贪吃蛇游戏

软件分析与设计说明书

**目录**

[1 系统概述 3](#_Toc2835)

[1.1 游戏规则 3](#_Toc9382)

[1.2 概要设计 3](#_Toc31674)

[1.2.1概述 3](#_Toc1094)

[1.2.2功能 4](#_Toc5556)

[1.3 环境要求 5](#_Toc8937)

[1.3.1硬件环境 5](#_Toc11970)

[1.3.2软件要求 5](#_Toc18685)

[1.3.3网络环境 6](#_Toc30821)

[1.3.4安全和隐私 6](#_Toc2445)

[1.3.5其他环境要求 6](#_Toc27366)

[2 模块设计 6](#_Toc28573)

[2.1 注册模块 6](#_Toc29095)

[2.2 登录模块 7](#_Toc23465)

[2.3 随机出现食物模块 10](#_Toc11063)

[2.4 移动模块 11](#_Toc14849)

[2.5 结束游戏模块 16](#_Toc25269)

[2.6 游戏日志模块 17](#_Toc16657)

[3 测试运行 18](#_Toc21889)

[3.1 注册登录测试 18](#_Toc7896)

[3.2 显示最终得分 19](#_Toc12114)

[3.3 显示游戏日志 19](#_Toc30485)

[4 总结 20](#_Toc14379)

# 1 系统概述

## 1.1 游戏规则

贪吃蛇游戏的目标是让玩家控制一条蛇在游戏界面上移动，并尽可能地吃到食物，以增加蛇的长度。游戏开始时，蛇的初始长度为一个单位，食物随机出现在游戏界面上的一个位置。

玩家通过键盘方向键或触摸屏幕上的滑动手势来控制蛇的移动方向。蛇每次移动一个单位距离，可以向上、向下、向左或向右移动。蛇的移动速度通常是固定的，但可能会根据游戏难度或特殊设置而改变。

蛇不能穿过自己的身体或游戏界面的边缘，否则游戏将结束。当蛇的头部接触到游戏界面的边缘或者与自己的身体相撞时，游戏结束，玩家得分将会显示在游戏结束画面上。

食物会在游戏界面上随机位置生成，但不会出现在蛇的当前位置或其身体的任何部分上。当蛇的头部接触到食物时，食物会被吃掉，蛇的长度会增加一个单位，并且玩家得到一定数量的分数。食物被吃掉后，会在游戏界面上的另一个随机位置重新生成。

玩家可以在游戏进行过程中随时暂停游戏，并在暂停菜单中查看当前得分和游戏设置。玩家还可以在游戏结束后选择重新开始游戏或返回主菜单。确保玩家理解并遵守游戏规则是保证游戏进行顺利的关键。

## 1.2 概要设计

### 1.2.1 概述

1）游戏名称：贪吃蛇

2）游戏类型：经典单机游戏

3）游戏目标：控制蛇在游戏界面上移动，尽可能多地吃到食物，增加蛇的长度，直到游戏结束。

4）游戏机制：

玩家通过键盘方向键或触摸屏控制蛇的移动方向，蛇每次移动一个单位距离；蛇的长度随着吃到食物而增加，蛇的移动速度可能会随着游戏进行而增加；蛇不能穿过自身身体或游戏界面的边缘，否则游戏结束。食物在游戏界面上随机位置生成，吃到食物后，蛇的长度增加，玩家得分增加，游戏随着时间的推移而变得更加困难，可能会减少食物出现的频率。

游戏特色：

5）随机生成食物：食物会在游戏界面上的随机位置生成，增加游戏的变化性和挑战性。

6）游戏界面设计：

简洁的二维图形界面，包括蛇、食物和背景，游戏界面清晰明了，方便玩家进行游戏操作和观察游戏状态。

7）用户交互：

支持键盘操作，玩家可以通过方向键控制蛇的移动，可在游戏进行中暂停、继续游戏或返回主菜单重新开始，方便玩家管理游戏进程和设置。

### 1.2.2功能

1.2.2.1游戏核心功能

1）控制蛇移动：玩家通过键盘或其他输入设备控制蛇的移动方向，使其吃到食物并避免碰撞到墙壁或自身身体。

2）食物生成：在游戏地图上随机生成食物，蛇吃到食物后长度增加，玩家得分。

3）碰撞检测：实时检测蛇头与墙壁、蛇身的碰撞情况，以及蛇头与食物的接触情况。

4）计分系统：记录玩家的得分，并在游戏界面显示。

1.2.2.2 用户界面功能

1）游戏开始界面：显示游戏标题、开始游戏按钮等，提供游戏开始和退出选项。

2）游戏进行界面：显示游戏地图、蛇、食物以及当前得分等游戏信息，提供游戏暂停和重新开始功能。

3）游戏结束界面：在游戏结束时显示玩家得分和游戏结束提示，提供重新开始和返回主菜单等选项。

1.2.2.3 非功能需求

1）性能需求：游戏应具有良好的性能，流畅运行且不卡顿。

2）可靠性需求：游戏应具有较高的稳定性和可靠性，不易崩溃或出现异常情况。

3）易用性需求：界面友好，操作简单，易于上手。

4）安全性需求：游戏应不含恶意代码，确保玩家账号和个人信息安全。

5）兼容性需求：支持不同操作系统和设备，如Windows、macOS、Linux，以及PC等平台。

## 1.3 环境要求

贪吃蛇游戏是一款经典的电脑游戏，为了确保游戏在各种系统和设备上顺利运行，环境要求部分需要明确描述游戏所需的硬件、软件和网络环境。

### 1.3.1硬件环境

1.3.1.1计算机/设备

最低配置：处理器要求至少1 GHz、内存至少256 MB。

推荐配置：处理器2 GHz或以上，内存1 GB或以上。

1.3.1.2图形和显示

最低配置：集成显卡，支持至少800x600分辨率。

推荐配置：独立显卡，支持1920x1080分辨率或更高。

1.3.1.3存储

最低配置：至少50 MB的可用硬盘空间。

推荐配置：100 MB或更多可用硬盘空间。

### 1.3.2软件要求

1.3.2.1操作系统

最低要求：Windows 7、macOS 10.10、Linux 发行版（如Ubuntu 16.04或更高版本）。

推荐要求：Windows 10、macOS 11、最新版本的Linux发行版。

1.3.2.2 开发工具和框架

开发工具：DEV-++，CodeBlocks等可编译C++的开发软件。

### 1.3.3网络环境

贪吃蛇游戏基本上是单机游戏，不需要网络连接。

### 1.3.4安全和隐私

1.3.4.1 安全性

确保游戏没有恶意软件和病毒。

1.3.4.2 隐私

支持注册用户，在使用时输入所注册的用户名和密码登陆游戏。

### 1.3.5其他环境要求

输入设备：键盘。

# 2 模块设计

## 2.1 注册模块

用户在首次进入游戏时需要注册用户名并设置密码。添加注册函数并将注册的用户信息存储在文件中。在注册用户时通过在注册函数registerUser()中调用 saveUsers()实现用户信息的保存。

void registerUser()

{

if (numUsers >= MAX\_USERS)

{

printf("无法继续注册新用户，达到最大用户数限制\n");

return;

}

users[numUsers].id = users[numUsers-1].id+1;

printf("请输入用户名: ");

scanf("%s", users[numUsers].username);

strcpy(storeUsername, users[numUsers].username);

printf("请输入密码: ");

scanf("%s", users[numUsers].password);

numUsers++;

saveUsers();

printf("注册成功！\n");

}

void saveUsers()

{

FILE \*file = fopen("users.txt", "w");

for (int i = 0; i < numUsers; i++)

{

fprintf(file, "%d %s %s\n", users[i].id, users[i].username, users[i].password);

}

fclose(file);

}

## 2.2 登录模块

用户若已经注册了账号则直接进行登录若没有注册账号则需要先进行注册，为了便于用户进行选择，对开始时的游戏界面进行优化，用户可以通过选择数字1,2或3进行注册，登录或退出游戏。

void welcometogame()

{

int loggedIn = 0;

while (!loggedIn)

{

char choice;

printf("欢迎来到贪食蛇游戏！\n");

printf("请选择操作：\n");

printf("1. 注册\n");

printf("2. 登录\n");

printf("3. 退出游戏\n"); // 添加退出游戏选项

scanf(" %c", &choice);

if (choice == '1')

{

registerUser();

}

else if (choice == '2')

{

if (verifyUser())

{

loggedIn = 1;

printf("登录成功！\n");

char playChoice;

printf("是否进入游戏？(Y/N): ");

scanf(" %c", &playChoice);

if (playChoice == 'Y' || playChoice == 'y')

{

printf("进入游戏...\n");

system("mode con cols=100 lines=30");

}

else

{

printf("感谢使用，再见！\n");

break; // 退出游戏

}

}

else

{

printf("登录失败，请重试。\n");

}

}

else if (choice == '3')

{

printf("感谢使用，再见！\n");

exit(0);

}

else

{

printf("无效的选择，请重新输入。\n");

}

}

system("cls");

Pos(40, 12);

printf("欢迎来到贪食蛇游戏！");

Pos(40, 25);

system("pause");

system("cls");

Pos(25, 12);

printf("用↑.↓.←.→分别控制蛇的移动， F1 为加速，2 为减速\n");

Pos(25, 13);

printf("加速将能得到更高的分数。\n");

system("pause");

system("cls");

}

添加上述函数中进行调用的登录函数verifyUser()验证用户输入的密码是否正确，若登录成功则可以选择进入游戏若登陆失败则需要重新进行登录。

int verifyUser()

{

char inputUsername[USERNAME\_LENGTH];

char inputPassword[PASSWORD\_LENGTH];

printf("请输入用户名: ");

scanf("%s", inputUsername);

printf("请输入密码: ");

scanf("%s", inputPassword);

for (int i = 0; i < numUsers; i++)

{

if (strcmp(users[i].username, inputUsername) == 0 && strcmp(users[i].password, inputPassword) == 0)

{

return 1; // 验证通过

}

}

return 0; // 验证失败

}

## 2.3 随机出现食物模块

createfood() 函数用于随机生成食物，并确保食物不会与蛇的身体重合，也不会出现在游戏区域的边界上。首先声明一些变量，包括食物的坐标 food\_x 和 food\_y，以及一个指向食物的指针 food\_1。

通过调用 srand((unsigned)time(NULL))来初始化随机数生成器，确保每次运行时都能产生不同的随机数。使用 do-while 循环来生成随机的食物坐标，同时检查生成的食物坐标是否与蛇的身体重合，以及是否出现在游戏区域的边界上。如果条件不满足，则继续生成新的随机坐标，直到满足条件为止。

一旦找到合适的食物坐标，就动态分配内存来创建一个新的食物节点 food\_1，并将其坐标设置为生成的随机坐标。然后将食物节点的坐标打印到游戏界面上，并将食物指针 food 指向新创建的食物节点。最后，打印一个代表食物的图标"■"表示食物的位置

void createfood()//随机出现食物

{

int food\_x;

int food\_y;

snake\* food\_1;

srand((unsigned)time(NULL));

do

{

food\_x = rand() % 54 + 2; // 生成 x 坐标

food\_y = rand() % 24 + 1; // 生成 y 坐标

q = head;

while (q != NULL)

{

if (q->x == food\_x && q->y == food\_y) // 判断食物是否与蛇身重合

{

break;

}

q = q->next;

}

}

while (q != NULL || food\_x % 2 != 0 || food\_x == 0 || food\_x == 56 || food\_y == 0 || food\_y == 26);

food\_1 = (snake\*)malloc(sizeof(snake));

food\_1->x = food\_x;

food\_1->y = food\_y;

Pos(food\_1->x, food\_1->y);

food = food\_1;

printf("■");

}

## 2.4 移动模块

根据蛇头的当前状态（上、下、左、右），分别更新蛇头的位置，并检查是否吃到了食物。如果吃到了食物，就将新的蛇头加入到蛇身链表的头部，并打印出新的蛇身。如果没有吃到食物，则将新的蛇头加入到蛇身链表的头部，同时删除蛇尾部分，使得蛇看起来像是在移动。最后，检查是否蛇咬到了自己，如果咬到了自己，则游戏结束。

void snakemove()//蛇前进,上U,下D,左L,右R

{

snake\* nexthead;

cantcrosswall();

nexthead = (snake\*)malloc(sizeof(snake));

if (status == U)

{

nexthead->x = head->x;

nexthead->y = head->y - 1;

if (nexthead->x == food->x && nexthead->y == food->y)//如果下一个有食物

{

nexthead->next = head;

head = nexthead;

q = head;

while (q != NULL)

{

Pos(q->x, q->y);

printf("■");

q = q->next;

}

score = score + add;

createfood();

}

else //如果没有食物

{

nexthead->next = head;

head = nexthead;

q = head;

while (q->next->next != NULL)

{

Pos(q->x, q->y);

printf("■");

q = q->next;

}

Pos(q->next->x, q->next->y);

printf(" ");

free(q->next);

q->next = NULL;

}

}

if (status == D)

{

nexthead->x = head->x;

nexthead->y = head->y + 1;

if (nexthead->x == food->x && nexthead->y == food->y) //有食物

{

nexthead->next = head;

head = nexthead;

q = head;

while (q != NULL)

{

Pos(q->x, q->y);

printf("■");

q = q->next;

}

score = score + add;

createfood();

}

else //没有食物

{

nexthead->next = head;

head = nexthead;

q = head;

while (q->next->next != NULL)

{

Pos(q->x, q->y);

printf("■");

q = q->next;

}

Pos(q->next->x, q->next->y);

printf(" ");

free(q->next);

q->next = NULL;

}

}

if (status == L)

{

nexthead->x = head->x - 2;

nexthead->y = head->y;

if (nexthead->x == food->x && nexthead->y == food->y)//有食物

{

nexthead->next = head;

head = nexthead;

q = head;

while (q != NULL)

{

Pos(q->x, q->y);

printf("■");

q = q->next;

}

score = score + add;

createfood();

}

else //没有食物

{

nexthead->next = head;

head = nexthead;

q = head;

while (q->next->next != NULL)

{

Pos(q->x, q->y);

printf("■");

q = q->next;

}

Pos(q->next->x, q->next->y);

printf(" ");

free(q->next);

q->next = NULL;

}

}

if (status == R)

{

nexthead->x = head->x + 2;

nexthead->y = head->y;

if (nexthead->x == food->x && nexthead->y == food->y)//有食物

{

nexthead->next = head;

head = nexthead;

q = head;

while (q != NULL)

{

Pos(q->x, q->y);

printf("■");

q = q->next;

}

score = score + add;

createfood();

}

else //没有食物

{

nexthead->next = head;

head = nexthead;

q = head;

while (q->next->next != NULL)

{

Pos(q->x, q->y);

printf("■");

q = q->next;

}

Pos(q->next->x, q->next->y);

printf(" ");

free(q->next);

q->next = NULL;

}

}

if (biteself() == 1) //判断是否会咬到自己

{

endgamestatus = 2;

endgame();

}

}

## 2.5 结束游戏模块

游戏的结束处理逻辑。在结束游戏后进行了一些操作：显示得分、询问是否继续游戏、以及记录日志。使用 GetLocalTime(&time)获取当前时间。int duration = sleeptime获取游戏持续时间。recordGameLog(storeUsername, duration, score)调用函数记录游戏日志，包括玩家名称、游戏持续时间和得分。游戏结束主要分为三种情况：

endgamestatus == 1: 表示玩家撞到了墙。输出相关信息。

endgamestatus == 2: 表示玩家咬到了自己。输出相关信息。

endgamestatus == 3: 表示玩家自行结束游戏。输出相关信息

游戏结束后用户可选择是否继续游戏若继续游戏则回到登录页面重新登录若否则退出游戏。

void endgame()

{

system("cls");

Pos(24, 12);

if (endgamestatus == 1)

{

printf("对不起，您撞到墙了。游戏结束!");

}

else if (endgamestatus == 2)

{

printf("对不起，您咬到自己了。游戏结束!");

}

else if (endgamestatus == 3)

{

printf("您已经结束了游戏。");

}

Pos(24, 13);

printf("您的得分是 %d\n", score);

char playChoice;

printf("是否继续游戏？(Y/N): ");

scanf(" %c", &playChoice);

if (playChoice == 'Y' || playChoice == 'y')

{

system("cls");

welcometogame(); // 返回到游戏开始界面

}

else

{

printf("感谢参与游戏，再见！\n");

SYSTEMTIME time;

GetLocalTime(&time);

int duration = sleeptime; // 游戏持续时间

recordGameLog(storeUsername, duration, score); // 记录游戏日志

}

}

## 2.6 游戏日志模块

void recordGameLog用于记录游戏日志。它接受三个参数：玩家用户名、游戏持续时间和得分。创建一个名为 newLog 的 GameLog 结构体实例，用于存储新的游戏日志。将 newLog 结构体添加到名为 logs 的数组中遍历存储游戏日志的数组。将新的游戏日志以指定格式写入到文本文件中。格式包括玩家用户名、游戏持续时间和得分。

void recordGameLog(const char\* username, double duration, int score)

{

GameLog newLog;

strcpy(newLog.username, username);

newLog.duration = duration;

newLog.score = score;

logs[numLogs++] = newLog;

FILE \*file = fopen("game\_logs.txt", "w");

for (int i = 0; i < numUsers; i++)

{

fprintf(file, "%s %2.f %d\n", newLog.username, newLog.duration, newLog.score);

}

fclose(file);

}

# 3 测试运行

## 3.1 注册登录测试

用户首次使用游戏，需要注册游戏用户名和密码：

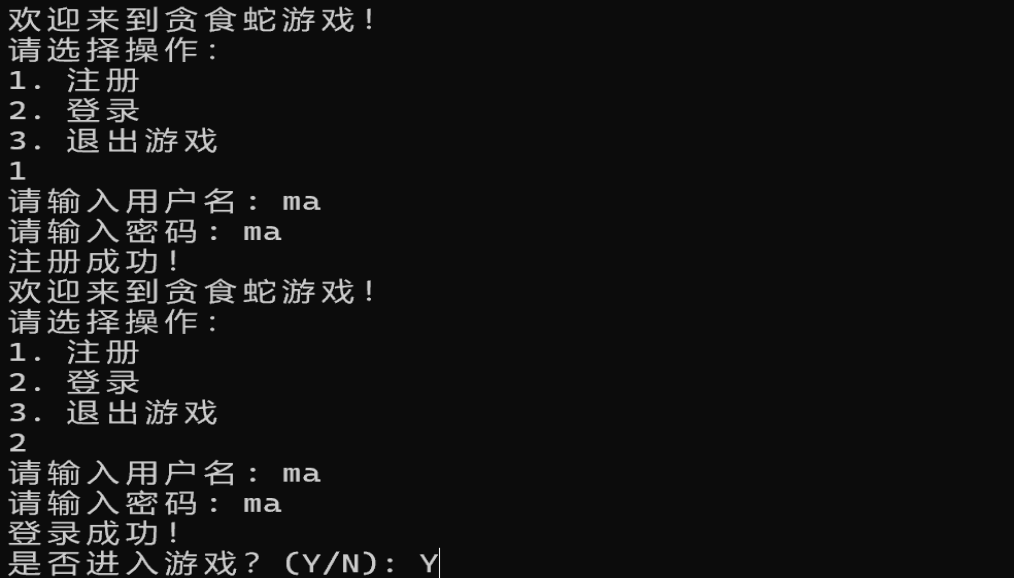


图3-1 注册登录测试结果

如果非首次使用游戏，可直接选择登录功能，直接登录：

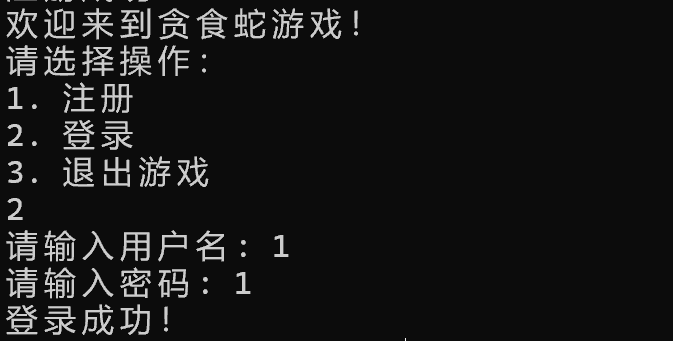


图3-2 登录测试结果

## 3.2 显示最终得分

用户进入游戏，玩游戏过程中成绩以及用户信息正确的显示在屏幕上：

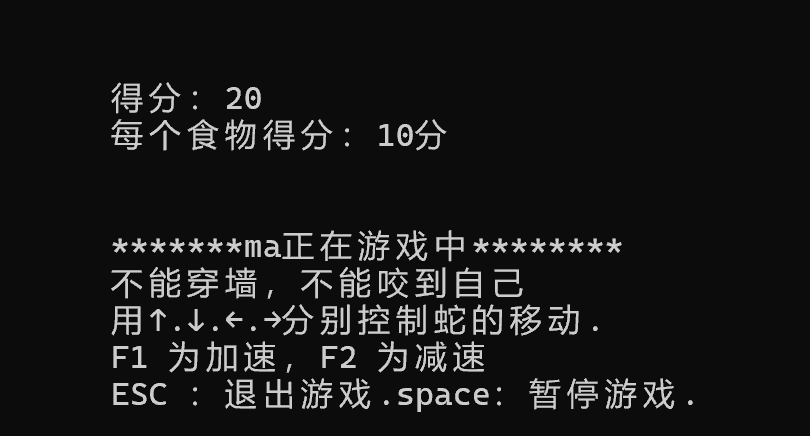


图3-3 游戏中记录测试

当蛇的头部接触到游戏界面的边缘或与自己的身体相撞时，游戏结束，用户的成绩以及撞墙原因显示在屏幕：

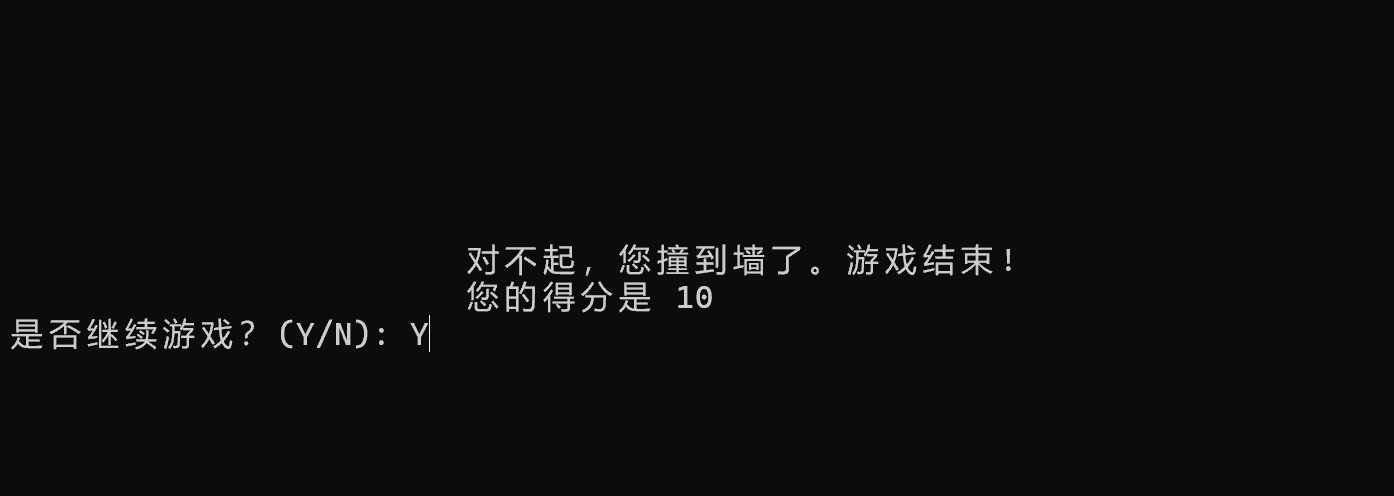


图3-4 分数显示测试

## 3.3 显示游戏日志

本游戏中用户的登录信息将记录在文件users.txt文件中：

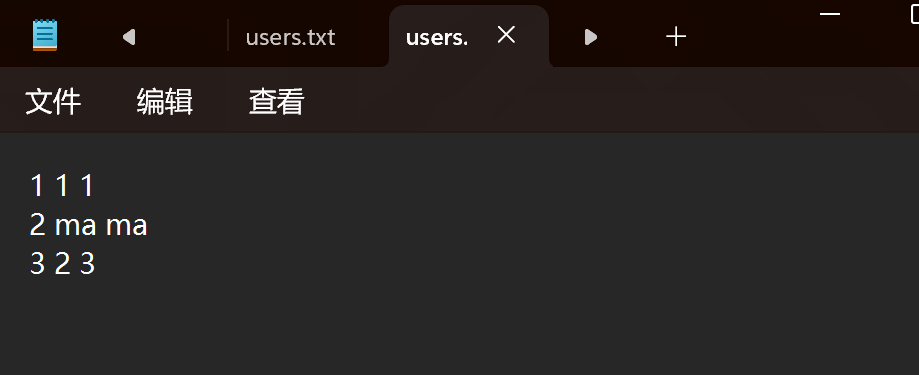


图3-5 用户表

本游戏用户的成绩以及游戏时间将会记录在game\_logs.txt文件中：

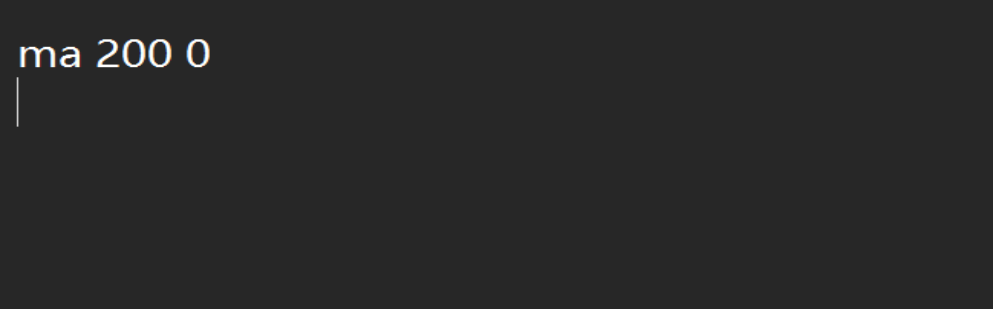


图3-6 游戏日志

# 4 总结

该贪吃蛇游戏实现了用户可自行进行注册和登录并进入游戏，在游戏的过程中可通过按动F1进行加速F2进行减速，通过方向键控制蛇的移动，每吃一个食物得10分，用户可以在游戏进行过程中随时暂停游戏，在游戏的过程中如果撞到墙，咬到自己或按ESC键则游戏结束。用户可以在game\_logs.txt文件中查看自己的游戏日志，通过完善游戏用户在一局游戏结束后可以选择继续游戏重新登录进行下一次。但也存在一些问题：该游戏在连续运行几次后登录进入游戏主页面时会出现卡顿的现象。