# 21点（树形牌面）

# 需求分析与概要设计

## 项目说明

## 项目目标：

实现一个可视化界面的树形牌面21点扑克牌游戏。

## 软硬件环境需求

介绍软件的运行环境，包括操作系统、数据库、网络等方面等要求。

操作系统：windows

数据库：mysql

不知能否实现 网络：和在线玩家玩，金币排名

## 使用的关键技术：

介绍软件使用到的关键技术，包括技术难点和亮点。

难点：连接网络，实现在线模式

亮点：数据库

图形用户界面

## 需求分析

## 系统用例

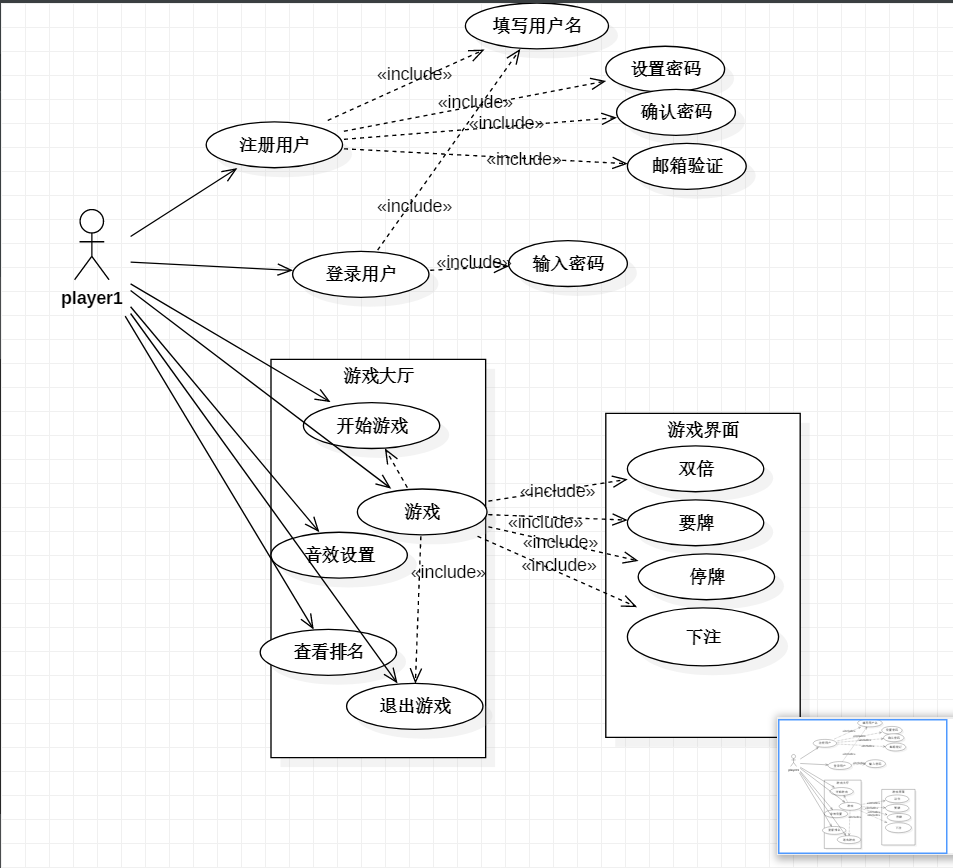


图 1 系统用例图

（用例图使用StarUML绘制，要画出参与者和对应的用例）

(对用例图中的各个用例进行详细说明，从使用者的角度说明如何使用系统，不用描述软件系统内部如何实现该功能)

**(1)注册用户**

**参与者： 玩家1**

**基本事件流：学生进入首页，点击注册用户，进入注册用户页面，填写用户名，设置密码确认密码，输入邮箱验证，拥有账号密码。**

(**2)登录用户**

**参与者：玩家1**

**基本事件流：在首页用户名输入框输入用户名，密码登录账号**

**(3)开始游戏**

**参与者：玩家1**

**基本事件流：在主菜单页面选择开始游戏，进入游戏**

**(4)音效设置**

**参与者：玩家1**

**基本事件流：在界面点击音效按钮选择开关音效**

**(5)查看排名**

**参与者：玩家1**

**基本事件流：玩家选择查看排名，显示玩家所在排名页面**

**(6)退出游戏**

**参与者：玩家1**

**基本事件流：在界面选择退出游戏，退出**

**(7)双倍**

**参与者：玩家1**

**基本事件流：玩家在游戏界面选择双倍，下注变成双倍**

**(8)要牌**

**参与者：玩家1**

**基本事件流：玩家在游戏界面选择要牌，获得一张牌**

**(9)停牌**

**参与者：玩家1**

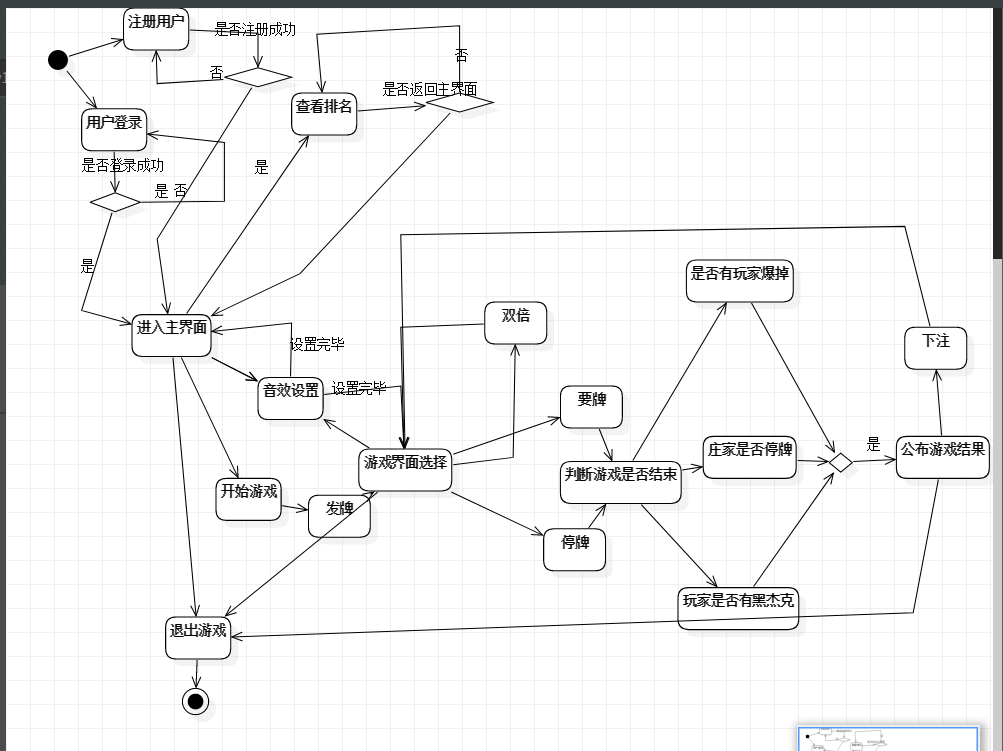
**基本事件流：玩家在游戏界面选择停牌，在此轮中不要牌**

**(10)下注**

**参与者：玩家1**

**基本事件流：一轮游戏结束后，玩家在游戏界面选择下注，开始新一轮游戏**

## 业务流程

(使用UML活动图（使用StarUML）画出系统的使用和操作过程。不用画出所有的使用场景，画出一两个关键用例的使用过程即可) 

## 概要设计

## 功能模块设计

21点游戏

图 3功能模块图

1. **用户登录**

**输入：**玩家设置的用户名，密码

**输出：**

**功能概述：** 用户登录，跳转至游戏主页面

1. **用户注册**

**输入：**用户名设置，密码

**输出：**

**功能概述：**用户点击注册按钮后，进入注册界面，设置用户名，输入密码，邮箱，点击验证，验证成功进入主界面

1. **查看排名**

**输入：**

**输出：**

**功能概述：**点击查看排名按钮，弹出排名界面

1. **游戏流程设计**

**输入：**

**输出：**

**功能概述：**玩家点击玩游戏后，进入游戏界面，系统发牌，判断玩家和庄家的点数，是否符合游戏结束条件，符合，公布结果，不符合，玩家继续游戏，选择“要牌”，“停牌”，直至符合游戏结束条件，公布结果，公布玩家赌注得失。

1. **游戏赔率设置**

**输入：**

**输出：**

**功能概述：**

游戏开始时，如果庄家亮牌为“A”，玩家选择是否买保险，如果玩家选择买，庄家为黑杰克，庄家获得玩家的保险金，玩家收回赌注，庄家不是黑杰克，玩家保险金归庄家所有，游戏继续

玩家为黑杰克，玩家获得自己赌注的1.5倍。

后续游戏，玩家和庄家点数相同，庄家赢，获得玩家所有赌注，玩家点数比庄家高，获得自己的赌注，庄家点数比玩家高，庄家获得玩家的赌注，玩家爆牌，庄家获得玩家的赌注，庄家爆牌，玩家获得自己的赌注。

1. **游戏结果公布**

**输入：**

**输出：**

**功能概述：**公布所有玩家获得的赌注。

1. **用户名**

**说明：**数据库属性，存储玩家的用户名

1. **账号**

**说明：**数据库属性，存储玩家的邮箱账号

1. **密码**

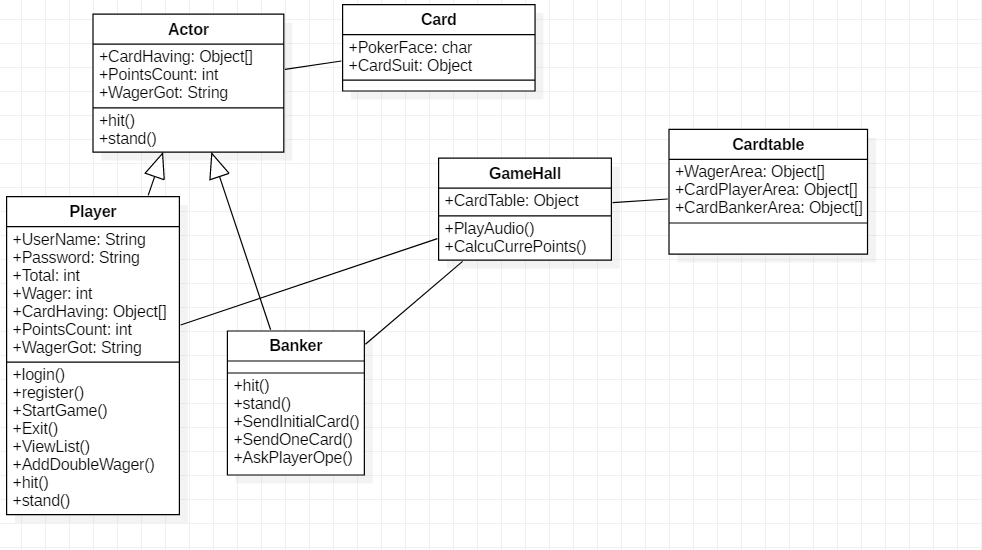
**说明：**数据库属性，存储玩家的登录密码

1. **金币数**

**说明：**数据库属性，存储玩家的金币数

## 核心类图

（画出系统中的关键类，以及相互之间的关系，类图可以使用StarUML画）

图 4核心类图

## 界面设计

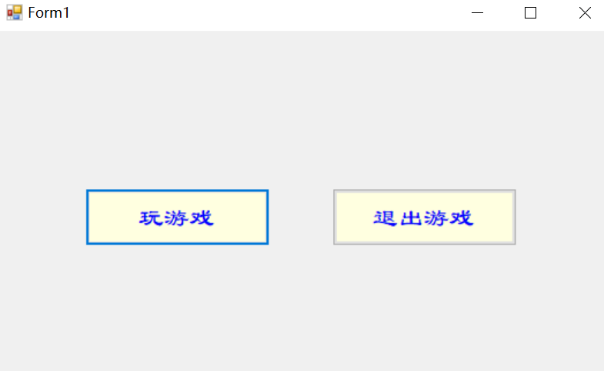
 

图 5 系统界面

# 项目过程中的问题及解决方案

鉴于一人之力和时间（邻近期末）有限，在老师的宽容和指点下只实现了项目的主体部分

项目过程中遇到了很多问题

**最初设计了三个窗体，游戏首页，主窗体，消息窗体**

在窗体的跳转上，为了在一局游戏结束后，给玩家发送游戏结果的信息，需要实例化消息窗体，并将主界面的交互性置为false, 在玩家关闭消息窗体后，再置为true, 但在实现过程中，关闭消息窗体后，主窗体就变得失去响应。在不置为false的情况下，依然无法响应，最后将公布结果信息放在主界面上显示。（为什么不用MessageBox， messagebox的字体大小，颜色无法设置，看起来实在不太适合公布结果（太丑啦！！！））

**关于生成一副牌**

最初想了两种方案

1，玩家要牌时实例化一张Card, 这样对于程序来说没有一副牌的概念，玩家要牌时生成一张随机牌。但这种实现和现实生活差距有些远，如果我后期要拓展，将游戏的难度分级，不太方便。与面向对象的思想也有些遥远。

2，在开始游戏时生成一副牌，玩家要牌，从这副牌里抽出一张牌给玩家，这里又有了两个方向（1）按顺序生成一副牌，玩家要牌时，生成一个随机数，将这个随机数对应的牌给玩家。（2）生成一幅随机牌，玩家要牌时，将最后一张牌给玩家。这里选择了（2）。

**关于异步编程**

最初为了保证发出的每张牌大数，小数的概率均匀分布，每开启一局，都会调用生成一副牌的方法，在这种情况下编译后，每次玩完一局，开始新的一局时，都会重新生成一副牌，但每次编译到这里，窗体就失去响应，为了解决这个窗体阻塞，将生成一副牌GenerateADeck和调用生成一副牌的play方法进行了异步编程，窗体阻塞是解决了，但程序仍然没有生成一幅牌出来。发现每次调试的时候在新的一局开始后，程序每次运行到GenerateADeck，就会跳出整个play方法，后面的代码都无法执行。考虑到GenerateADeck的开销可能比较大，将从每一局生成一副牌修改为每三局生成一副牌。但三局过后，情形和上述相同，目前不知如何解决。

**关于点数计算**

因为不能确定游戏参与者会拿到多少个A, 每次游戏参与者进行要牌后，都会重新计算游戏参与者拥有的点数，将每一张牌A放入临时变量List<Card> cA 中，计算多种可能，在不超出21点的情况下，算出最大值。