计算机程序设计课程实践项目报告

(Programmed in C#)

**完成人：曹高翔 PB20000061 潘廷岳 PB20000096**

**完成时间：2020年 12月**

摘 要

本项目的设计灵感来自于世界知名纸牌游戏UNO，能够在增进参与者友谊的同时提升玩家的反应速度和反应能力。同时，通过开发本游戏，笔者的编程能力和自我学习能力均得到了有效提升。

本程序基于Microsoft .NET Framework 4.8构建，所有代码完全在Microsoft Visual Studio 2019上使用C#编写。应用类别为Windows Form窗体应用程序，代码总量138K，总计3174行。

关键词：UNO，纸牌，单人游戏，多人联机

目 录

[1. 需求分析 3](#_Toc24442708)

[1.1 项目特点 3](#_Toc24442709)

[1.2 需求分析 3](#_Toc24442710)

[1.2.1 功能分析 3](#_Toc24442711)

[1.2.2 性能分析 3](#_Toc24442712)

[2. 概要设计 3](#_Toc24442713)

[2.1 系统总体设计 3](#_Toc24442714)

[2.2 主要数据结构 4](#_Toc24442715)

[2.3 模块定义与接口 4](#_Toc24442716)

[2.4 开发语言和开发工具 4](#_Toc24442717)

[3. 系统详细设计 5](#_Toc24442718)

[3.1 主体功能设计 5](#_Toc24442719)

[3.1.1 主函数 5](#_Toc24442720)

[3.1.2 菜单模块 5](#_Toc24442721)

[3.1.3 打印模块 5](#_Toc24442722)

[3.1.4 判断模块 5](#_Toc24442723)

[3.1.5 游戏逻辑模块 5](#_Toc24442724)

[3.2 其他功能设计 6](#_Toc24442725)

[3.2.1 其他模块 6](#_Toc24442726)

[4. 系统调试 6](#_Toc24442727)

[5. 系统运行 7](#_Toc24442728)

[6. 结论 7](#_Toc24442729)

[7. 参考文献 8](#_Toc24442730)

# 项目概述

## 项目特点

本游戏基于微软的.NET环境开发，编程语言为C#，具有较好的可移植性，能够在任意安装了.NET Framework 4.8环境的Windows操作系统上运行。本游戏使用Windows Form构建应用程序，既方便笔者开发，也方便使用者操作。同时，生成的项目窗口与Windows兼容性强，也符合Windows用户的使用习惯。

## 项目设计

相对笔者日常的计算机编程作业，此游戏的代码量与复杂程度都远不在一个数量级，需要更加细致地规划项目内容。最终我们决定将每个子功能独立出来封装成一个类库便于开发、调试和管理。本游戏主要包含以下项目：

1. UNO++（Windows 窗体应用（.NET Framework））：包含了服务器和客户端的代码实现，包含所有的窗体代码，是项目的主体。

2. Core（类库（.NET Framework））：包含了静态类Core，封装了游戏的内核代码，包含游戏的规则与具体实现。

3. User（类库（.NET Framework））：包含窗体使用的用户。

4. CardControl（Windows窗体控件库（（.NET Framework）））：包含窗体使用的牌控件，实现对鼠标的响应。

### 功能分析

1. 单机游戏，自娱自乐。
2. 多人联机，增进友谊。
3. 人机友好，窗体便于使用。
4. 游戏可玩性强，耐玩性好。

### 性能分析

1. 代码效率高，无卡顿
2. 通信效率高，低延迟
3. 网络通信部分使用异步编程，防止阻塞UI线程。

# 系统设计

## 系统总体设计

小游戏有简单的菜单选项，用来选择相应的功能，包括：

1. 用户设置。
2. 退出游戏。

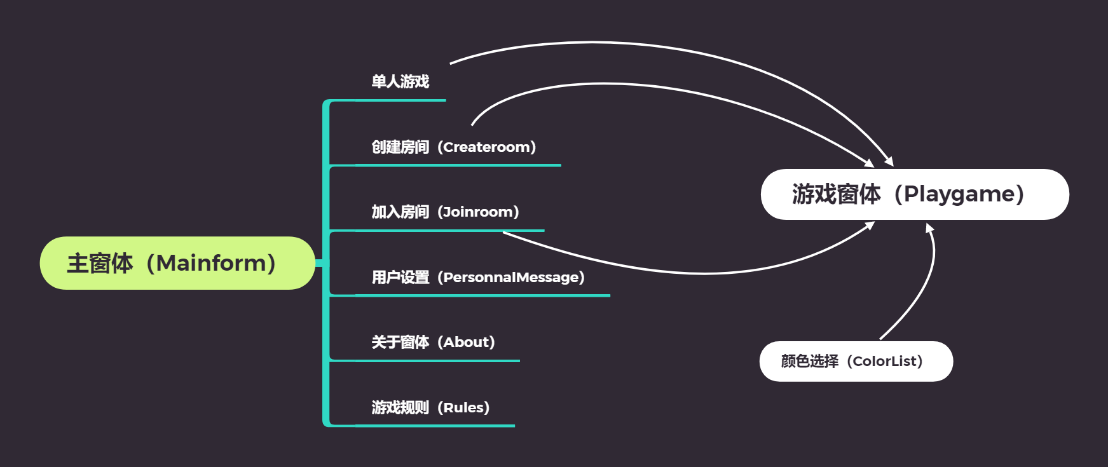
小游戏主窗体有简单的按钮，用来执行特定的操作，包括：

1. 开始单人游戏。
2. 创建服务器并准备与客户端建立连接。
3. 创建客户端并与服务器建立连接。
4. 设置游戏时使用的用户信息。

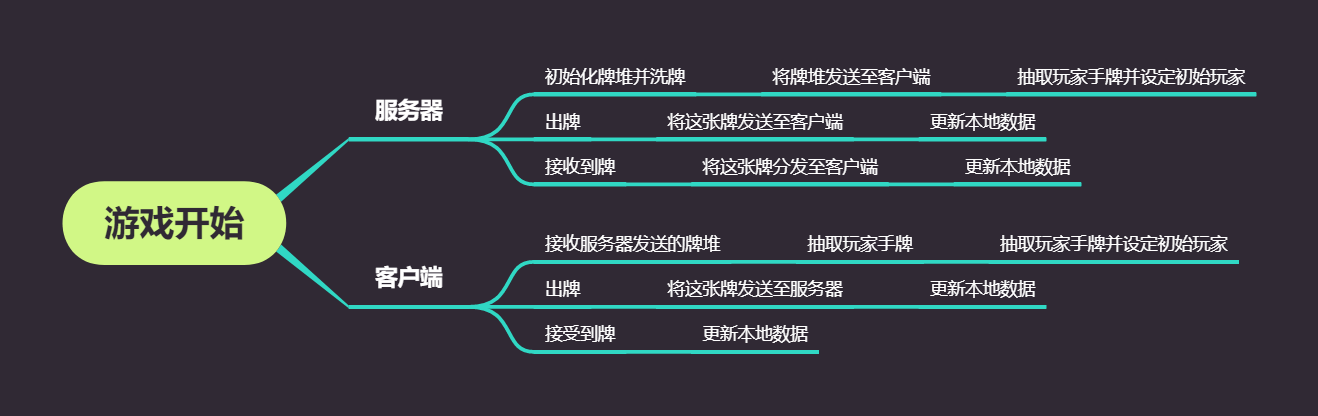
在游戏中应体现的功能包括：

1. 全自动的发牌、摸牌和流程跳转。
2. 玩家手牌的实时更新。
3. 胜利与失败的判定

该项目的窗体结构图如下：



该项目的游戏流程图如下：



## 主要数据结构

类：User（用户信息）, Card（卡牌）, Communication（网络通讯存储信息）

## 窗体设计

### 主窗体（Mainform）

应用程序的入口点，提供游戏各个内容的接口。包含启动单人游戏、创建房间窗体、加入房间窗体、设置用户信息窗体的按钮。

### 创建房间窗体（Createroom）

包含服务器代码的实现，使用Socket实现与客户端的连接和数据传输。要求用户输入房间号（端口号）。在用户创建房间后实时更新房间内的用户姓名，同时提供开始游戏按钮，使房主能够随时控制游戏的开始。

此窗体必须在设置用户信息后才能够访问。且用户一旦进入此窗体将无法返回主窗体。

### 加入房间窗体（Joinroom）

包含客户端代码的实现，使用Socket实现与服务器的连接和数据传输。要求用户输入IP和房间号（端口号）。当房主开始游戏时，此窗体将自动创建新的线程，并异步启动游戏窗体。

此窗体必须在设置用户信息后才能够访问。且用户一旦进入此窗体将无法返回主窗体。此窗体关闭时将询问是否关闭，并在关闭后终止游戏所有线程。

### 用户设置窗体（PersonalMessage）

用于设置用户信息的窗体，获取用户的用户名和ID，其中ID为游戏中的唯一标识符。

此窗体必须在设置用户信息后才能够访问。此窗体关闭时将询问是否关闭，并在关闭后终止游戏所有线程。

### 游戏规则窗体（Rules）

包含了游戏规则的详细说明。

### 关于窗体（About）

包含了游戏的版本号以及开发者信息。

### 游戏窗体（Playgame）

用于将内核代码运行的结果以图形界面的形式显示出来，同时接受用户出牌的信息反馈给内核代码。此窗体由创建房间窗体创建，或在客户端接收到游戏开始信息时在独立线程中以异步方式调用。

此窗体关闭时将询问是否关闭，并在关闭后终止游戏所有线程。

### 判断模块

该模块主要功能是逻辑判断功能模块，判断的内容主要是：

1. 游戏是否结束：判断游戏是否结束，根据是否存在玩家出完所有手牌来判断游戏是否结束。
2. 当前是否轮到该玩家出牌：判断这部分，主要是为了保证游戏顺序的正常进行。
3. 当前玩家手牌中是否有可出的牌：轮到该玩家时，通过对玩家全部手牌的检索，来判断玩家是否拥有可出的牌，若无则直接执行惩罚阶段。
4. 当前出牌玩家所选中的牌是否可出：通过对上家打出的牌与当前选中的牌进行处理，来判断选中之牌是否可出。

### 游戏逻辑模块

该模块是游戏的主要模块，游戏是需要有逻辑规则来保证趣味性的，该模块主要包括的功能函数有：

1. 随机化牌堆函数：保证抽牌的随机性。
2. 出牌合法性判定函数：保证出牌的合法性。
3. 玩家出牌顺序处理函数：保证出牌玩家顺序的合理性。
4. 玩家手牌维护函数：保证玩家手牌的有序性以及实时更新性。
5. 抽牌函数：用于实现玩家的抽牌操作。

### 网络通讯模块

该模块实现网络通讯相关功能。本游戏利用Socket使用TCP协议实现不同设备之间的网络通信和数据交互。本模块分为服务器端和客户端。

服务器端：设定网络终结点为本机网络的所有端口，并将Socket绑定至网络终结点，设置Socket为监听状态，为每一个客户端Socket创建一个新线程接受消息。实现将指定信息发送至所有客户端和将收到的信息发送至除发送者以外的所有客户端的功能。

## 开发语言和开发工具

本游戏使用.NET Framework 4.8中的C# 8.0编写，使用的开发工具是由美国Microsoft公司开发的 Microsoft Visual Studio 2019 Professional。

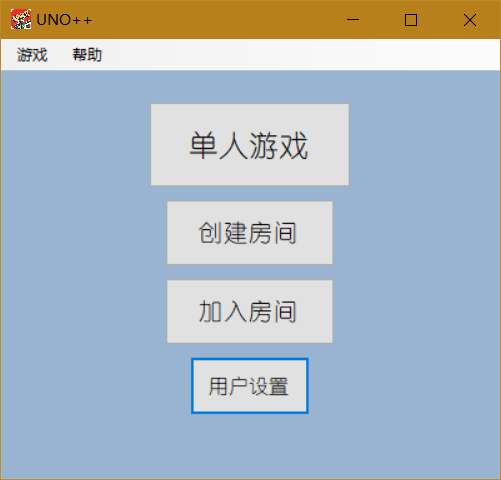
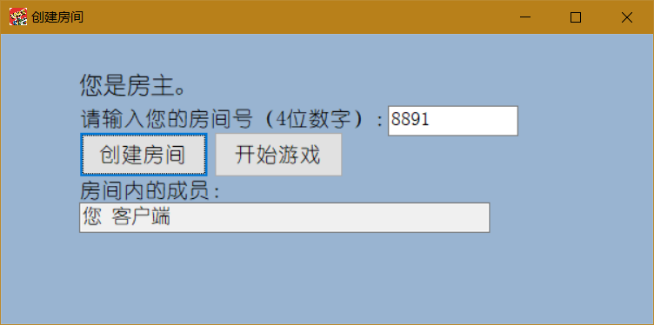
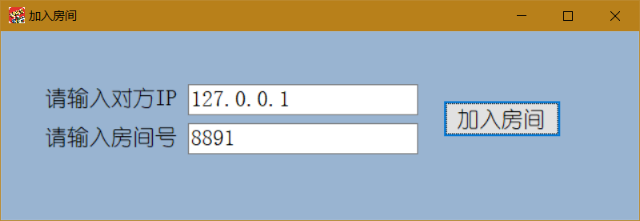
C#是由美国微软公司开发的新的高级程序设计语言，是微软.NET框架的主力语言，基于C++语言再次大幅提升，成为完全面向对象的程序设计语言。它与Java具有很多相似点，如都解释成中间语言，也与C/C++的语法具有较多相似之处，但C#又具备自己的特点与优势，如与Windows操作系统具有更高层次的兼容性等等。

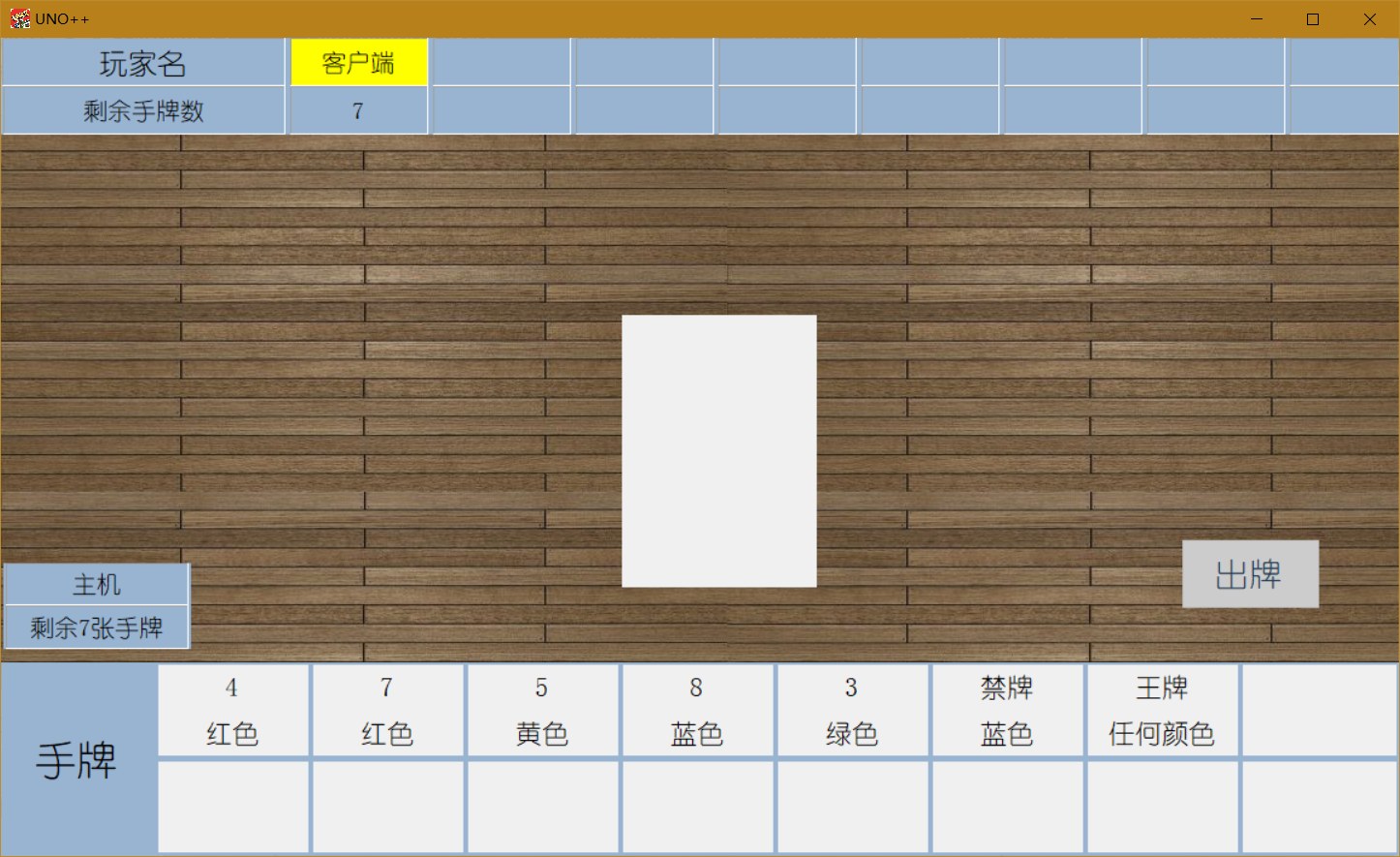
Visual Studio是微软开发的著名大型IDE，集成了相当强大的面向对象程序设计工具和调试器，尤其在开发可视化的桌面应用程序上具有巨大优势。同时，Visual Studio的强大的代码分析器和调试器也极大地方便了笔者的开发工作。

# 系统测试

## 系统运行

游戏的主要窗体截图如下：





介绍游戏界面、游戏控制方法。

说明游戏体验。（应找多个非项目设计者进行体验）

## 系统测试

本章给出对哪些模块做了哪些测试。例如多次运行是否出错，观察内存是否有泄露，CPU占用率等等。

/\*

下面是对系统测试的一些介绍，你应参考它来进行自己的测试，而不是复制粘贴它。

系统测试的目的是为了发现错误而执行程序的过程，成功的测试是发现了至今尚未发现的错误的测试。

测试的目的就是希望能以最少的人力和事件发现潜在的各种错误和缺陷，我们应该根据开发个阶段的需求。设计等文档或程序的内部结构精心设计测试实例，井利用这些实例来运行程序，以便发现错误的过程。

系统测试的原则应该遵循以下基本规则。

(1) 应该尽早并且不断地进行测试。测试不是在系统开发完之后才进行的，由于原始问题的复杂性、开发阶段的多样性以及参加人员的细条等因素，在开发过程中的各个阶段都有可能出现错误，因此，测试应该贯穿在开发的各个阶段，我们应该尽早纠正作物，消除隐患。

(2) 测试工作应该避免由本人来进行，一方面，开发人员往往不愿否认自己的工作，总认为自己开发的软件过程序没有错误，另一方面，开发人员往往很难发现自己的错误，具有局限性，所以我们应该找其他人来帮忙检查程序。这样会变得更客观，更有效。

(3) 设计测试方案时，不仅要确定输入数据，而且要根据系统功能来确定预期的输出效果和结果。将实际输出结果和预期结果相比较就能发现测试的对象是否正确，是否有一些隐患。

(4) 在设计测试用例时，不仅要设计有效合理的输入条件，还要包含不合理、失效的输入条件。测试的时候，人们往往习惯按照合理的、正常的情况下进行测试，从而忽略了对异常、不合理，意想不到的情况进行测试，而这些情况可能都是隐患，可能会造成系统的崩溃。

(5) 在测试程序时，不仅要检验程序是否做了该做的事情，还要检验程序是否做了不该做的事情。多余的工作会带来副作用，影响程序的效果，有时会带来潜在的危害或者错误。

(6) 严格按照测试计划来进行，避免测试的随意性。测试计划应该包括测试内容、进度安排、测试环境、测试工具和测试资料等。严格按照测试计划可以保证进度，使各方面都得以协调进行。

(7) 妥善保存测试计划、测试用例，作为软件文档的组成部分，为了以后维护的方便做了铺垫。

(8) 测试用例都是精心设计出来的，可以为重新测试或者追加测试提供方便。当纠正错误、系统功能扩充后，都需要重新进行测试，而这些工作重复性很高，可以利用以前的测试用例，或者在其基础上进行修改，然后在进行测试，这样科技简化系统的维护时间并且提高了维护的效率。

在系统调试过程中，我们可以利用一组简单的数据对程序进行调试，来观察各函数是否有出错现象。然后用复杂数据对程序进行调试。然后我们要对系统的各项能力进行测试，看看有没有漏洞或者死循环现象。

数据测试时，如果程序出现错误，要首先找到出错的函数，对函数中各个变量进行观察，有时变量很多，循环多重不要急，对各个数据的观察，对程序的调试有很大的帮助。你也可以先对几个变量熟悉后，在多重循环里，确定正确的数据，将其跳过调试，对有疑问的数据循环进行仔细观察。虽然这是比较省力气的活，但对复杂的数据错误并不是很有效，所以仔细观察每个数据的变化对调试的成功与否有决定性作用。

测试时，根本的错误井不一定在运行错误的函数内，或许在前面的函数内已经奠定了这些错误数据的基础，需要反过头来，对所有的的函数进行一项项的调试！从程序运行的第一个函数开始调试。按照程序运行的步骤下去。一个一个地确定函数的正确性，每确定一个函数，在以后的调试中可以减少对该函数的调试次数，或者直接跳到下一个函数。所谓的各个击破便是如此。对头文件中的各个函数的返回值和它所要传入的参变量及其功能更是需要注意。

调试时，需要对错误缩小范围，从整个程序到指定函数，再到指定区域，再到指定行，在对程序缩小范围或对编译没通过的程序缩小错误范围时可以用”/\*\*/”来进行调试，确定正常范围，缩小错误范围。

\*/

# 结论和体会

经过以上各章节的功能，已经基本完成了UNO纸牌游戏的开发。UNO纸牌游戏是一个趣味性和逻辑性很强的游戏。该游戏通过较为简单的纸牌玩法，让玩家能够充分锻炼自己的判断能力以及如何出牌的逻辑能力。游戏过程中友好的小提示给予玩家参与的满足感和亲切感，游戏的背景界面则设计成了温和的、自然的木板，让玩家在更舒适的视觉环境中进行游戏。同时，本游戏支持多人游戏，旨在促进朋友间的友谊，享受朋友带来的欢乐。从以上几点，我们可以得出一些结论。一个容易被接受的游戏，应该有几点需要被考虑。

一、界面的友好。

二、交互功能的灵活友好。

三、玩法的简单和创新。

遗憾的是，由于制作时间较短，本游戏还有部分功能有待实现。我们会结合玩家反馈，考虑对本游戏进行进一步的创作。

通过这次编程，我们深深的感受到代码的变量命名和代码内注释格式等对于这类需要拼接的程序的重要性。良好的编写习惯，不但有助于代码的移植和纠错，也有助于不同人员之间的协作。 我们还要有模块化思维能力，通过完成对一个个独立模块的编写，以及最终将其组装成我们所需要的产品，这正是一个大工程所需要，也是一个程序员的基本思想。同时，该游戏的制作也充分锻炼了编程能力以及自主学习能力。用户界面的编写所涉及的大量知识让我们不得不查阅大量的书籍和资料，并将所了解到的知识融会贯通，这充分锻炼了我们自主查找知识的能力；网络通信方面更是花费了我们大量时间和精力去学习和完成；游戏逻辑方面的设计则考验了我们设计游戏的逻辑性和思维的严密性、连贯性。更重要地，我们应该学会对自己编写的项目进行经常性的总结，这样才能在总结中认识到自己的不足，从而得到更快的进步。

# 参考文献

1. C#Windows游戏设计 毕文斌、孙明亮 清华大学出版社
2. C#入门经典（第八版）Benjamin Perkins 清华大学出版社