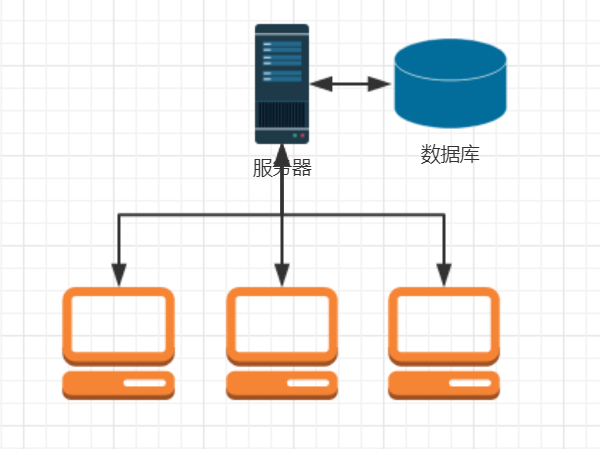
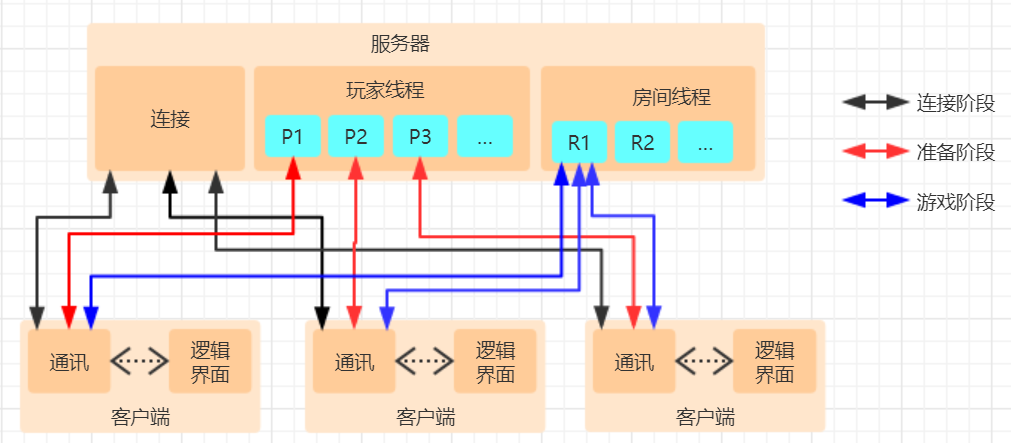
设计文档

整体框架

客户端与服务器通过socket，服务器使用数据库使数据持久化，保存和使用用户账户、密码等信息。



通讯框架



数据通讯按照程序运行的阶段可分为三个阶段，划分三个阶段的依据主要为服务器的线程状态：

* 连接阶段：多个客户端对应一个服务器线程
* 准备阶段：一对一
* 游戏阶段：三对一

准备阶段，服务器在主线程监听某端口，收到客户端连接请求时，服务器将为每一个客户端创建一个线程（玩家线程），并携带socket用于通讯。

准备阶段，服务器在玩家线程与客户端通讯。客户端通过登陆/注册界面提交信息，并等待反馈信息。服务器接收来自客户端的注册/登陆信息，从数据库修改或查询用户信息并回应对应的信息。客户端通过创建/加入房间界面提交信息，并等待服务器反馈。服务器收到创建房间信息后，会开启房间线程，对应客户端的玩家线程将结束，等待全部（三个）玩家加入；服务器收到加入房间信息，查询房间列表，将对应玩家的socket加入房间进程，应客户端的玩家线程将结束，否则返回错误信息。玩家全部加入后，房间线程拥有三个客户端对应的socket可以进行通讯。

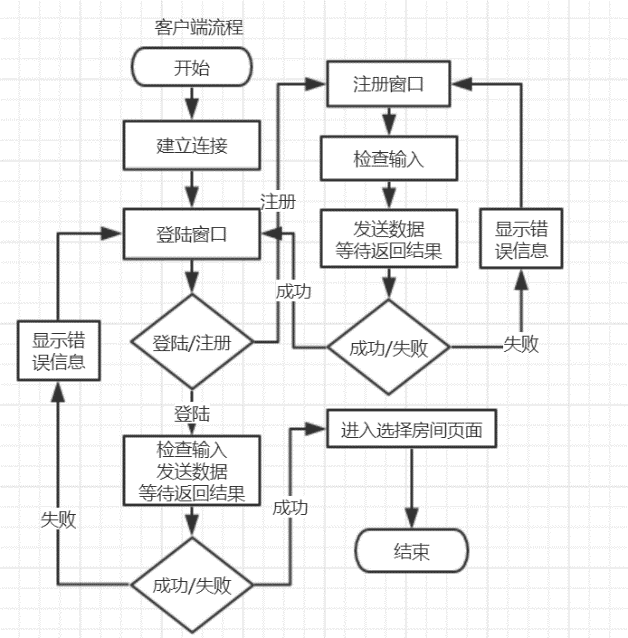
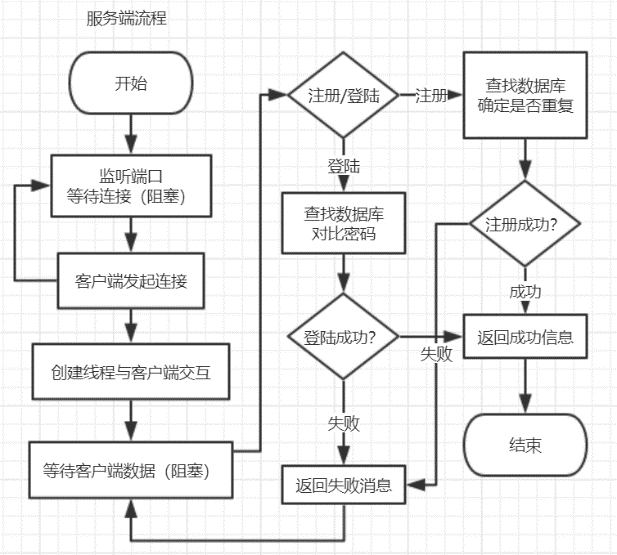
游戏阶段，客户端通过界面按钮提交信息，服务器在服务线程接收来自客户端的信息，简单处理后转发给另外两个客户端，服务器监听下一个socket。另外两个客户端收到信息后，按相应规则呈现在界面。

程序流程

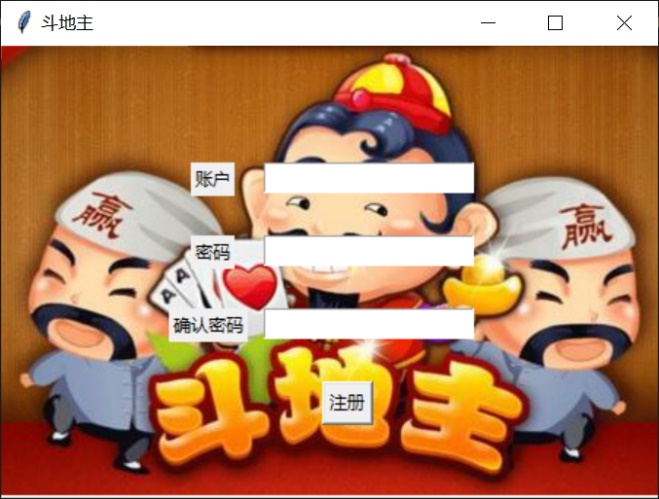
按照程序的流程，可将程序分为以下三个子流程：

* 注册/登陆子流程
* 创建/加入房间子流程
* 游戏子流程

注册/登陆子流程



客户端

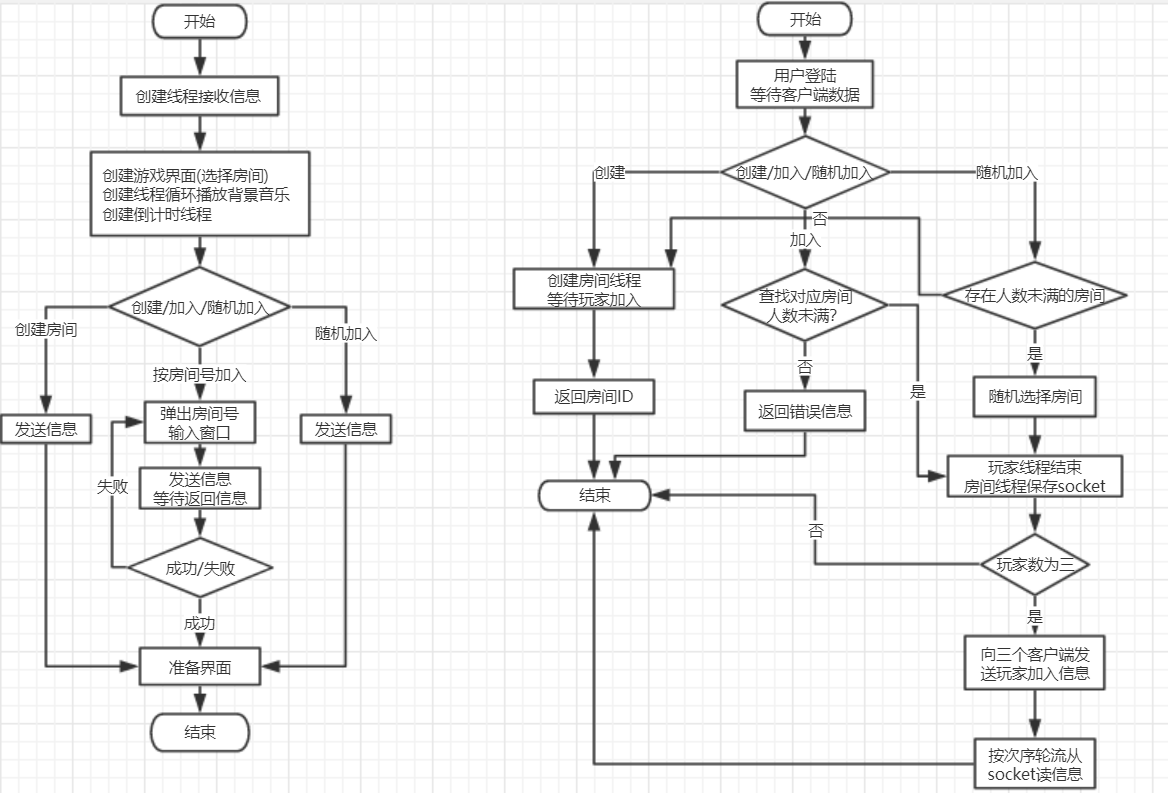


使用Tkinter创建界面

服务器

使用SQLite数据库，存储用户“用户名”、“密码”字段。

创建/加入房间子流程



客户端

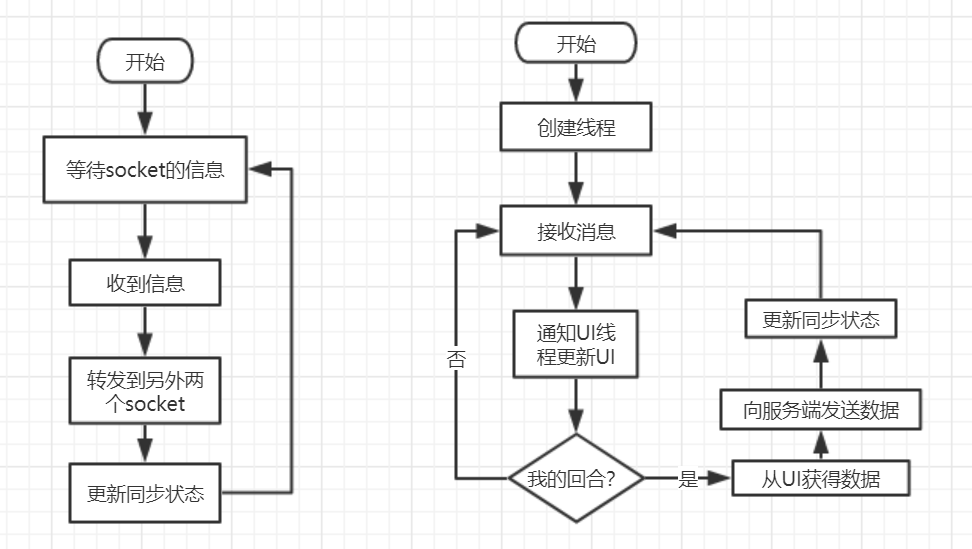


使用pygame绘制界面

创建线程循环播放背景音乐

创建线程接收服务端发送信息

游戏阶段



服务器和客户端分别维护一个同步数据，以便指示当前是哪个玩家的回合。

游戏阶段客户端工作：

更新其他玩家的信息，剩余牌数，上次出牌。

指示当前回合，用时钟图标指示当前哪个玩家在操作。

倒计时，规定时间内没有出牌，则跳过或由程序出牌（必须要出牌的情况）。

提示功能，按照一定要求给出一组可以出的牌。



这个图时钟位置不对，

算法一：牌型分析



构造词典存储所有合法出牌方式，并按一定方式组织：

* 牌按牌面大小顺序排列，以"3", "4", "5", "6", "7", "8", "9", "0", "J", "Q", "K", "A", "2", "w", "W"顺序标识。
* 字典的键为牌型，数据类型为字符串。
* 字典的值为合法的出牌方式集合，类型为数组，出牌方式数据类型为字符串。
* 出牌方式数组按牌面大小排列，便于比较大小。

用1-54标识不同花色的54张牌，当接收到若干牌时，忽略花色，用"3", "4", "5", "6", "7", "8", "9", "0", "J", "Q", "K", "A", "2", "w", "W"标识牌，按牌面大小排序并生成字符串，然后在字典中搜索相应的字符串。若找不到，则牌型不合法，若找到，则可以获得牌型的标识（字典中对应的键）以及牌面大小（数组中的下标）。

算法二：提示算法

作用：根据上家的出牌以及自己手上的牌，选择提供备选的出牌方式

按如下

1. 如果是空，返回第一张（牌面最小）
2. 如果是王炸，返回空列表
3. 找与上家相同牌型下牌面更大的牌组合，若有，则返回
4. 找是否有炸弹或王炸，如有，则返回
5. 返回空列表

如何判断手上的牌是否包含上家牌型（例如，“33445566”中是否包含“3456”）

对牌型的按牌面计数

“33445566”🡪{‘3’:2, ‘4’:2, ‘5’:2, ‘6’:2}

“3456”🡪{‘3’:1, ‘4’:1, ‘5’:1, ‘6’:1}

比较上家牌面的数量是否都小于自己牌面的数量。

由于“33445566”中包含的牌型的数量均大于“3456”中，所以包含。