C#课程论文

2017301500250-季祥昊

目录

[C#课程论文 1](#_Toc43315160)

[1. 系统介绍 2](#_Toc43315161)

[1.1概述 2](#_Toc43315162)

[1.2游戏规则 2](#_Toc43315163)

[1.2.1 普通模式 2](#_Toc43315164)

[1.2.2挑战模式 2](#_Toc43315165)

[2. 设计思路 2](#_Toc43315166)

[3. 关键技术 4](#_Toc43315167)

[3.1登录与注册 4](#_Toc43315168)

[3.1.1数据库设计 5](#_Toc43315169)

[3.1.2用户名密码认证 5](#_Toc43315170)

[3.2游戏初始化 5](#_Toc43315171)

[3.2.1从图库获取图片 5](#_Toc43315172)

[3.2.2拼图打乱 6](#_Toc43315173)

[3.3游戏机制 6](#_Toc43315174)

[3.3.1图片移动 6](#_Toc43315175)

[3.3.2挑战模式 7](#_Toc43315176)

[4. 个人贡献 8](#_Toc43315177)

[5. 个人成果展示 10](#_Toc43315178)

[5.1登录注册界面 10](#_Toc43315179)

[5.2拼图的制作及重新排列 10](#_Toc43315180)

[开发体会 12](#_Toc43315181)

# 系统介绍

## 1.1概述

拼图是一个非常经典的游戏，也是一种广受欢迎的智力游戏，它的变化多端，让人百玩不厌。此外，拼图游戏还有许多的变种，此项目是实现8数码式的拼图游戏。

## 1.2游戏规则

### 1.2.1 普通模式

游戏开始之前，系统会先从图片库随机选择一张图片，根据玩家选择的难度将其切割并打乱，然后显示在玩家眼前，玩家需要做的就是将打乱后的图片还原成原来的样子。由于此拼图游戏是8数码形式的拼图，图片被打乱之后会存在一个空白格，玩家只允许将在空白格的上、下、左、右四个位置的图片格与空白格进行交换，因此要还原成原来的图片存在着一定的难度，当玩家成功还原图片时，系统便会跳出相应的提示信息来祝贺玩家完成拼图。玩家在还原拼图的过程中，系统也会记录玩家的移动步数，可以让玩家知道自己此次游戏的游玩情况。

### 1.2.2挑战模式

除了普通的模式，我们还额外设计了挑战模式，用来让玩家挑战自己的记录。挑战模式则是在有限的时间内完成玩家自己指定难度的拼图，系统会记录玩家游玩时的移动步数，系统会记录该难度下最少的移动步数并将其作为最高纪录，玩家可以尝试用更少的步数完成拼图来突破自己的纪录，以达到挑战的目的。

# 设计思路

本项目是为了实现一个拼图游戏，由于拼图游戏存在许多变种，从而让游戏存在各种各样的玩法，本系统的目标是实现一个8数码式的拼图游戏，其功能组成如下：

**登录功能：**我们设计了挑战模式是想让玩家不断冲击自己的最高纪录，因此为了将不同的玩家区分开，我们设置了一个登录的界面，系统会将玩家输入的用户名和密码与数据库中存储的进行比对，正确则让玩家进入游戏；错误则会提示错误消息。

**注册功能：**为了让未注册的玩家也能游玩游戏，我们加入了注册功能，系统会判断玩家输入的用户名是否与数据库中已有的用户名重合，未重合会提示玩家注册成功；否则向玩家报错。

**拼图制作：**系统从已有的图片库随机选择一张图片，将其按难度切割成由n\*n个图片块组成的拼图并打乱摆放顺序。

**拼图移动：**玩家通过鼠标拖动图片进行移动，玩家能选定的图片块只有空白格四周的图片块，即位于空白格上、下、左、右方向上的4个图片块，鼠标按住选定并在空白格上松开后，两个图片块便会交换位置。

**难度选择：**难度决定图片被切割后形成的n\*n个图片块中n的值，由玩家在游玩过程中自行决定，初始时难度被默认设置为2。

**切换图片：**玩家可以自行切换图片库中的图片，点击之后系统则在图片库中重新选一幅图进行切割打乱。

**试玩新图：**让玩家可以自己导入新的图片来进行游玩。

**图片重排：**如果玩家觉得当前拼图难以还原，则可以将该拼图重新打乱后再次尝试。

**挑战模式：**让玩家在有限时间内尝试以最少的步数完成拼图。

**胜利判定：**判断玩家是否将图片还原。

# 关键技术

## 3.1登录与注册

我们在开头设置了**登录**及**注册**功能，用以区分不同的玩家，登录及注册功能的流程图如下：

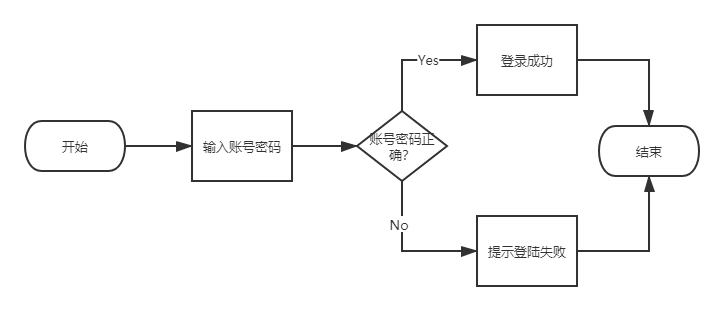
****

图3.1 登录功能

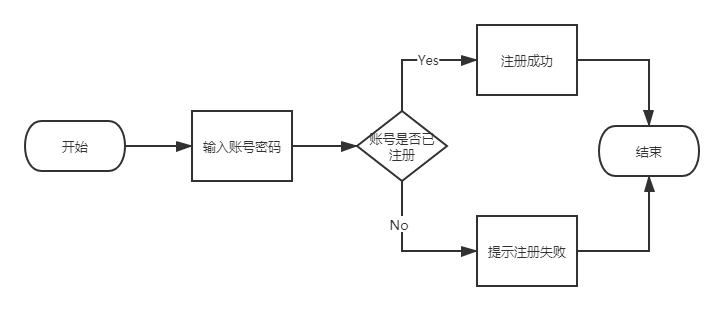


图3.2 注册功能

### 3.1.1数据库设计

玩家的用户名、密码及最高纪录我们用数据库进行保存，我们设计的数据库表设计如下，id为主键索引，user为账号，pass密码，score记录当前玩家的最佳记录，默认值为1000

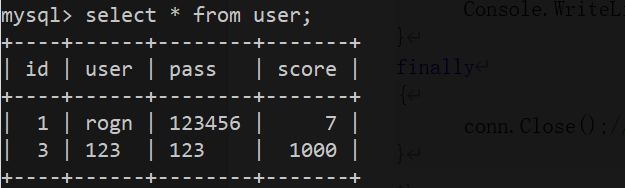


图3.3 user表

### 3.1.2用户名密码认证

添加两个输入框，分别代表用户名输入框（TextBox1）和密码输入框（TextBox2），利用其Text属性获得对应的值，再添加两个Label按钮，表示登陆和注册，分别监听其上的点击事件。首先创建MSqlConnection对象，并建立与Mysql的连接，将username和passward做字符串的连接操作生成一条查询语句。如果能查到该记录，说明该用户名和账号存在且与匹配，否咋说明该用户尚未注册，登陆失败。

但是这里有一个问题，游戏主界面需要得到当前玩家的用户名，如何将用户名从登陆界面传递给主界面呢？我们采用本地存储的方式，将当前用户名存到一个文本文件中。

注册的情况与登陆十分类似，只是将sql的查询语句换成插入语句

## 3.2游戏初始化

### 3.2.1从图库获取图片

在主程序的根目录下有一个Picture文件夹，用于存放游戏所用到的图片素材，当程序启动时会从中随机选取一张。对于选取的图片，切分成n\*n的小块，并进行随机打乱。具体的，通过 Application.StartupPath.ToString() 可以获取到程序当前所在的目录，把它与Picture、jpg以及随机出来的数字进行拼凑作为图片的路径。通过路径，利用 Image.FromFile(path) 很容易获取图片。除此之外，我们需要将大图切成n\*n个小图并绘制出来，调用的 img.GetThumbnailImage(Width, Height, null, IntPtr.Zero)。

其中有个问题，我们需要空缺一个位置（游戏设定只有与空白块相邻才能交换），这要怎么实现呢？从一个纯白图片中截取同等大小的块即可。

### 3.2.2拼图打乱

拼图游戏的主要内容就是将打乱的图进行还原，所以一个等概率、高效的算法极为重要。

**Fisher–Yates shuffle**算法由Ronald Fisher和Frank Yates于1938年提出，在1964年由Richard Durstenfeld改编为适用于电脑编程的版本。其思路是：将最后一个数和前面任意 n-1 个数中的一个数进行交换，然后倒数第二个数和前面任意n-2个数中的一个数进行交换，重复此操作，直到第一个。可以证明按此方法生成的某个排列其概率是1/n!，也就是说每个排列出现的概率相同，达到了完全随机的效果。

## 3.3游戏机制

### 3.3.1图片移动

玩家通过交换空白图块和其相邻的块，逐步还原，以成功通过游戏。图片交换时，我们需要监听鼠标左键单击事件、鼠标左键释放事件、鼠标移动事件，以及鼠标点击的坐标，将这些事件与每一个图片块绑定。

当鼠标左键点击时，我们通过GetPictureBoxByHashCode（）得到每个图片块的哈希码，由于哈希码的唯一性，从而可以获取鼠标点击位置的图片，记录点击的坐标，同时将全局变量isDrag设置为true。这样当鼠标移动时，如果isDrag==true，则可以拖动图片，实现位置更新

当鼠标左键释放时，首先判断移动到的位置是否超出图片范围，其次判断是否移动到空白块。如果满足这两者，调用Judge()函数检测是否拼图成功。其中判断是否相邻有四种情况，按上、下、左、右依次考虑。

每移动一次，都需要进行以上过程，同时还需要将移动步数加1.

### 3.3.2挑战模式

1. **计时模块设计**

挑战模式需要在30秒内完成拼图，如果所用时间少于历史记录，还应更新相应的数据表。

采用System.Windows.Forms.Timer类型，当一个System.Windows.Forms.Timer类被构造时，当前定时器会和当前线程进行关联。而当计时器的计满后，一个定时器消息将被插入到当前线程的消息队列中。当前线程逐一处理消息中的所有消息，并一一派发给各自的处理方法。System.Windows.Forms.Timer类型并没有涉及多线程的操作，因此是线程安全的，不会发生回调方法重入的问题。

其对象可以绑定一个函数 timer1\_Tick()，获取当前计数值，并用一个label显示：

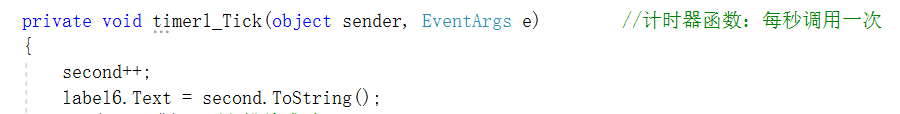


图3.4 计时器代码

1. **判断是否挑战成功**

首先从文件record.txt中读取当前玩家的用户名，通过该用户名去mysql数据库中查找分数记录，如果当前消耗时间小于记录值，则挑战成功；否则挑战失败。当然，如果挑战成功，需要在数据库中update分数。



图3.5 挑战成功代码

1. **结束挑战模式设计**

当玩家没有选择挑战模式时，自然也不能显示出“结束挑战模式”按钮，考虑到玩家可能经常在两个模式之间互换，所以我们并没有将该控件删除，而是设置 label2.Visible = false; ，并将计数值置零

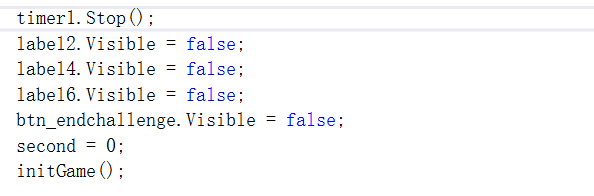


图3.6 挑战结束代码

# 个人贡献

我在此项目中的分工是实现拼图的制作，包括对原始图片进行压缩、切割及打乱，登录注册界面的设计、图片的移动、游戏难度的选择及图片的重新排列功能。

关于拼图的制作过程及图片的移动功能的实现已在上文中提及，登录注册的界面如图4.1.

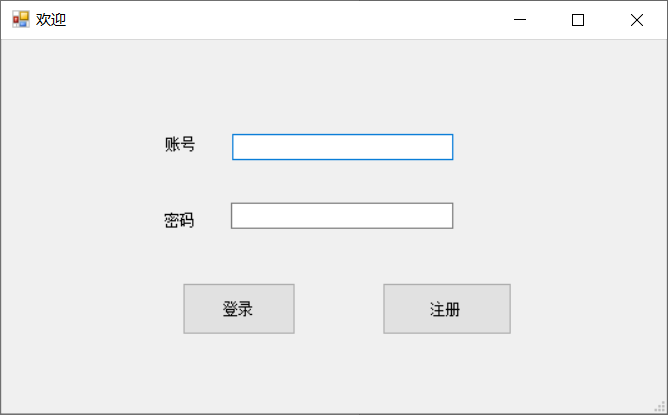


图4.1 登录注册界面

拼图的难度选择，我添加了一个NumericUpDown组件，用于选择难度等级，后台根据在此选择的数字把拼图划分成对应的n\*n个拼图块，例如难度选择为2，那么拼图将会被制作成3\*3个块。



图4.2 NumericUpDown组件

将图片进行重新排列的功能，就是再次调用一遍拼图制作功能中会用到的函数，但是此功能去掉了拼图制作中的随机选择图片这一步，因为将图片重新排列相当于是重新游玩当前的这幅图片。

# 个人成果展示

## 5.1登录注册界面

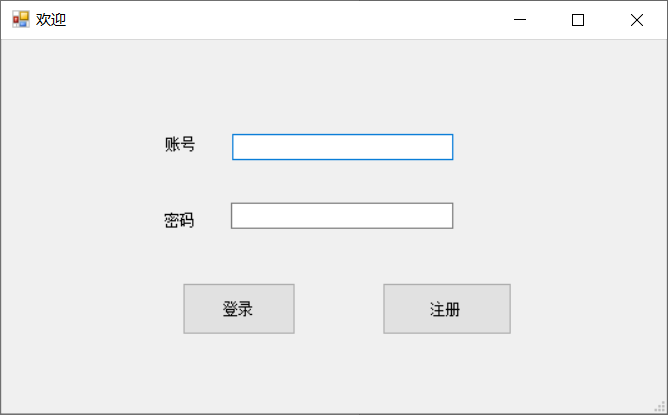


图5.1 登录注册界面

## 5.2拼图的制作及重新排列

  
图5.2 原始图片

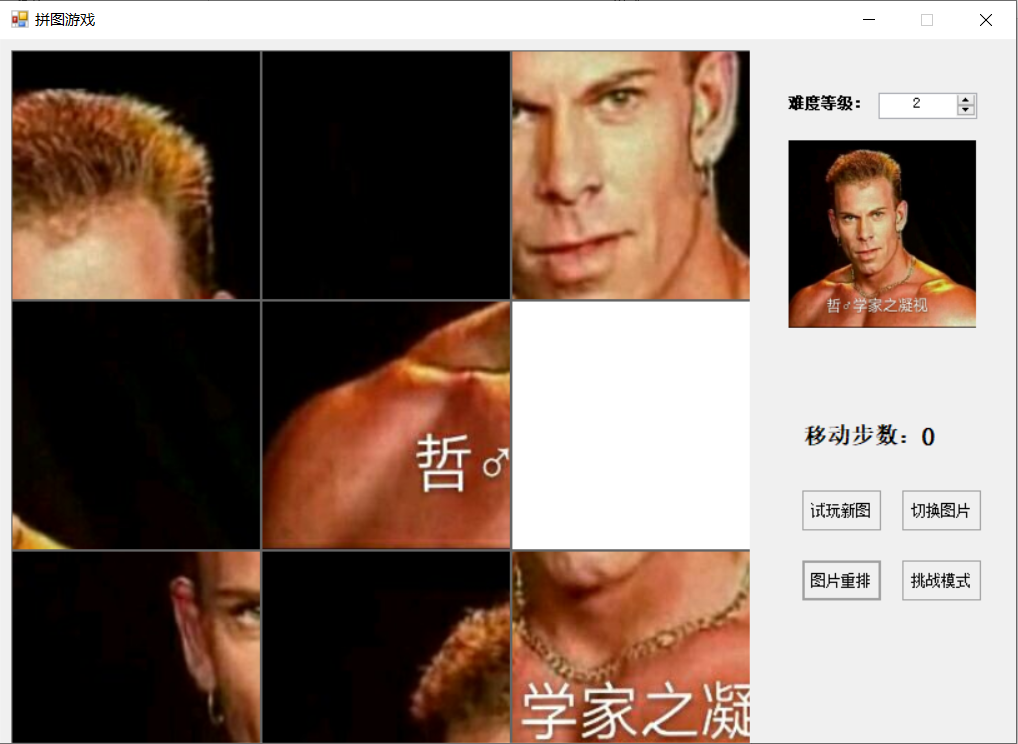


图5.3 制作成拼图之后的图

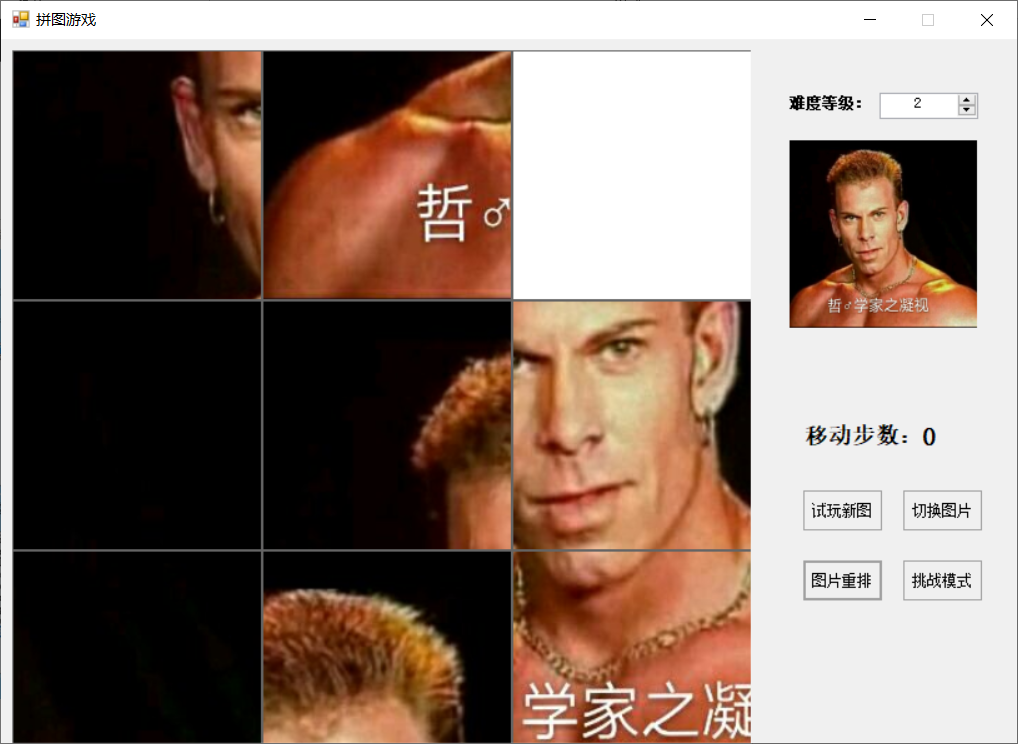


图5.4 重新排列之后的图

# 开发体会

此次的项目开发也是以小组的形式完成，我和班上的另一个同学刘发荣组了队，我身为组长决定好了我和他两个人的分工，我很高兴的一点是我的组员是那种能按时完成分配任务的人；因为之前的一些小组作业的经历，我遇到过那种光会说却几乎不动手做的组员，常常是快到规定的时间时才刚开始做分配给他的任务，这样无疑会增大组内所有人的工作量，也因此会阻碍整个项目的推进进程。如果组内的大家都是说到做到干实事的人，那么一起合作做的小组项目不仅对自己的代码能力及自学能力会有一个非常大的提升，同时也能很有效地提高自己的团队协作、与人沟通的能力，这对整个小组的所有人都是获益匪浅的。

在开发的过程中，由于我自己同时还是这个小组的组长，因此我得对整个项目都有一个清晰的了解，不仅仅得要明白我自己负责的那一块，剩下给我组员负责的那一块任务我也得要有明确的了解。但在开发的初期，由于自己缺乏C#这门编程语言的使用经验，我不了解它的一些独有的特性以及一些有效的库函数，导致了我在这个阶段编写代码时屡屡碰壁，开发的进度推进十分缓慢；但也正是在这个阶段，我通过了上网查阅资料、参考别人的项目源码，解决了我在这个项目中遇到的大部分问题，也加快了我在中后期开发的速度与效率。在这个过程中我收获了许多，像是清楚地构造出整个项目的流程，明晰项目中使用的系统库函数的功能、整个项目的类层级已经类之间的分工等等，这些对我的逻辑思维和逻辑构造的能力是一个非常大的提升，这对我以后的学习及工作有很重大的意义；同时，通过代码来实现各个类和组件以及他们之间的功能，也是一个锻炼我的代码能力的好机会；此外，在开发初期阶段，我在查阅各种资料的过程中，也是在不断地深入学习C#这门语言的精髓，了解课堂以外的知识，同样对以后的学习及工作有很大的帮助。

最后，我很感谢老师能够布置这么一个作业给我，这门作业不仅有助于我加深对课堂知识的理解，也有助于我更加深入的了解C#这门语言的精髓；在提升个人能力的方面，这门大作业在对我的自学能力、逻辑分析能力、团队协作与沟通能力、编写代码的能力都有非常大的帮助，完成这个大作业，让我获益匪浅。