要则是依靠继承库给出的类来使用库，二者的思维方式还是挺不一样的。

代码复用：

由于 GUI 不太能够代码复用，毕竟每个界面都长得不太一样，所以我只提取了页面布局作为代码复用的重点。除了主菜单之外的所有界面都使用我定义的 CenterBlockSizer 进行布局，使得总体看上去一致，修改起来也比较简单。

多线程和线程安全

多线程带来了多个线程同时修改同一个变量的情况。最开始我对这个问题的处理方式是使用 C++ 11 提供的 atomic 类型，但发现后者只能针对拥有“trivial constructor”的类型（我觉得它的意思就是 built-in type），对于 string 这样比较复杂的变量没法处理。同时我并不只是

需要保证某个变量的操作的原子性，实际上我要保证的是整个游戏状态的原子性。因此，最终我使用了 C++ 提供的 mutex。这里又遇到了一些问题，由于之前在数据库的课程上我学过锁相关的知识，所以我一直以为锁是“加在某个数据”上的，在这反而被误导了，其实只要我们在操作前上锁，操作后解锁，这个锁可以锁住任何东西（当然，本质上是锁住数据，但锁住什么数据不再是锁-数据一一对应了，而是更加灵活了）。

架构和规划的重要性

在这里我可能得倒倒苦水了。在五月份的时候，我们曾经开过一次线上的会议，会议结束我起草了一份游戏模块的头文件，希望大家进行修改讨论，结果无人响应。到了期末考试前一周写游戏模块的同学才告诉我他已经按照他自己的想法设计、实现好了。没有办法，我只好重构了不少代码来缝合他的代码。我承认他设设计地很完善，以至于后面我们也没怎么改动，但这样的合作显然是不愉快的。到最后我甚至发现同样的一个东西，合作的三个同学竟然用了三套完全不一样的方式去表示，信息的交流非常不顺畅。

4.AI 模块

难点：既要保证出牌的合法性，又要尽可能地保证规则的多样性，对不同游戏的出牌有不同的应对方式。同时在首先出牌的情况下，要根据自身的手牌和场上的情况，灵活地出牌。考虑到出牌的复杂度，枚举所有情况是不可能的。

解决：跟牌时，采用贪心的策略，尽量找到比同类型的牌面较大的牌；如果没有，再考虑是否有炸弹或王炸。首次发牌时，则从众多出牌选择中随机选择。

五、总结

此次面向对象程序设计的实践，为组员们提供了切实而印象深刻的面向对象编程体验，大大丰富了其编程经验。就面向对象程序设计的小组合作编程而言，我们总结了很多相关的经验：

1.团队要保持积极的沟通

团队合作编程要成功，这一点是首要的。有很多地方需要沟通：总体框架的搭建和沟通；由于需求的发现，后期某些接口的参数需要改变，这需要两者的妥协；接口的具体使用规则与方法；模块在单元测试下没有注意或发现的问题在联合测试时被发现，需要进行问题的及时反馈；在进度上要保持透明，防止任务拖到后期无法进行抢救；

而要保持积极的沟通，应该选择适宜的线上平台，同时定期线下交流。沟通本身并不是没有成本，但是不沟通可能会有更多的损失。

2.做好完备的规划

良好的分工，需要良好的规划。规划的工作最好由一个人来，雏形定下以后，结合大家的意见进行细节的修改，得到一个以抽象类（接口）为基本元素的类关系图。此次设计中，对接口并没有抽象化，限制了其可拓展性。架构的规划，使得程序员不需要花太多的精力进行结构的设计，而是专注于实现代码逻辑，从而提高工作效率。为了防止对结构的大幅度修改，在结构设计之初就要有完备的构想，减少后续工作中由于逻辑缺陷所带来的修改成本。同时尽量遵循面向对象设计规则，尽量多用组合，减少继承。在此次的设计过程中，我们就由于初期的设计不当，导致中期接口的大幅度修改，增加了很大的修改成本。

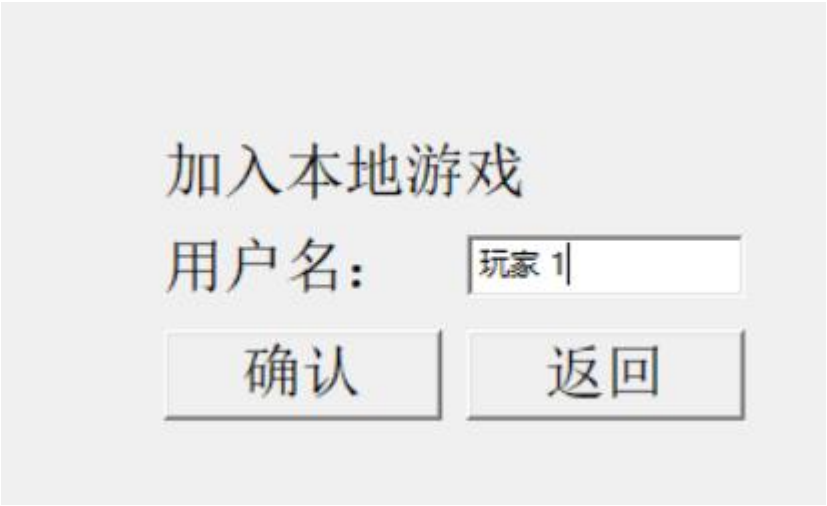
3.做好单元测试

模块在被拿去使用前，应做好完备的单元测试，考虑所有情况（保证在多样且大规模的输入中保持正常），减小联合测试时出错的可能性，因为联合测试时出错的原因复杂，难以调试。以网络模块为例，要实现一个简单的接发装置，在本地或其他计算机上进行通讯调试，确认成功连接-发送-接收-关闭等功能正常无误后才能够提交。

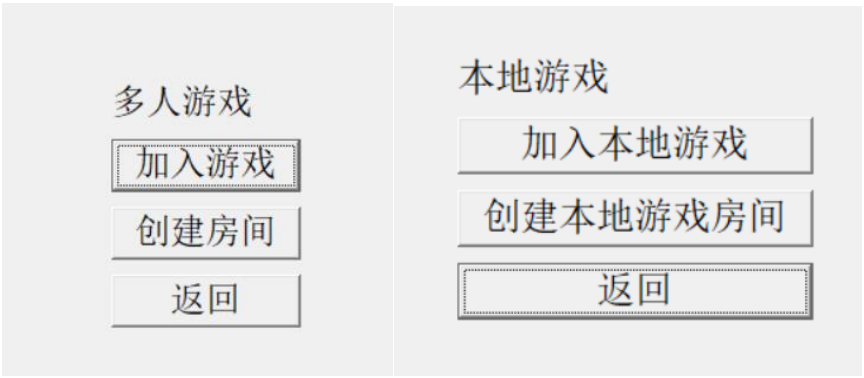
附录

程序使用说明

游戏界面如图，可选择单人或多人游戏。
若选择单人游戏，则在本地打开多个应用程序进行自机上的扑克牌游戏操作，无需输入 IP；

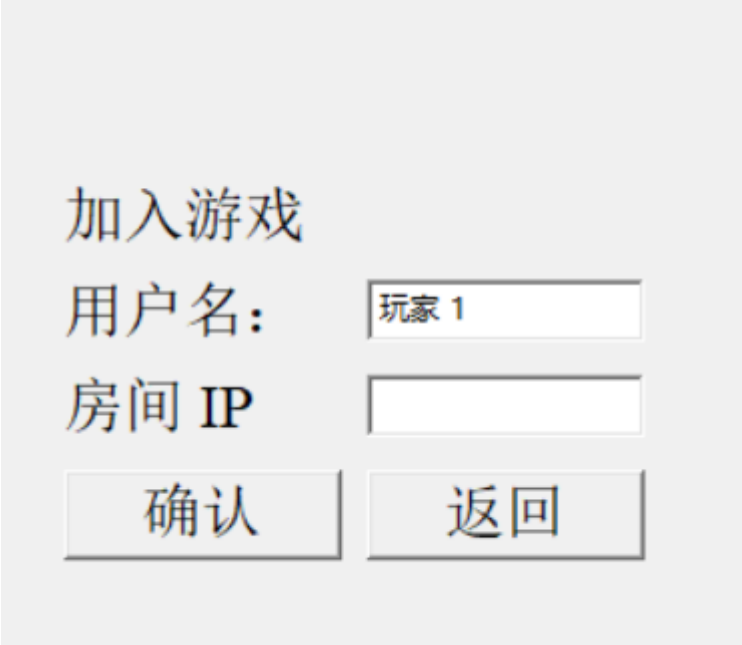


若选择多人游戏，则多台电脑可以联机互通。
无论多人还是本地游戏，都需要选择是自己创建房间还是加入已有的房间，类似于下图

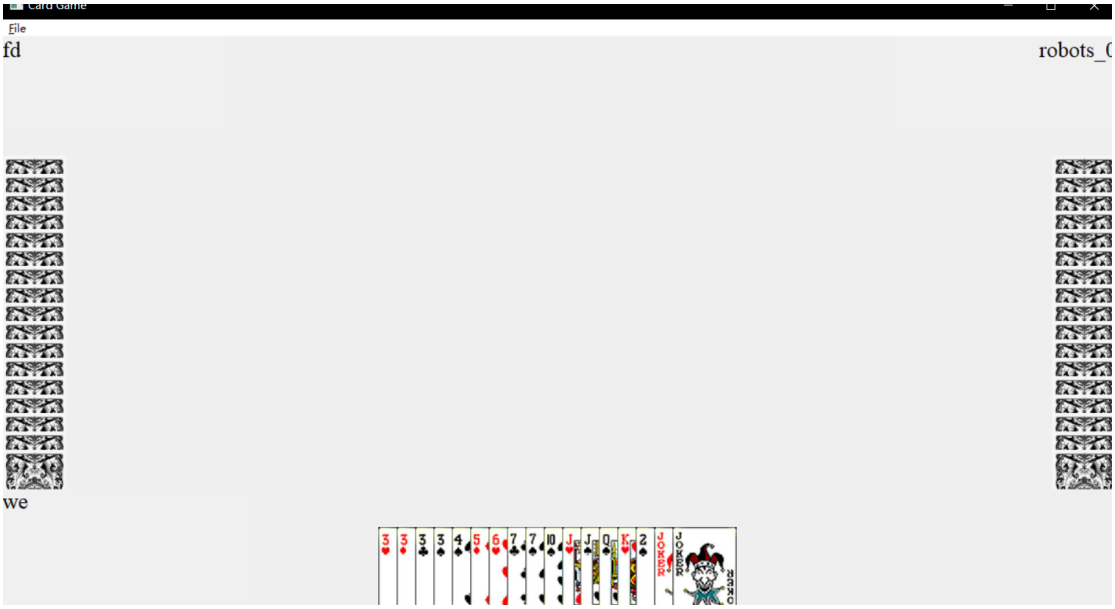


如果选择创建房间，则需要指定玩家人数和游戏类型。若玩家数小于该类型游戏所必要的人数，剩下的角色则由机器人扮演。

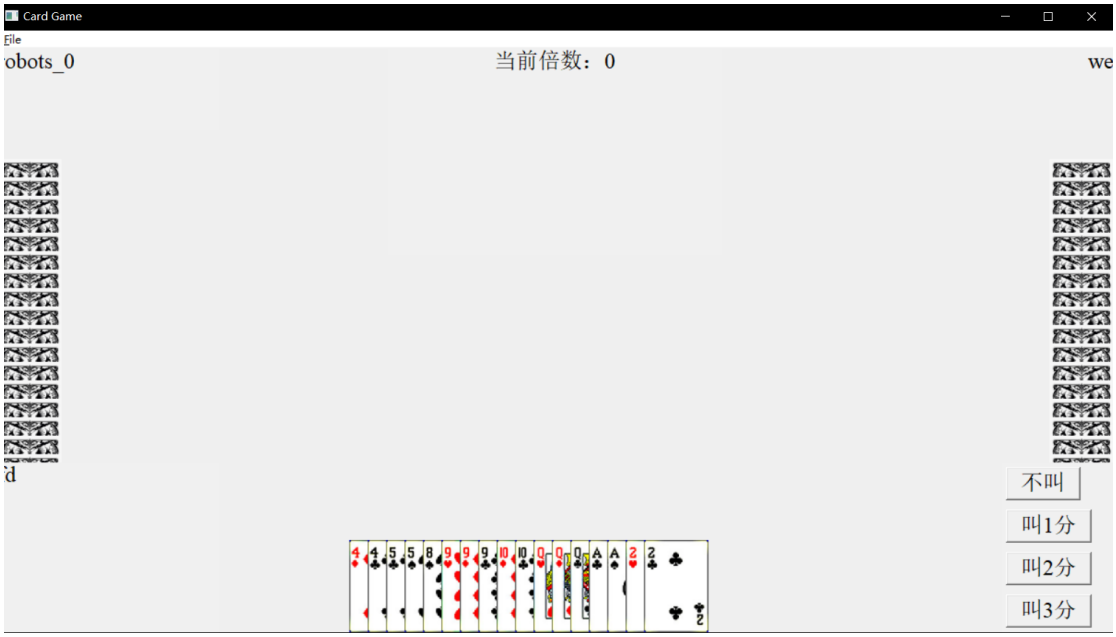
而在多人游戏下选择加入房间时，只需要输入用户名和房主计算机的 IP 地址，即可进行联机。（注意，要让房主先打开房间再加入，不然可能导致加入失败）



游戏界面如图



根据右边的按钮，
选择叫分、出牌等功能



选中要出的牌，即可按下出牌；否则可以选择跳过

如果超时，直接设为 pass

最后一方将牌打完后，即获得胜利

系统开发日志

5 月 5 日
准备开始选题

5 月 6 日
确定选题为扑克牌

5 月 16 日
第一次线上讨论，规划设计大纲，同时进行分工，初步设定游戏类型为扑克牌和 UNO

6 月 1 日
完成中期报告

6 月 2 日
初步的代码提交，更改游戏类型为斗地主和双扣

6 月 20 日
框架临时改动，组员们线下讨论修改

6 月 22 日
游戏逻辑模块初步代码提交，并进行讨论修改

6 月 23 日
整合代码，通过编译

6 月 24 日
增加 m_think 信号
网络模块通过测试

7 月 10 日
除 AI 模块，扑克牌的 UI、网络和游戏模块均已完备，且通过测试

7 月 11 日
补充 AI 部分，完备程序并提交
完成报告

小组分工
网络模块：朱理真
交互界面：龙静毅
游戏逻辑：夏熙浩
游戏 AI：龙静毅、朱理真（接替苏哲）