**北京交通大学**

**软件工程**

**连连看设计文档**

题目： **连连看小游戏**

组长姓名： 程威

组员姓名： 尹业立 张小明

专 业： 软 件 工程

完成时间： 2018年 5月----2018年 6 月

北京交通大学制

2018 年 6月16日

# 项目报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目目标：**  **在三周时间内，制作出连连看基本游戏，包括项目启动，项目计划，需求分析，系统设计，开发“创建工程”功能，开发“绘制欢迎语”功能，开发“主界面设计”功能，开发“游戏界面设计”功能，开发“绘制游戏地图”功能，开发“同色消子”功能，搭建程序框架，搭建程序框架，以及网络通信，背景音乐等基本功能。** | | | | | |
| **项目设计进度计划** | | | | | |
| **起至日期** | | **工作内容** | | | **备注** |
| 5月25日  6月6日  6月18日 | | 书写立项报告，并且做出图形界面，及相关的小功能  制作出连连看一般必备的功能  制作出连连看人机对战的功能 | | |  |
| **参考文献、资料索引** | | | | | |
| 序号 | 文献、资料名称 | | 编著者 | 出版单位 | |
| [1] MFC文档  [2] CSDN官网  [3] WINSOCK | | | | | |

**目录**

目录

[项目报告 2](#_Toc517106779)

[摘 要 4](#_Toc517106780)

[第1章：需求分析 5](#_Toc517106781)

[1.1连连看背景 5](#_Toc517106782)

[1.2程序设计 6](#_Toc517106783)

[程序需求分析 6](#_Toc517106784)

[程序功能分析 6](#_Toc517106785)

[第2章：系统设计报告 9](#_Toc517106786)

[2.1注册，登陆（网络功能） 9](#_Toc517106787)

[2.2 游戏核心模块 16](#_Toc517106788)

[2.3玩家获胜排行榜 27](#_Toc517106789)

[2.4 声音控制 34](#_Toc517106790)

[2.5 人机对战 38](#_Toc517106791)

[2.6界面绘制 42](#_Toc517106792)

# 摘 要

游戏“连连看”顾名思义就是找出相关联的东西，这个连连看在网上基本是用在小游戏中，就是找出相同的两样东西，在一定的规则之内可以做为相关联处理。“连连看”的发展经历了从桌面游戏、在线游戏、社交游戏三个过程。

游戏“连连看”是源自台湾的桌面小游戏，自从流入大陆以来风靡一时，也吸引众多程序员开发出多种版本的“连连看”。这其中，顾方编写的“阿达连连看”以其精良的制作广受好评，这也成为顾方“阿达系列软件”的核心产品。并于2004年，取得了国家版权局的计算机软件著作权登记证书。

# 第1章：需求分析

## 1.1连连看背景

游戏“连连看”顾名思义就是找出相关联的东西，这个连连看在网上基本是用在小游戏中，就是找出相同的两样东西，在一定的规则之内可以做为相关联处理。“连连看”的发展经历了从桌面游戏、在线游戏、社交游戏三个过程。

游戏“连连看”是源自台湾的桌面小游戏，自从流入大陆以来风靡一时，也吸引众多程序员开发出多种版本的“连连看”。这其中，顾方编写的“阿达连连看”以其精良的制作广受好评，这也成为顾方“阿达系列软件”的核心产品。并于2004年，取得了国家版权局的计算机软件著作权登记证书。

随着Flash应用的流行，网上出现了多种在线Flash版本“连连看”。如“水晶连连看”、“果蔬连连看”等，流行的“水晶连连看”以华丽界面吸引了一大批的女性玩家。

2008年，随着社交网络的普及和开放平台的兴起，“连连看”被引入了社交网络。“连连看”与个人空间相结合，被快速的传播，成为一款热门的社交游戏，其中以开发者Jonevey在Manyou开放平台上推出的“宠物连连看”最为流行。

网络小游戏、网页游戏越来越受网民欢迎，除了玩的方法简单外（不像其他游戏还需要注册下载繁琐过程），很多游戏不乏经典。连连看游戏就是典型。

不管走到哪个网页游戏网站，连连看游戏总是排在受玩家欢迎排名的前5位，休闲、趣味、益智是连连看玩不厌的精华，且不分男女老少、工薪白领，是一款适合大众的经典网络、单机休闲小游戏。

## 1.2程序设计

### 程序需求分析

基于连连看的背景概述，程序需要具有登陆功能，注册功能，而且游戏分为三大模式，分别为基本模式，简单模式，以及游戏模式。游戏为了创新，还加入了人机对战模式，方便游戏者自己锻炼自己的反应能力。此外，还加入了排行榜，一遍玩家查阅自己在当前游戏中的排名。在游戏主界面，设有帮助按钮，可以帮助玩家方便清楚游戏模式，此外还设置暂停按钮，方便游戏暂停的时候去忙其他事情，还有提示按钮，可以解决自己无法找到旗子的问题，重排按钮方便打乱棋盘继续进行游戏，背景音乐，消子音乐更增加了游戏的趣味性。

### 程序功能分析

界面设计

1. 注册界面
2. 登陆界面
3. 菜单界面
4. 设置界面
5. 人机对战界面
6. 排行榜界面
7. 本地游戏主界面
8. 帮助界面

注册，登陆（网络功能）

玩家进入游戏会首先进入登录界面。每个玩家至少需要一个固定的唯一的id的pin。新玩家可以注册，输入账号和密码，在网络连接正常的情况下，注册成功然后登陆即可。若网络不正常，无论是注册还是登陆都会显示失败。

游戏模式

1. 简单模式
2. 中等模式
3. 困难模式

人机对战

左侧棋盘为用户使用的区域，而右侧为电脑消除，在该模式下，用户可以重排，可以提示，也可以选择认输，认输不计入排行榜，只有胜利才会计入排行榜，也可以选择背景音乐以及消去音效。

设置功能

菜单栏的设置：通过菜单栏可以选择背景音乐的开关以及消去音效的开关。

设置界面的设置：通过设置界面也可以选择背景音乐的开关以及消去音效的开关。并且二者可以交互控制彼此。

帮助功能

主要为一个对话框，里面写的游戏规则，供玩家参考以及阅读。

排行榜

每次游戏结束玩家可以选择保存或不保存游戏成绩。若保存游戏成绩，则成绩被写入phb.txt文件。当有人查看排行榜时，系统读取phb.txt文件中保存的信息，并且按分数高低进行排列，从高到低依次显示。

提示功能

提示的实现通过调用bool Game::getHint()对地图 map 所有点遍历，调用 函数bool Game::judge(MyPoint start, MyPoint end)找出匹配的两个点。

重排功能

重排的实现的基础是对 map 的重新排列，通过调用void Game::rearrangeMap()实现，思想与 初始化 map的void Game::initMap()类似:创建一个新的长度为 difficulty \* images 的一维数组 s，将map值拷贝给 s，之后调用void randomMapWithSource(int \* source)打乱所有元素值顺序，再将 s复制给map；

暂停功能

在暂停的时候，所有棋子从棋盘消失，计时以及计分暂停工作，开始按钮，提示按钮，以及重拍按钮将不可用，暂停按钮将变为继续游戏按钮，再次点击即可继续游戏。所有按钮恢复，且游戏计时以及计分继续。

消子方式

判断点击的两图片是否匹配的函数为 judge()函数，该函数每次被调用时重新初始化 辅助数组 int \*\* visited，对其他成员重新辅助后，根据传入的两个点对 一维数组MyVector \*dirct 排序，以判断优先从哪一个方向搜索，之后根据 MyVector \*dirct数组从四个方向依次调用bool DFS(MyPoint p, int direction, MyVector \*dirct) 进行搜索；bool DFS(MyPoint p, int direction, MyVector \*dirct)函数以深度优先的思想，每次以一个方向访问 map 的辅助数组 visited，对值为非零的点若为终点（两点匹配），则返回true，否则false，对值为0的点则继续以某个方向递归访问下一个点，并设其值为-1，对方向改变的拐点将其添加到 记录所访问路径的链表LinkedList<MyPoint> \*path中，当path 大小超过3即路线线段超过3条时，该路径不符合，函数返回false；

连线绘制

主要获取设备DC以及棋子的相关XY坐标，使用异或的方式绘制，这样在擦除的时候就可以按照原路径在绘制一遍最终擦除，且为了可以让用户看见线，时间设置恰好适中。绘制线的时候根据拐点不同分为三种，一种是没有拐点，一种是只有一个拐点，另一种是有两个拐点，进行条件判断，然后绘制线条。

用户相关属性

class User有成员： string name，string password，int id，static int idCount，LinkedList<Record> records;功能如其名，其中records记录每次游戏数据（用户id，时间，分数等），为LinkedList<Record>类型

音乐播放

播放功能的实现封装在 MusicPalyer类中，class Mci 底层实际上使用Windows API函数 mciSendStringA。当要播放音频时，声明一个 MusicPlayer 实例后，调用其成员函数 bool MusicPlayer::load(const std::string& \_filename) 加载音频文件后，再调用bool MusicPlayer::playAndRepeat(bool repeated, int start\_ms, int end\_ms)播放音频文件；

进度条

主要使用MFC类的进度条组件，刚开始的值设置为零，然后采用每消除一对，进行进度条递增，最终填满进度条，然后重开游戏的时候清空重新继续。

# 第2章：系统设计报告

## 2.1注册，登陆（网络功能）

玩家进入游戏会首先进入登录界面。每个玩家至少需要一个固定的唯一的id的pin。新玩家可以注册，输入账号和密码，在网络连接正常的情况下，注册成功然后登陆即可。若网络不正常，无论是注册还是登陆都会显示失败。

//登陆相关代码

CString s1, s2;

GetDlgItemText(IDC\_EDIT2, s1); //第一个变量为控件名，第二个为赋值的地址

GetDlgItemText(IDC\_EDIT1, s2); //第一个变量为控件名，第二个为赋值的地址

std::string strID(CW2A(s1.GetString())); //CString 转string

std::string strPW(CW2A(s2.GetString())); //CString 转string

if (s1.GetLength() == 0) {

AfxMessageBox(\_T("请输入账号！"));

}

else {

string sendstr = strID + "##" + strPW;

int ok = hasifOnserver(sendstr);

if (ok == 1)

{

AfxMessageBox(\_T("登录成功！"));

user->setId(atol(strID.c\_str()));

user->setPassword(strPW);

MENU \*menu = new MENU();

menu->Create(IDD\_MENU);

menu->ShowWindow(SW\_SHOW);

this->hideWindow();//隐藏主窗口

//CDialogEx::OnOK();

}

else {

AfxMessageBox(\_T("登录失败！"));

}

}

//注册

CString s1, s2;

GetDlgItemText(IDC\_EDIT2, s1); //第一个变量为控件名，第二个为赋值的地址

GetDlgItemText(IDC\_EDIT1, s2); //第一个变量为控件名，第二个为赋值的地址

std::string strID(CW2A(s1.GetString())); //CString 转string

std::string strPW(CW2A(s2.GetString())); //CString 转string

if (s1.GetLength() == 0) {

AfxMessageBox(\_T("请输入账号账号！"));

}

else {

string sendstr = strID + "##" + strPW;

int ok = addIDOnserver(sendstr);

if (ok == 1)

{

AfxMessageBox(\_T("账号注册成功！"));

CDialogEx::OnOK();

}

else {

AfxMessageBox(\_T("此账号已存在！"));

}

}

//网络客户端

#include "stdafx.h"

#include "Client.h"

int hasifOnserver(string id) {

int has = 0;

/\*加载winsock文件\*/

WSADATA wsaData;

WORD sockVersion = MAKEWORD(2, 0);

WSAStartup(sockVersion, &wsaData);

/\*创建服务器端的套接字\*/

SOCKET s = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (s == INVALID\_SOCKET)

{

printf("Failed socket()\n");

WSACleanup();

//system("pause");

}

/\*socket中装入地址信息\*/

sockaddr\_in servAddr;

servAddr.sin\_family = AF\_INET;

servAddr.sin\_port = htons(8800); /\*接收服务器13端口号\*/

//servAddr.sin\_addr.S\_un.S\_addr = inet\_addr("218.249.50.112");

//servAddr.sin\_addr.S\_un.S\_addr = inet\_addr("172.30.116.184");//my

//servAddr.sin\_addr.S\_un.S\_addr = inet\_addr("172.30.65.254");//cw

servAddr.sin\_addr.S\_un.S\_addr = inet\_addr("127.0.0.1");//本地ip

/\*连接服务器\*/

if (connect(s, (sockaddr\*)&servAddr, sizeof(servAddr)) == -1)

{

printf("Failed connect()\n");

WSACleanup();

//system("pause");

}

/\*接收数据并打印到屏幕上\*/

char buff[1];

send(s, id.data(), 256, 0);

int nRecv = recv(s, buff, 1, 0);

if (nRecv>0)

{

//printf("接收到数据：%c\n", buff[0]);

if (buff[0] == '1')

{

has = 1;

}

}

else

{

cout << "接受消息失败！" << endl;

}

return has;

}

int addIDOnserver(string id) {

if (hasifOnserver(id) != 1) {

//cout << "has not it and add one" << endl;

for (int i = 0; i < id.length(); i++)

{

if (id[i] == '#')

{

id[i] = '$';

}

}

hasifOnserver(id);//注册新的账号

return 1;

}

else

{

//cout << "has it" << endl;

return 0;

}

}

int send(string str) {

return hasifOnserver(str);

}

## 2.2 游戏核心模块

游戏核心模块是程序的关键，该模块的设计直接关系到程序的运行速率和运行结果的正确与否。该模块封装在 Game类中，MyPoint类和MyVector类为两个辅助类。

Game类中成员int \*\* map为二维数组，记录图片排列情况；

每次开始游戏时，调用void setDifficulty(int d)函数后删除原数组再重新根据难度创建新数组；void initMap()重新初始化 成员数组map，对map的初始化的思想为：创建一个新的长度为 difficulty \* images 的一维数组 b，每 int images 长度对该块内元素赋相同值，下一块元素值+1，之后调用void randomMapWithSource(int \* source)打乱所有元素值顺序，再将 b复制给map；

判断点击的两图片是否匹配的函数为 judge()函数，该函数每次被调用时重新初始化 辅助数组 int \*\* visited，对其他成员重新辅助后，根据传入的两个点对 一维数组MyVector \*dirct 排序，以判断优先从哪一个方向搜索，之后根据 MyVector \*dirct数组从四个方向依次调用bool DFS(MyPoint p, int direction, MyVector \*dirct) 进行搜索；bool DFS(MyPoint p, int direction, MyVector \*dirct)函数以深度优先的思想，每次以一个方向访问 map 的辅助数组 visited，对值为非零的点若为终点（两点匹配），则返回true，否则false，对值为0的点则继续以某个方向递归访问下一个点，并设其值为-1，对方向改变的拐点将其添加到 记录所访问路径的链表LinkedList<MyPoint> \*path中，当path 大小超过3即路线线段超过3条时，该路径不符合，函数返回false；

重排的实现的基础是对 map 的重新排列，通过调用void Game::rearrangeMap()实现，思想与 初始化 map的void Game::initMap()一样；

提示的实现通过调用bool Game::getHint()对地图 map 所有点遍历，调用 函数bool Game::judge(MyPoint start, MyPoint end)找出匹配的两个点。

代码：

**Game.h**

#ifndef GAME\_H\_

#define GAME\_H\_

#include <stdlib.h>

#include "LinkedList.h"

using namespace std;

enum direction {

UP = 0, DOWN = 1, LEFT = 2, RIGHT = 3

};

class MyVector {

public:

MyVector();

MyVector(int weight, int direction);

~MyVector();

inline MyVector operator=(MyVector v);

int weight = 0;

int dirct = 0;

};

inline MyVector::MyVector() : weight(0), dirct(0) {};

inline MyVector::MyVector(int weight, int direction) : weight(weight), dirct(direction) {}

inline MyVector::~MyVector() {}

inline MyVector MyVector::operator=(MyVector v)

{

this->dirct = v.dirct;

this->weight = v.weight;

return v;

}

class MyPoint {

public:

MyPoint();

MyPoint(int x, int y);

~MyPoint();

inline bool operator==(MyPoint &p);

int x = 0;

int y = 0;

};

inline MyPoint::MyPoint() {}

inline MyPoint::MyPoint(int x, int y)

{

this->x = x;

this->y = y;

}

inline MyPoint::~MyPoint() {}

inline bool MyPoint::operator==(MyPoint & p)

{

return (this->x == p.x && this->y == p.y);

}

class Game

{

public:

enum {

EASY = 8,

NORMAL = 10,

DIFFICULT = 12,

};

int \*\* map = nullptr;

LinkedList<MyPoint> \*path;

MyPoint start;

MyPoint end

int restCount = difficulty \* images;

int grade;

private:

int \*\*visited = nullptr;

MyVector \*dirct;

int difficulty;

int images;

bool hasFound = false;

bool prioritySwitch = false;

public:

Game();

~Game();

int getDifficulty();

void setDifficulty(int d);

void initMap();

void rearrangeMap();

bool getHint();

bool judge(MyPoint start, MyPoint end);

void printVisited();

private:

bool DFS(MyPoint p, int direction, MyVector \*dirct);

void sortDirection(MyPoint start, MyPoint end, MyVector \*dirct);

MyPoint getPointByDirct(MyPoint p, int dirct);

void randomMapWithSource(int \* source);

void reInitVisited();

void reCreateMap();

void deleteMap();

};

#endif

**Game.cpp**

#include "stdafx.h"

#include "Game.h"

#include "time.h"

#include <iostream>

#include <algorithm>

#include <thread>

using namespace std;

#define \_IsNIndex(x,y) ((x > images + 1) || (x < 0) || (y > difficulty + 1) || (y < 0))

Game::Game()

{

path = new LinkedList<MyPoint>();

dirct = new MyVector[4];

}

Game::~Game()

{

deleteMap();

delete[] dirct;

delete path;

}

void Game::setDifficulty(int d)

{

deleteMap();

difficulty = d;

switch (d)

{

case EASY:

images = 10;

break;

case NORMAL:

images = 12;

break;

case DIFFICULT:

images = 14;

break;

default:

break;

}

reCreateMap();

}

int Game::getDifficulty()

{

return difficulty;

}

/\*

\* 搜索全图，如果找到可消去的两点返回 true；否则 false

\* 如果 true，用 game.path.getFirst() and game.path.getLast() 获取两个点

\*/

bool Game::getHint()

{

for (int i = 1; i < difficulty + 1; i++) {

for (int j = 1; j < images + 1; j++) {

for (int m = i; m < difficulty + 1; m++) {

for (int n = j + 1; n < images + 1; n++) {

if (map[i][j] != map[m][n] || map[i][j] == 0 || map[m][n] == 0) {

continue;

}

if (judge(MyPoint(j, i), MyPoint(n, m))) {

cout << "( " << path->getFirst().x << "," << path->getFirst().y << ") ->" << "( " << path->getLast().x << "," << path->getLast().y << ")" << endl;

return true;

}

}

}

}

}

}

/\*

\* 重排 Map

\*/

void Game::rearrangeMap()

{

int \*s = new int[difficulty \* images];

int k = 0;

for (int i = 1; i < difficulty + 1; i++) {

for (int j = 1; j < images + 1; j++) {

s[k] = map[i][j];

k++;

}

}

randomMapWithSource(s);

delete[] s;

}

//after choose the difficulty, before start the game, initial the map

void Game::initMap()

{

int \*b = new int[difficulty \* images];

int i, j;

for (i = 0; i < difficulty \* images; i += images) {

for (j = 0; j < images; j++) {

b[i + j] = i / images + 1;

}

}

randomMapWithSource(b);

delete[] b;

}

/\*

\* if there is a path from vertex start to vertex end, return true and you can get the path by varOfGame.path

\*/

bool Game::judge(MyPoint start, MyPoint end)

{

//如果不同图片（对应数字不同） || 其中有空的图片（对应零）

if (map[start.y][start.x] != map[end.y][end.x] || map[start.y][start.x] == 0 || map[end.y][end.x] == 0) {

return false;

}

reInitVisited();

this->start = start;

this->end = end;

hasFound = false;

prioritySwitch = false;

path->clear();

sortDirection(start, end, dirct);

path->add(start);

//从 start 点开始，根据 方向优先队列 四个方向依次搜索路径

for (int i = 0; i < 4; i++) {

if (DFS(getPointByDirct(start, dirct[i].dirct), dirct[i].dirct, dirct)) {

int s = path->getSize();

//test

for (int i = 0; i < s; i++) {

cout << "(" << path->get(i).x << ", " << path->get(i).y << ") ->";

}

cout << endl << endl;

return true;

}

}

return false;

}

/\*

\* 深度优先找

\*/

bool Game::DFS(MyPoint p, int direction, MyVector \*dirct)

{

if (hasFound) { //has found the path

return true;

}

//the num of lines is greater than 3 or index is illegal

if (path->getSize() > 3 || \_IsNIndex(p.x, p.y)) {

return false;

}

//we find it

if (p == end) {

path->add(p);

return hasFound = true;

}

//此处有图片

if (visited[p.y][p.x] != 0) {

return false;

}

visited[p.y][p.x] = -1;

//printVisited();

//if (p.x == end.x || p.y == end.y) {

// //重新构造队列

// sortDirection(p, end);

//}

MyVector \*dirctArray = dirct;

//防止多余的搜索路径，重新排序方向队列

if (p.x == end.x || p.y == end.y) {

MyVector array2[4];

sortDirection(p, end, array2);

dirctArray = array2;

}

//根据 方向优先队列 依次深搜

for (int i = 0; i < 4; i++) {

if (direction == dirctArray[i].dirct) {

DFS(getPointByDirct(p, dirctArray[i].dirct), dirctArray[i].dirct, dirctArray);

}

else {

path->add(p); //add 拐点 to path

if (!DFS(getPointByDirct(p, dirctArray[i].dirct), dirctArray[i].dirct, dirctArray)) {

path->removeLast();

}

}

visited[p.y][p.x] = 0;

if (hasFound) { //has found the path

return true;

}

}

return false;

}

/\*

\* 以最适宜的 方向 构造一个优先队列

\*/

void Game::sortDirection(MyPoint start, MyPoint end, MyVector \*dirct)

{

//initial array

dirct[0].weight = dirct[1].weight = dirct[2].weight = dirct[3].weight = 0;

dirct[0].dirct = UP; dirct[1].dirct = DOWN; dirct[2].dirct = LEFT; dirct[3].dirct = RIGHT;

int dx = end.x - start.x;

int dy = end.y - start.y;

if (abs(dx) < abs(dy)) {

dirct[0].weight = dirct[1].weight += 1;

}

else if (abs(dx) > abs(dy)) {

dirct[2].weight = dirct[3].weight += 1;

}

else {

dirct[0].weight = dirct[1].weight = dirct[2].weight = dirct[3].weight += 1;

}

if (dx > 0) {

dirct[3].weight++;

dirct[2].weight -= 2;

}

else if (dx < 0) {

dirct[2].weight++;

dirct[3].weight -= 2;

}

if (dy > 0) {

dirct[1].weight++;

dirct[0].weight -= 2;

}

else if (dy < 0) {

dirct[0].weight++;

dirct[1].weight -= 2;

}

//insertion-sort

for (int i = 1; i < 4; i++) {

MyVector key;

key = dirct[i];

int j = i - 1;

for (; j >= 0 && key.weight > dirct[j].weight; j--) {

dirct[j + 1] = dirct[j];

}

dirct[j + 1] = key;

}

}

/\*

\* get a point around p by a direction

\*/

MyPoint Game::getPointByDirct(MyPoint p, int d)

{

switch (d) {

case 0:

return MyPoint(p.x, p.y - 1);

break;

case 1:

return MyPoint(p.x, p.y + 1);

break;

case 2:

return MyPoint(p.x - 1, p.y);

break;

case 3:

return MyPoint(p.x + 1, p.y);

break;

default:

return MyPoint();

break;

}

}

/\*

\* 重新排列并随机化 source 数组并 copy 到 map

\*/

void Game::randomMapWithSource(int \* source)

{

//random\_shuffle()用来对一个元素序列进行重新排序（随机的）andom\_shuffle()有两个参数，第一个参数是指向序列首元素的迭代器，第二个参数则指向序列最后一个元素的下一个位置

srand((unsigned)time(NULL));

std::random\_shuffle(source, source + difficulty \* images);

int k = 0, i, j;

//initial map = {0}

for (i = 0; i < difficulty + 2; i++) {

for (j = 0; j < images + 2; j++) {

map[i][j] = 0;

}

}

//map[1 to difficulty+1][1 to difficulty+1] = source[0 to difficulty][0 to difficulty]

for (i = 1; i < difficulty + 1; i++) {

for (j = 1; j < images + 1; j++) {

map[i][j] = source[k];

k++;

}

}

}

/\*

\* 重新初始化 visited 数组

\*/

void Game::reInitVisited()

{

for (int i = 0; i < difficulty + 2; i++) {

for (int j = 0; j < images + 2; j++) {

visited[i][j] = map[i][j];

}

}

}

/\*

\* 重新创建 map

\*/

void Game::reCreateMap()

{

map = new int\*[difficulty + 2];

for (int i = 0; i < difficulty + 2; i++) {

map[i] = new int[(images + 2)];

}

visited = new int\*[difficulty + 2];

for (int i = 0; i < difficulty + 2; i++) {

visited[i] = new int[(images + 2)];

}

}

void Game::deleteMap()

{

if (map != nullptr) {

for (int i = 0; i < difficulty + 2; i++) {

delete[] map[i];

}

}

if (visited != nullptr) {

for (int i = 0; i < difficulty + 2; i++) {

delete[] visited[i];

}

}

}

void Game::printVisited()

{

for (int i = 0; i < difficulty + 2; i++) {

for (int j = 0; j < images + 2; j++) {

if (visited[i][j] > 9 || visited[i][j] < 0) {

cout << visited[i][j] << " ";

}

else {

cout << visited[i][j] << " ";

}

}

cout << endl;

}

cout << endl;

}

## 2.3玩家获胜排行榜

每次游戏结束玩家可以选择保存或不保存游戏成绩。若保存游戏成绩，则成绩被写入phb.txt文件。当有人查看排行榜时，系统读取phb.txt文件中保存的信息，并且按分数高低进行排列，从高到低依次显示。本功能有六个函数来实现，其中5个用于文件信息的筛选和排序。

//获取最大分数

int getAtMAX(vector<int>\* vi) {

int sc = 0;

int pos = 0;

for (size\_t i = 0; i < vi->size(); i++)

{

if (sc < vi->at(i))

{

sc = vi->at(i);

pos = i;

}

}

//cout << "max score" << sc << endl;

//tihuan

vector<int>::iterator vit;

vit = find(vi->begin(), vi->end(), sc);

\*vit = 0; //newValue是你要赋的新值，把vier看成指针就好

return pos;

}

//分割字符串

int SplitString(const CString str, char split, CStringArray &strArray)

{

strArray.RemoveAll();

CString strTemp = str;

int iIndex = 0;

while (1)

{

iIndex = strTemp.Find(split);

if(iIndex >= 0)

{

strArray.Add(strTemp.Left(iIndex));

strTemp = strTemp.Right(strTemp.GetLength()-iIndex-1);

}

else

{

break;

}

}

strArray.Add(strTemp);

return strArray.GetSize();

}

//获得字符串中的score信息

CString getScore(CString str) {

CStringArray strArray;

int d = SplitString(str, '!', strArray);

//cout <<"有str个数："<< d << endl;

return strArray.GetAt(2);

}

//获得分数vector

vector<int> getScoreArray(CStringArray\* csa) {

vector<int> scos;

int sco = 0;

string str;

CString strtemp;

//cout << "截取字符串" << endl;

for (size\_t i = 0; i < csa->GetSize(); i++)

{

strtemp = getScore(csa->GetAt(i));//截取字符串

//AfxMessageBox(strtemp);

CString st = strtemp.Mid(7, strtemp.GetLength() - 7);

sco = \_ttoi(st);

//cout << i << "->" << sco << endl;

scos.push\_back(sco);

}

//ok

return scos;

}

//分数排序

void sortOnScores(CStringArray\* csa) {

CStringArray temp;

int pos = 0;

int len = csa->GetSize();

vector<int> scos = getScoreArray(csa);

//ok

//cout << csa->GetSize() << endl;

for (size\_t i = 0; i < len; i++)

{

pos = getAtMAX(&scos);

//cout <<"位置"<<i<<"---"<< pos << endl;

temp.Add(csa->GetAt(pos));

}

csa->Copy(temp);

/\*string str;

for (size\_t i = 0; i < csa->GetSize(); i++)

{

str = CT2A(csa->GetAt(i));

cout << "str" << "->" << str.data() << endl;

}

\*/

}

//排行榜具体文件操作以及读写

void LOCAL\_LIST::setText() {

CEdit\* pBoxOne;

pBoxOne = (CEdit\*)GetDlgItem(IDC\_EDIT1);

CString strFileName = \_T("localphb.txt");

CString strValue = \_T("");

CStringArray cstrArray;

if (!PathFileExists(strFileName))

{

return exit(0);

}

CStdioFile file;

if (!file.Open(strFileName, CFile::modeRead))

{

return exit(0);

}

while (file.ReadString(strValue))

{

cstrArray.Add(strValue);

}

file.Close();

//按分数排序

sortOnScores(&cstrArray);

CString strshow = \_T("");

for (int i = 0; i < cstrArray.GetSize(); i++) {

strshow += cstrArray.GetAt(i);

strshow += \_T("\r\n");

}

strshow.Replace(\_T("!!"), \_T("\t"));//替换文件中字符串

pBoxOne->SetWindowText(strshow);

}

## 2.4 声音控制

本游戏具有背景播放以及点击音效以及相关控制的功能，上述功能的实现封装在 MusicPalyer类中，class Mci 底层实际上使用 mciSendStringA 函数。当要播放音频时，声明一个 MusicPlayer 实例后，调用其成员函数 bool MusicPlayer::load(const std::string& \_filename) 加载音频文件后，再调用bool MusicPlayer::playAndRepeat(bool repeated, int start\_ms, int end\_ms)播放音频文件；

代码：

MusicPlsyer.h

#ifndef MUSICPLAYER\_H

#define MUSICPLAYER\_H

#include<string>

#include<Windows.h>

#pragma comment(lib,"Winmm.lib")//放音乐的程序

typedef int(\_\_stdcall\* w32mci)(const char\*, char\*, int, int);

typedef int(\_\_stdcall \* w32mcierror)(int, char\*, int);

class Mci

{

private:

HINSTANCE hins;

w32mci wmci;

w32mcierror wmcierror;

public:

Mci();

~Mci();

char buf[256];

bool send(std::string command);//error return false

};

/\*

\* Actually, we can call it soundPlayer.

\* you can create a instance at first and use musicPlayer.load(filename)

\* to open a audio file and use musicPlayer.player() to play it.

\*/

class MusicPlayer

{

private:

Mci mci;

std::string filename;

std::string alias;

int length\_ms;

public:

MusicPlayer();

~MusicPlayer();

bool load(const std::string& \_filename);

bool playAndRepeat(bool repeated = 0, int start\_ms = 0, int end\_ms = -1);

bool stop();

bool pause();

bool unpause();

int milliseconds();

};

#endif

MusicPlayer.cpp

#include "stdafx.h"

#include "MusicPlayer.h"

#include <mciapi.h>

#include<iostream>

#include<random>

#include<time.h>

Mci::Mci()

{

HINSTANCE hins = LoadLibraryA("winmm.dll");

wmci = (w32mci)GetProcAddress(hins, "mciSendStringA");

wmcierror = (w32mcierror)GetProcAddress(hins, "mciGetErrorStringA");

}

Mci::~Mci()

{

FreeLibrary(hins);

}

bool Mci::send(std::string command)

{

int errcode = wmci(command.c\_str(), buf, 254, 0);

if (errcode)

{

wmcierror(errcode, buf, 254);

return false;

}

return true;

}

MusicPlayer::MusicPlayer()

{

//do nothing

}

MusicPlayer::~MusicPlayer()

{

std::string cmd;

cmd = "close " + alias;

mci.send(cmd);

}

bool MusicPlayer::load(const std::string& \_filename)

{

filename = \_filename;

for (unsigned int i = 0; i < filename.length(); i++)

{

if (filename[i] == '/')

filename[i] = '\\';

}

alias = "mp3\_";

srand(time(NULL));

char randstr[6];

\_itoa(rand() % 65536, randstr, 10);

alias.append(randstr);

std::string cmd;

cmd = "open " + filename + " alias " + alias;

if (mci.send(cmd) == false) {

return false;

}

cmd = "set " + alias + " time format milliseconds";

if (mci.send(cmd) == false) {

return false;

}

cmd = "status " + alias + " length";

if (mci.send(cmd) == false) {

return false;

}

length\_ms = atoi(mci.buf);

return true;

}

bool MusicPlayer::playAndRepeat(bool repeated, int start\_ms, int end\_ms)

{

if (end\_ms == -1) end\_ms = length\_ms;

std::string cmd;

char start\_str[16], end\_str[16];

\_itoa(start\_ms, start\_str, 10);

\_itoa(end\_ms, end\_str, 10);

cmd = "play " + alias + " from ";

cmd.append(start\_str);

cmd.append(" to ");

cmd.append(end\_str);

if (repeated) {

cmd.append(" repeat");

}

return mci.send(cmd);

}

bool MusicPlayer::stop()

{

std::string cmd;

cmd = "stop " + alias;

if (mci.send(cmd) == false) {

return false;

}

cmd = "seek " + alias + " to start";

if (mci.send(cmd) == false) {

return false;

}

return true;

}

bool MusicPlayer::pause()

{

std::string cmd;

cmd = "pause " + alias;

if (mci.send(cmd) == false) {

return false;

}

return true;

}

bool MusicPlayer::unpause()

{

std::string cmd;

cmd = "resume " + alias;

if (mci.send(cmd) == false) {

return false;

}

return true;

}

int MusicPlayer::milliseconds()

{

return length\_ms;

}

## 2.5 人机对战

棋盘分为两个部分，分别为玩家的棋盘以及系统的棋盘，在规定的时间内，玩家和系统进行对战，如果玩家获胜，则获胜，否则失败。在游戏的过程中，玩家可以使用提示功能以及重拍功能，来降低游戏难度，游戏获胜，排名不计入排行榜，如果觉得对战不下去，可以选择认输功能来认输。

相关实现代码：

if (game\_NET->getHint())

{

MyPoint p1;

MyPoint p2;

*CWindowDC* dc2(this); // 用于绘制的设备上下文

dc2.*SetROP2*(*R2\_NOT*);//那么第一次画，出现线条，再画，就抹去了。因为他对像素点做的异或运算（同样像素值异或清0了）

*CPen* pen(*PS\_SOLID*, 5, *RGB*(255, 0, 0));

dc2.*SelectObject*(pen);

*SetTimer*(1, 100, 0);

pt1.x = (game\_NET->path->getFirst().x - 1) \* 50 + 770;

pt1.y = (game\_NET->path->getFirst().y) \* 50 + 155;

pt2.x = (game\_NET->path->getLast().x - 1) \* 50 + 770;

pt2.y = (game\_NET->path->getLast().y) \* 50 + 155;

for (int i = 0; i < (game\_NET->getDifficulty())\*(game\_NET->getDifficulty() + 2); i++)

{

CBUTTON \*btn = (CBUTTON \*)m\_btnGroup\_net.*GetAt*(i);

if (game\_NET->path->getFirst().y==btn->location.*x*&&game\_NET->path->getFirst().x == btn->location.*y*)

{

btn->*ShowWindow*(false);

btn->ID = 0;

game\_NET->map[btn->location.*x*][btn->location.*y*] = 0;

}else if (game\_NET->path->getLast().y == btn->location.*x*&&game\_NET->path->getLast().x == btn->location.*y*)

{

btn->*ShowWindow*(false);

btn->ID = 0;

game\_NET->map[btn->location.*x*][btn->location.*y*] = 0;

}

}

int i = game\_NET->path->getSize();

if (i == 2)

{

dc2.*MoveTo*(pt1);

dc2.*LineTo*(pt2);

*Sleep*(300);

//等待0.1秒消除线条

//没有拐弯的情况

dc2.*MoveTo*(pt1);

dc2.*LineTo*(pt2);

}

else if (i == 3)

{

//一个拐弯

p1 = game\_NET->path->get(1);

CBUTTON::conner1.*x* = p1.y;

CBUTTON::conner1.*y* = p1.x;

pt3.x = (CBUTTON::conner1.*y* - 1) \* 50 + 770;

pt3.y = (CBUTTON::conner1.*x*) \* 50 + 155;

dc2.*MoveTo*(pt1);

dc2.*LineTo*(pt3);

dc2.*LineTo*(pt2);

*Sleep*(300);

//等待0.1秒消除线条

//没有拐弯的情况

dc2.*MoveTo*(pt1);

dc2.*LineTo*(pt3);

dc2.*LineTo*(pt2);

}

else if (i == 4)

{

//两个拐弯

p1 = game\_NET->path->get(1);

CBUTTON::conner1.*x* = p1.y;

CBUTTON::conner1.*y* = p1.x;

p2 = game\_NET->path->get(2);

CBUTTON::conner2.*x* = p2.y;

CBUTTON::conner2.*y* = p2.x;

pt3.x = (CBUTTON::conner1.*y* - 1) \* 50 + 770;

pt3.y = (CBUTTON::conner1.*x*) \* 50 + 155;

pt4.x = (CBUTTON::conner2.*y* - 1) \* 50 + 770;

pt4.y = (CBUTTON::conner2.*x*) \* 50 + 155;

dc2.*MoveTo*(pt1);

dc2.*LineTo*(pt3);

dc2.*LineTo*(pt4);

dc2.*LineTo*(pt2);

*Sleep*(300);

//等待0.1秒消除线条

//没有拐弯的情况

dc2.*MoveTo*(pt1);

dc2.*LineTo*(pt3);

dc2.*LineTo*(pt4);

dc2.*LineTo*(pt2);

}

if (IsFinished\_())

{

*KillTimer*(3);

*KillTimer*(2);

*MessageBox*(*\_T*("你输了！"), *\_T*("警告"));

MENU \*menu = new MENU();

menu->*Create*(IDD\_MENU);

menu->*ShowWindow*(*SW\_SHOW*);

*CDialogEx*::*OnCancel*();

}

}

else

{

*MessageBox*(*\_T*("无法消除，请尝试重排！"), *\_T*("警告"));

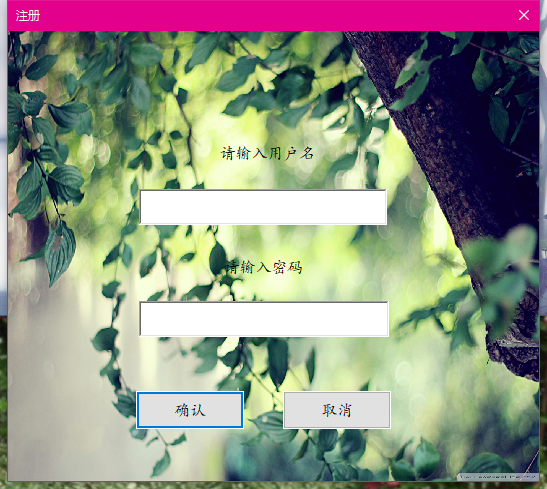
}

## 2.6界面绘制

本游戏有众多界面组成，以下一一介绍：  
**登陆界面：**



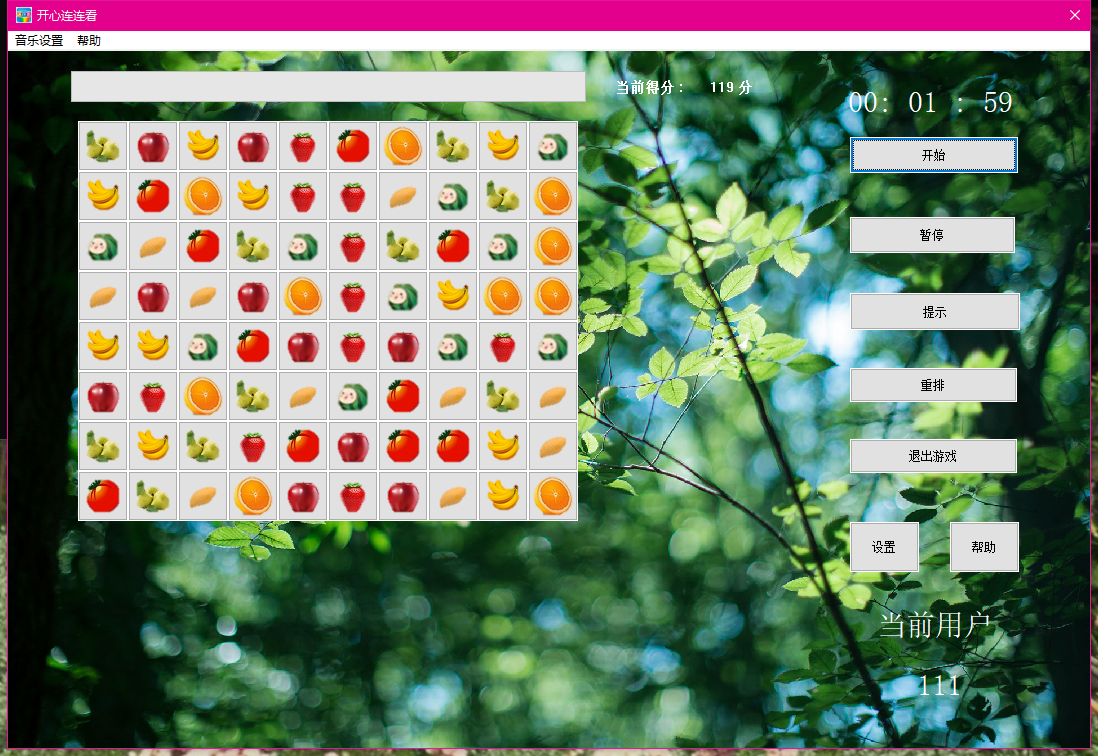
**注册界面：**



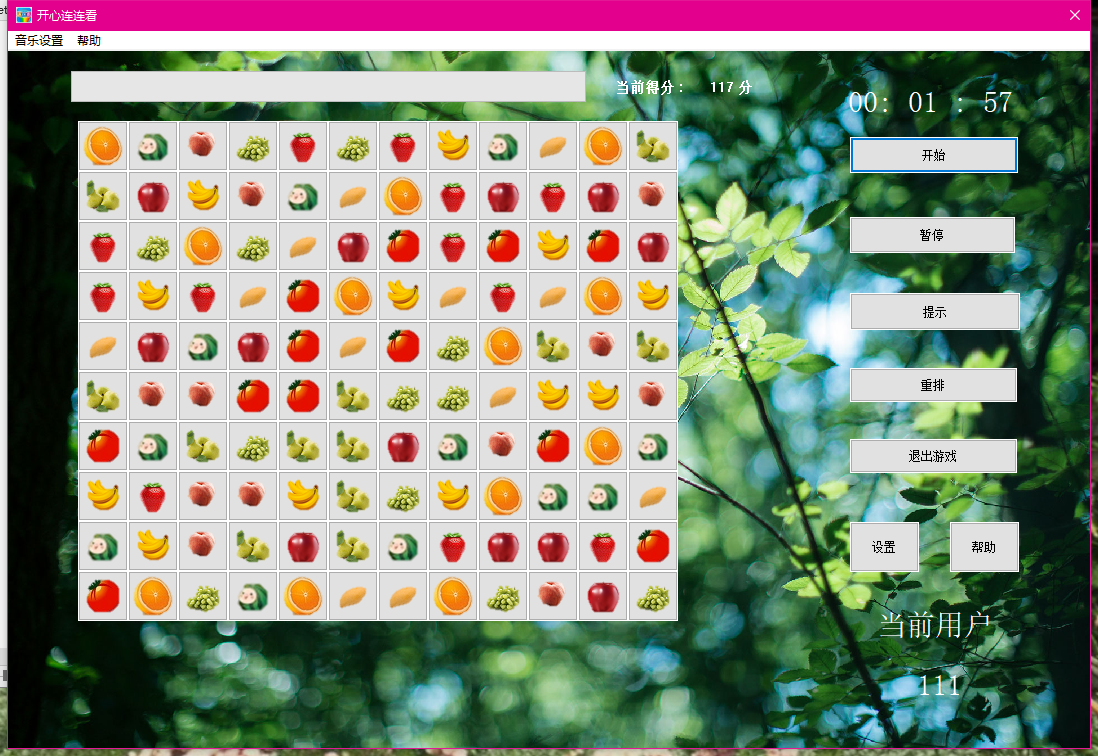
**菜单界面：**



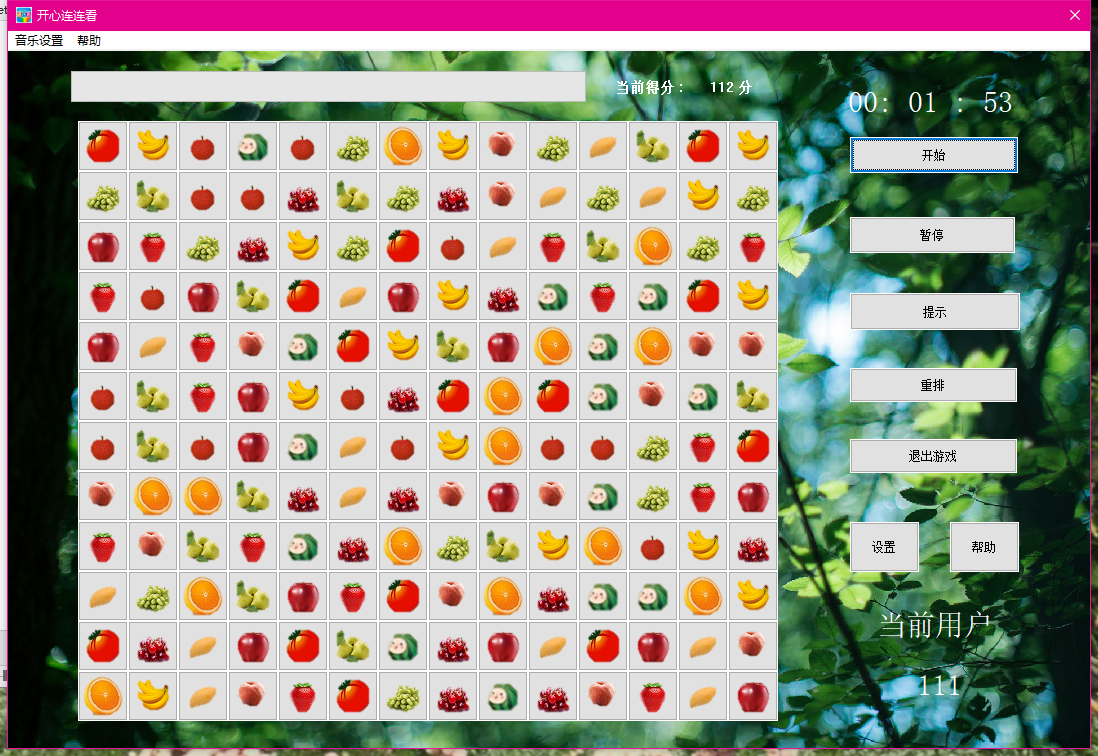
**游戏界面：基本模式**



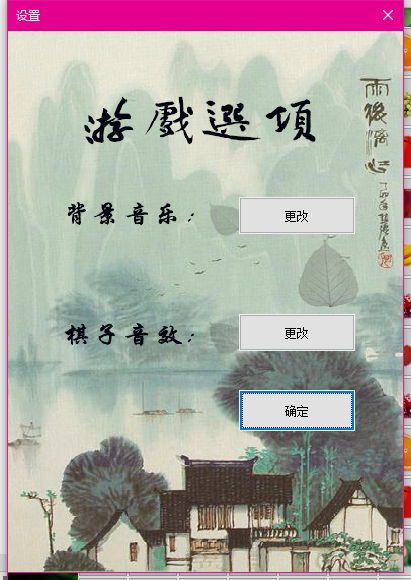
**游戏界面：简单模式**



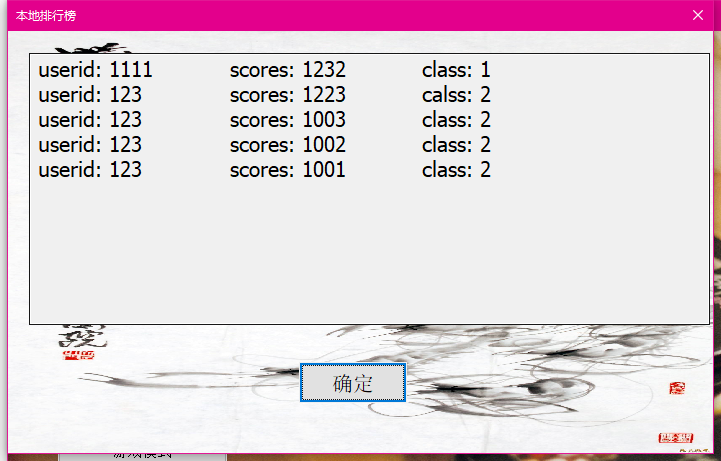
**游戏界面：游戏模式**



**设置界面：**



**排行榜：**



**人机对战:**



**帮助界面：**

