同濟大學
TONGJI UNIVERSITY

高级语言程序设计大作业

装

订

线

目录

装 订 线

1. 设计思路与功能描述

1.1 设计思路：

本次大作业核心在于使用面向对象编程，实现贪吃蛇小游戏的逻辑设计。下面对类进行设计。我把类看作程序设计中的元素，可能是实物元素，也可能是抽象元素。比如在贪吃蛇中，显然的实物元素就是蛇、食物、墙。而考虑到要对历史纪录进行操作，所以可以把历史信息作为一个抽象元素。除了历史纪录这一部分外，便是整体的游戏，所以可以把整体的游戏作为一个抽象元素。由此我们便设计出了五个类。进而考虑每个类的属性和方法。

蛇属性：位置、预留位置(方便吃掉食物后延长蛇身)、长度、颜色、生命、移动方向。

蛇方法：移动、吃食物、判断是否撞墙、判断是否吃自己、生成蛇、画蛇、变成食物、变成墙。

食物属性：位置(可同时标明食物所代表的分数，以及是否是精灵果、恶魔果)、食物数量。

食物方法：生成食物、画食物、判断是否占满空间。

墙属性：位置。

墙方法：开始时生成四周的墙、画墙、判断是否占满空间。

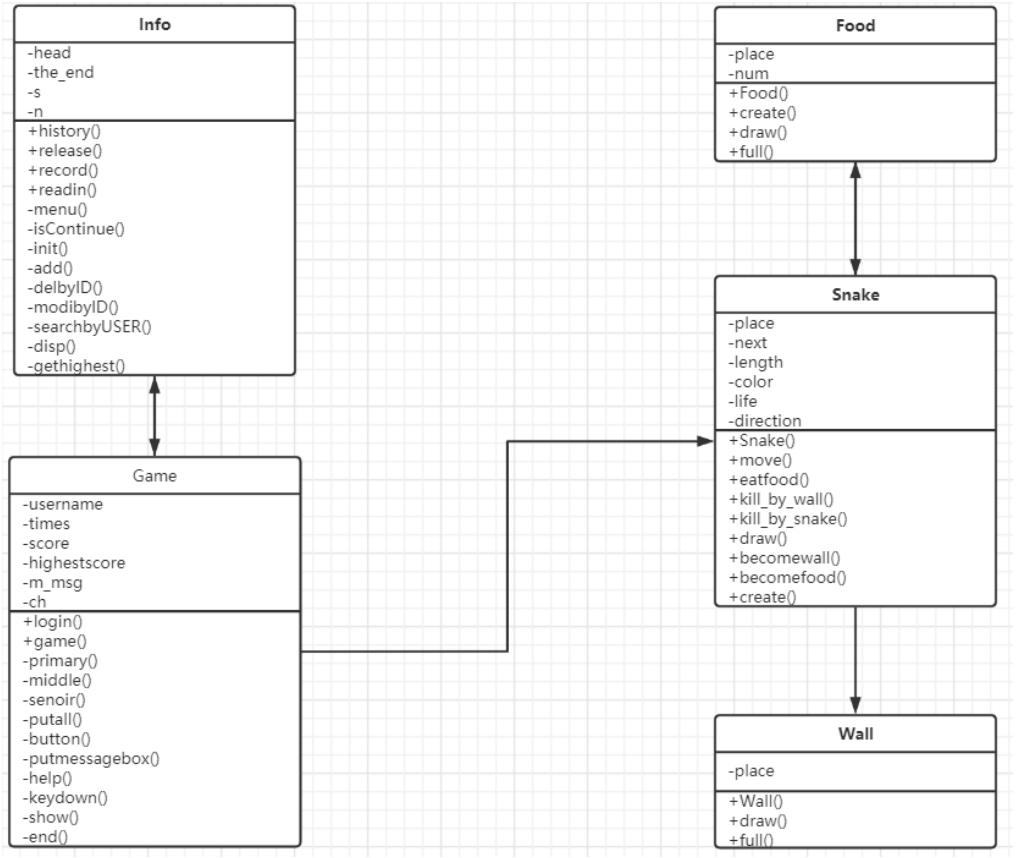
历史信息属性：链表头、链表尾、工作指针、总信息数。

历史信息方法：形成历史纪录入口、释放内存、游戏结束后进行记录、将信息链表读入文件、操作菜单、判断是否继续、初始化信息链表、增、删、查、改、显示、获得历史最高分。

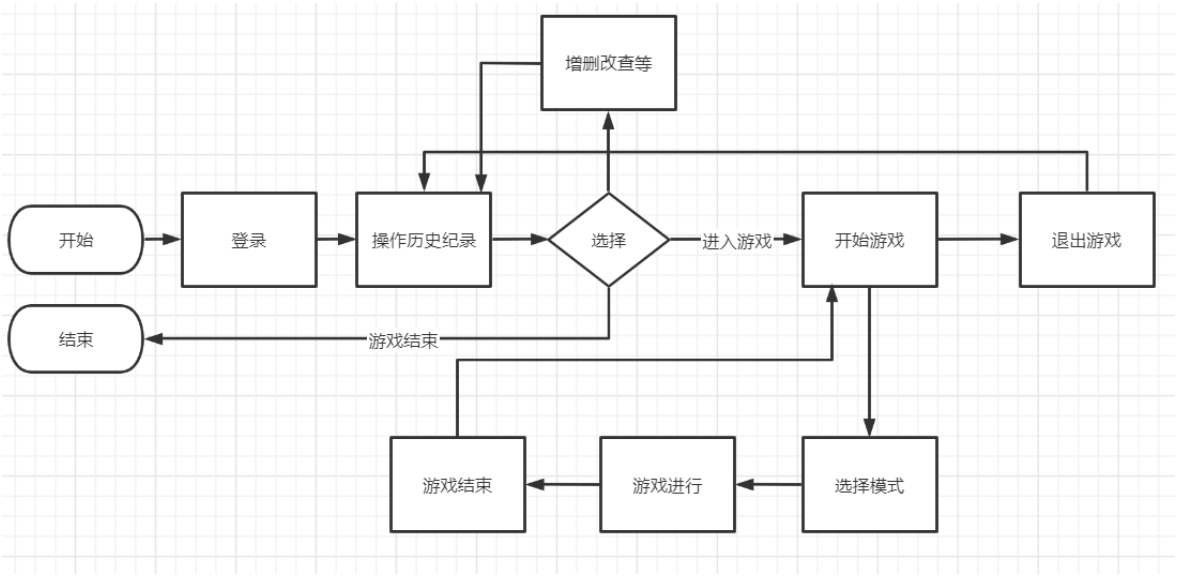
游戏属性：用户名、游戏时长、游戏得分、三个版本的历史最高分、鼠标信息、键盘信息。

游戏方法：登录、形成游戏入口、入门、进阶、高级、绘制背景、按钮、文本框、帮助信息、按键、部分 UI 展示、游戏结束后的提示。

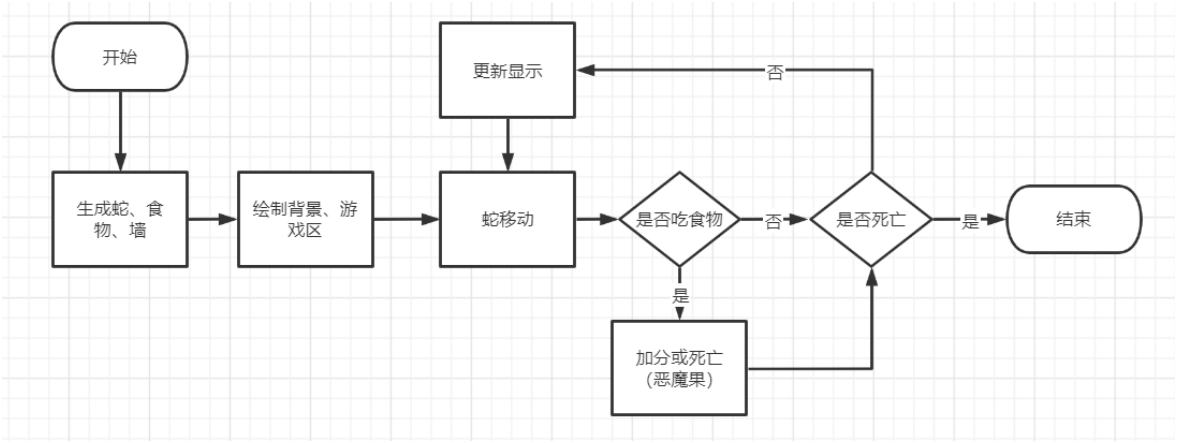
根据上述类及其属性和方法，形成类图如下：



程序总流程图如下：



游戏进行流程图如下：



1.2 功能描述：

1）Game:

void login()：

打印提示信息和登录

void game(Info B)：

游戏入口，游戏开始界面，选择游戏模式

void primary()：

入门

void middle()：

进阶

void senior()：

高级

void putall()：

绘制背景

bool button(int a, int b, const wchar_t str[]):
    放置按钮
int putmessagebox(int x, int y, int wight, int hight, const wchar_t title[], const wchar_t* text[], int g_num, int var = MY_OK):
    输出对话框
void help():
    帮助信息
int keydown(int& direction):
    按键
void show(int level):
    部分 UI--游戏时间、得分、历史最高分
bool end():
    每次游戏结束后的对话框

2）Info:

void history(Game &A):
    操作历史纪录入口
void release():
    释放内存
void record(int level,char username[],int score):
    每次游戏结束后记录
void readin():
    读入文件
void menu():
    操作历史纪录菜单
bool isContinue():
    选择是否继续
void init():
    初始化记录链表
void add():
    添加
void delbyID():
    删除
void modibyID():
    修改
void searchbyUSER():
    查找
void gethighest(Game &A):
    获取最高分

3）Snake:

Snake():

构造函数
void move()：
移动
int eatfood(Food &food)：
吃食物
bool kill_by_wall(Wall wall)：
撞墙而死
bool kill_by_snake()：
撞蛇而死
void draw()：
画出蛇
void becomewall(Wall& wall)：
变成墙
void becomefood(Food& food)：
变成食物
void create(Wall wall, Food food)：
生成蛇

4）Wall
Wall()：
构造函数
void draw()：
画出墙
bool full()：
判断是否还有空间生成蛇

5）Food
Food()：
构造函数
void create(Snake snake,int n)：
制造食物
void draw()：
画出食物
bool full()：
判断是否还有空间生成蛇

2.在实验过程中遇到的问题及解决方法

2.1 类的构建：

如何合理构建类？

通过分析贪吃蛇大作战中重要的元素，比如文件、游戏主体、游戏中基本元素(蛇、食物、墙)，划分成不同的类。另外，关键是类与类之间的交互。比如蛇和食物之间的交互，制造蛇的

时候需要考虑这一位置有没有食物、有没有墙；制造食物的时候也要考虑这一位置有没有蛇、有没有墙。

2.2 图形化：

如何简便的实现部分图形化？

这个游戏图形化最大的难点在于控件和文本框的使用。Cpp + EasyX 做控件其实是不太方便的。通过建立 Game 类中 button 这一方法，参数为位置和该位置要放的控件名称；putmassagebox 这一方法，参数为位置及文本框中的文本，较为简便地实现了部分图形化。

对于操作历史纪录这一部分的控件以及增删查改对应的各类文本框难以实现，同时出于逻辑考虑，所以没有做操作历史纪录和登录的图形化。

3. 心得体会

3.1 程序和算法设计的体会：

1）面向对象编程，最重要的一点是合理构建类，注意类与类之间的交互。

2）面向对象一方面要注重封装，但不能寄希望实现完全的封装，在效率和安全之间要合理平衡。

3）注重接口思想。引入接口的好处就是，我们可以通过操作接口来控制内部的一些东西，而不需要直接控制，使程序更有层次性，更简便。

3.2 编程过程的体会：

1）编程过程中要善于 Debug。程序中出现 bug 非常正常，通过断点、单步调试、变量监控等往往能快速锁定 bug。

2）实践出真知。通过亲自实验、反复尝试，就能找到相对较好的程序和算法设计。

3）搜索引擎是自学过程中的好老师。在遇到自己不了解的部分时，可以去 cppreference、微软 c++文档、scan、stackoverflow 等地方寻找解决方法。

4. 源代码：

Head.h

```
#pragma once
//头
#include <time.h>
#include <conio.h>
#include <vector>
#include <graphics.h>
#include<iostream>
#include<fstream>
using namespace std;
//按钮
#define MY_OK 0//只有“确定”按钮
#define MY_YESNO 1//有“是”按钮和“否”按钮
//枚举蛇的方向
enum position { up, down, Left, Right };
```

Game.h

```
#pragma once      // 防止该头文件被多次调用
#include "Head.h" // 初始化头文件
class Snake;
class Wall;
class Food;
class Info;
class Game;
//坐标属性
struct pos
{
    int x;
    int y;
};
//记录属性
struct info
{
    int level;//入门、进阶、高级
    char username[10];//用户名
    int score;//得分
    info* next;
};
//游戏
class Game
{
public:
    void game(Info B);//游戏入口，游戏开始界面，选择游戏模式
    void login();//登录
private:
    void primary();//入门
    void middle();//进阶
    void senior();//高级
    void putall();//绘制背景
    bool button(int a, int b, const wchar_t str[]);//放置按钮
    int putmessagebox(int x, int y, int wight, int hight, const wchar_t title[], const wchar_t*
text[], int g_num, int var = MY_OK);//输出对话框
    void help();//帮助信息
    int keydown(int& direction);//按键
    void show(int level);//UI：游戏时间、得分、历史最高分
    bool end();//每次游戏结束后产生对话框
    char username[10];//用户名
    double times;//游戏时间
    char ch;//键盘消息
    MOUSEMSG m_msg;//鼠标消息
    int score;//得分
    int highestscore[3];//历史最高分
    friend class Info;
};
//蛇
class Snake
{
public:
    Snake();//构造函数
```

```
    void move();//移动
    int eatfood(Food &food);//吃食物
    bool kill_by_wall(Wall wall);//撞墙而死
    bool kill_by_snake();//撞蛇而死
    void draw();//画出蛇
    void becomewall(Wall& wall);//变成墙
    void becomefood(Food& food);//变成食物
    void create(Wall wall, Food food);//出现蛇
private:
    vector <pos> place;//每节坐标
    pos next;//为下一节预留的位置
    int length;//长度
    COLORREF color;//每节颜色
    int life;//生命数
    int direction;//方向
    friend class Food;
    friend class Game;
};
//墙
class Wall
{
public:
    Wall();//构造函数
    void draw();//画出墙
    bool full();//判断是否还有空间生成蛇
private:
    bool place[45][45];//墙位置
    friend class Snake;
};
//食物
class Food
{
public:
    Food();//构造函数
    void create(Snake snake,int n);//出现食物
    void draw();//画出食物，部分UI
    bool full();//判断是否还有空间生成蛇
private:
    int place[45][45];//食物位置及所占分数或不同食物
    int num;//食物总数
    friend class Snake;
};
//历史信息
class Info
{
public:
    void history(Game &A);//操作历史纪录入口
    void release();//释放内存
    void record(int level,char username[],int score);//每次游戏结束后记录
    void readin();//读入文件
private:
    void init();//初始化记录链表
    bool isContinue();//选择是否继续
```

```
    void menu();//操作历史纪录菜单
    void disp();//显示
    void searchbyUSER();//查找
    void delbyID();//删除
    void modibyID();//修改
    void add();//添加
    void gethighest(Game& A);//获取最高分
    info* head, * s, * the_end;//链表头指针、工作指针、尾指针
    int n;//总记录数
    friend class Game;
};
```

main.cpp

```
#include "Game.h"
// 程序主函数
int main()
{
    srand((int)time(0));
    Game *A=new Game();
    A->login();//登录
    Info B;
    B.history(*A);//操作历史纪录入口
    A->game(B);//开始游戏
    closegraph();//游戏结束
    B.readin();//将所有进行的游戏记录读入文件
    B.release();//释放内存
    B.history(*A);//操作历史纪录入口
    delete A;
    B.release();//释放内存
    return 0;
}
```

info.cpp

```
#include "Game.h"      // 游戏头文件
bool Info::isContinue()
{
    cout << "是否继续?(y/n): \n";
    char tmp;
    cin >> tmp;
    if (tmp == 'Y' || tmp == 'y')
        return true;
    return false;
}
////操作历史纪录菜单
void Info::menu()
{
    cout << endl;
    for (int i = 0; i < 57; i++)
    {
        cout << '*';
    }
```

```
    cout << endl;
    cout << "\t1 查找记录\t2 删除记录\t3 修改记录\n";
    cout << "\t4 添加记录\t5 显示记录\t0 退出操作\n";
    for (int i = 0; i < 57; i++)
    {
        cout << '*';
    }
    cout << endl;
}
//初始化记录链表
void Info::init()
{
    ifstream fin("history.txt");
    if (!fin)
    {
        cerr << "文件打不开" << endl;
        exit(-1);
    }
    fin >> n;
    info* s = new(nothrow)info;//尾插法建立链表
    if (s == NULL)
    {
        cout << "没有内存" << endl;
        exit(1);
    }
    else
    {
        head = the_end = s;
        for (int i = 0; i < n; i++)
        {
            fin >> s->level >> s->username >> s->score;
            s = new(nothrow)info;
            if (s == NULL)
            {
                cout << "没有内存" << endl;
                exit(1);
            }
            else
            {
                the_end->next = s;
                the_end = s;
                s->next = NULL;
            }
        }
    }
    fin.close();
}
//显示
void Info::disp()
{
    info* q = head;
    while (q != the_end)
    {
```

```
            cout << "版本：";
            if (q->level == 1)
                cout << "入门版";
            else if (q->level == 2)
                cout << "进阶版";
            else
                cout << "高级版";
            cout << " 用户：" << q->username << " 得分：" << q->score << endl;
            q = q->next;
        }
    }
    //查找
    void Info::searchbyUSER()
    {
        cout << "请输入用户名" << endl;
        char username[10];
        cin >> username;
        info* q = head;
        while (q != the_end)
        {
            if (!strcmp(q->username, username))
            {
                cout << "版本：";
                if (q->level == 1)
                    cout << "入门版";
                else if (q->level == 2)
                    cout << "进阶版";
                else
                    cout << "高级版";
                cout << " 用户：" << q->username << " 得分：" << q->score << endl;
            }
            q = q->next;
        }
    }
    //删除
    void Info::delbyID()
    {
        while (1)
        {
            int new_no;
            cout << "请输入序号(根据所有用户的全部记录的序列，可先使用显示记录显示)" << endl;
            cin >> new_no;
            if (new_no > n)
                cout << "不存在" << endl;
            else if (head == NULL)
                cout << "没有记录" << endl;
            else
            {
                n--;
                info* p = head, * q = p;
                int i = 1;
                while (p != the_end && i!= new_no)
                {
```

装 订 线

```
                i++;
                q = p;
                p = p->next;
            }
            if (p == head)
            {
                head = p->next;
                delete p;
            }
            else if (p != the_end)
            {
                q->next = p->next;
                delete p;
            }
        }
        if (!isContinue())
            break;
    }
}
//修改
void Info::modibyID()
{
    while (1)
    {
        int new_no;
        cout << "请输入序号(根据所有用户的全部记录的序列，可先使用显示记录显示)" << endl;
        cin >> new_no;
        if (new_no > n)
            cout << "不存在" << endl;
        else
        {
            info* q;
            int i=0;
            char username[10];
            for (q = head; q != the_end; q = q->next)
            {
                i++;
                if (i == new_no)
                {
                    cout << "请输入要修改的用户名" << endl;
                    cin >> username;
                    strcpy_s(q->username, username);
                }
            }
        }
        if (!isContinue())
            break;
    }
}
//添加
void Info::add()
{
    while (1)
```

```
    {
        cout << "请输入版本(输入1代表入门，2代表进阶，3代表高级)、用户名和得分" << endl;
        cin >> the_end->level >> the_end->username >> the_end->score;
        s = new(nothrow)info;
        if (s == NULL)
        {
            cout << "没有内存" << endl;
            exit(1);
        }
        else
        {
            n++;
            the_end->next = s;
            the_end = s;
            s->next = NULL;
        }
        if (!isContinue())
            break;
    }
}
//内存释放
void Info::release()
{
    info* q = head, * p;
    while (q != the_end)
    {
        p = q;
        q = q->next;
        delete p;
    }
    delete q;
}
//获取最高分
void Info::gethighest(Game& A)
{
    info* q = head;
    for(int i=0;i<3;i++)
        A.highestscore[i] = 0;
    while (q != the_end)
    {
        if (!strcmp(q->username, A.username))
        {
            if (A.highestscore[q->level-1] < q->score)
                A.highestscore[q->level-1] = q->score;
        }
        q = q->next;
    }
}
//每次游戏结束后记录
void Info::record(int level,char username[],int score)
{
    the_end->level = level;
    strcpy_s(the_end->username,username);
```

装 订 线

```
    the_end->score = score;
    s = new(nothrow)info;
    if (s == NULL)
    {
        cout << "没有内存" << endl;
        exit(1);
    }
    else
    {
        the_end->next = s;
        the_end = s;
        s->next = NULL;
    }
}
//读入文件
void Info::readin()
{
    ofstream fout("history.txt");
    if (!fout)
    {
        cerr << "文件打不开" << endl;
        exit(-1);
    }
    info* q = head;
    int num = 0;//先得到总记录数
    while (q->next != NULL)
    {
        num++;
        q = q->next;
    }
    fout << num << endl;
    q = head;
    while (q->next != NULL)
    {
        fout << q->level << " " << q->username << " " << q->score << endl;
        q = q->next;
    }
    fout.close();
}
//操作历史纪录入口
void Info::history(Game& A)
{
    cout << endl << "现在可对历史纪录进行增删改查" << endl;
    menu();//菜单
    init();//初始化链表
    gethighest(A);//获取最高分
    int choice;
    char c;
    while (1)
    {
        cout << "选择菜单项(0~5):";
        cin >> choice;
        if (choice == 0)        //选择退出
```

装 订 线

```
        {
            cout << "确定退出吗?(y/n)" << endl;
            cin >> c;
            if (c == 'y' || c == 'Y')
            {
                break;
            }
            else
                continue;
        }
        switch (choice)
        {
        case 1: searchbyUSER(); break;
        case 2: delbyID(); break;
        case 3: modibyID(); break;
        case 4: add(); break;
        case 5: disp(); break;
        default:
            cout << "输入错误，请从新输入" << endl;
        }
    }
}
```

装 订 线

game.cpp

```
#include "Game.h"
//登录
void Game::login()
{
    cout << "本程序为贪吃蛇大作战，程序进行顺序如下：\n";
    cout <<"首先登录，然后可操作历史记录，然后进入游戏，游戏结束后可再次操作历史纪录，然后结束\n";
    cout << "文件保存在history.txt(里面有部分初始数据)\n";
    cout << "主要实现加分项：\n";
    cout <<"1.部分图形化. 发现操作历史纪录这一部分的控件太难做，同时考虑逻辑因素，所以没有做登录和历史纪录的图形化\n";
    cout << "2.更多的游戏逻辑:加速区(上半部)减速区(下半部)、精灵果(白色)和恶果(黑色)\n";
    cout << "-------------------------------------------------------------\n";
    cout << "欢迎来到贪吃蛇大作战，请先输入账号(长度小于10)进行登录\n\n账号：";
    cin >> username;
    cout << "\n登陆成功\n";
}
//游戏入口，游戏开始界面，选择游戏模式
void Game::game(Info B)
{
    cout << "现在进入真正的游戏";
    Sleep(3000);
    initgraph(640, 450);
    setbkmode(TRANSPARENT);
    BeginBatchDraw();
    while (1)
    {
        times = 0;
```

```
        score = 0;

        if (_kbhit())//键盘消息获取
        {
            ch = _getch();
            if (ch == 27)//按 Esc 退出
            {
                break;
            }
        }

        if (MouseHit())//鼠标消息获取
            m_msg = GetMouseMsg();

        putall();//绘制背景
        settextcolor(WHITE);
        settextstyle(70, 0, L"微软雅黑");
        outtextxy(235, 85, L"贪吃蛇");//标题输出
        if (button(280, 200, L"  入门  "))//控件
        {
            primary();//入门
            B.record(1, username, score);//记录游戏信息
        }

        if (button(280, 250, L"  进阶  "))
        {
            middle();//进阶
            B.record(2, username, score);
        }

        if (button(280, 300, L"  高级  "))
        {
            senior();//高级
            B.record(3, username, score);
        }

        if (button(280, 350, L"  帮助  "))
            help();//帮助信息

        settextcolor(WHITE);
        outtextxy(249, 400, L"按 Esc 退出游戏");

        FlushBatchDraw();
    }

    EndBatchDraw();
}
//绘制背景
void Game::putall()
{
    setbkcolor(RGB(50, 50, 50));
    cleardevice();
}
```

```
//放置按钮
bool Game::button(int a, int b, const wchar_t str[])
{
    static int x, y;//鼠标坐标

    x = m_msg.x;//获取坐标
    y = m_msg.y;

    setfillcolor(RGB(100, 100, 100));//绘制边框
    solidrectangle(a - 25, b, a + 19 * wcslen(str) + 25, b + 30);

    if (x > a - 25 && (size_t)x < a + 19 * wcslen(str) + 25 && y > b && y < b + 30)//获得焦点
显示
    {
        setfillcolor(RGB(150, 150, 150));
        solidrectangle(a - 25, b, a + 19 * wcslen(str) + 25, b + 30);
        if (m_msg.uMsg == WM_LBUTTONUP)
        {
            m_msg.uMsg = WM_MOUSEMOVE;
            return 1;
        }
    }
    settextstyle(25, 0, L"微软雅黑");//输出文字
    settextcolor(WHITE);
    outtextxy(a, b + 3, str);

    return 0;
}
//绘制对话框
int Game::putmessagebox(int x, int y, int wight, int hight, const wchar_t title[], const wchar_t*
text[], int g_num, int var)
{
    setbkmode(TRANSPARENT);
    while (1)
    {
        if (_kbhit())//键盘消息清空
            ch = _getch();

        if (MouseHit())
            m_msg = GetMouseMsg();//鼠标消息获取

        setfillcolor(RGB(25, 25, 25));//绘制边框
        solidrectangle(x, y, x + wight, y + hight);

        settextstyle(30, 0, L"微软雅黑");//输出标题
        outtextxy(x + 20, y + 10, title);
        for (int i = 0; i < g_num; i++)
        {
            settextstyle(18, 0, L"微软雅黑");
            outtextxy(x + 20, y + 45 + i * 20, text[i]);
        }

        if (var == MY_OK)//按钮放置
```

装 订 线

```
        {
            if (button(x + wight - 70, y + hight - 37, L"确定"))
                return 0;
        }
        else if (var == MY_YESNO)
        {
            if (button(x + wight - 125, y + hight - 37, L"是"))
                return 1;

            if (button(x + wight - 50, y + hight - 37, L"否"))
                return 0;
        }
        FlushBatchDraw();
    }
    return 0;
}
//帮助信息
void Game::help()
{
    while (1)
    {
        if (MouseHit())//鼠标消息获取
            m_msg = GetMouseMsg();

        putall();//绘制背景

        settextcolor(WHITE);//标题输出
        settextstyle(52, 0, L"黑体");
        outtextxy(250 + 3, 30 + 3, L"帮 助");

        setfillcolor(RGB(100, 100, 100));//绘制边框
        setlinecolor(BLACK);
        setlinestyle(PS_SOLID, 5);
        fillrectangle(125, 120, 510, 360);

        settextstyle(25, 0, L"微软雅黑");//信息输出
        outtextxy(140, 130, L"游戏开始选择不同关卡");
        outtextxy(140, 170, L"有三种：");
        outtextxy(140, 200, L"入门");
        outtextxy(140, 230, L"进阶");
        outtextxy(140, 260, L"高级");
        outtextxy(140, 290, L"选择后，进入游戏");
        outtextxy(140, 320, L"WSAD或上下左右操作");

        if (button(280, 400, L"回到主页"))//回到主页按钮
            break;

        FlushBatchDraw();
    }
}
//按键
int Game::keydown(int &direction)
{
```

装 订 线

```
    char userKey = _getch();
    if (userKey == 27)//提前退出游戏，返回5则结束
    {
        if (end())
            return 5;
        else
            return 0;
    }
    if (userKey == -32)
        userKey = -_getch();//获取具体方向
    switch (userKey)
    {
    case 'w':
    case 'W':
    case -72:
        if (direction != down)
            direction = up;
        break;
    case 's':
    case 'S':
    case -80:
        if (direction != up)
            direction = down;
        break;
    case 'a':
    case 'A':
    case -75:
        if (direction != Right)
            direction = Left;
        break;
    case 'd':
    case 'D':
    case -77:
        if (direction != Left)
            direction = Right;
        break;
    }
    return direction;
}
//部分UI显示
void Game::show(int level)
{
    settextstyle(20, 0, L"微软雅黑");
    TCHAR s[10];
    outtextxy(480, 150, _T("分数"));
    _stprintf_s(s, _T("%d"), score);
    outtextxy(550, 150, s);
    outtextxy(480, 200, _T("历史最高分"));

    if (score > highestscore[level-1])//判定最高分
        highestscore[level-1] = score;

    _stprintf_s(s, _T("%d"), highestscore[level-1]);
```

```
    outtextxy(550, 200, s);
    int second = int(times);
    int decimal = int((times * 100)) % 100;
    outtextxy(480, 250, _T("时间"));
    _stprintf_s(s, _T("%d:"), second);
    outtextxy(550, 250, s);
    _stprintf_s(s, _T("%d"), decimal);
    if (second < 10 && second>0)
        outtextxy(565, 250, s);
    else if(second < 100 && second>10)
        outtextxy(575, 250, s);
    else if (second>100)
        outtextxy(585, 250, s);
    outtextxy(480, 300, _T("按ESC可提前结束游戏"));
}
//每次游戏结束后产生对话框
bool Game::end()
{
    wchar_t title[50];
    swprintf_s(title, L"游戏时长 %d s\0", int(times));
    while (1)
    {
        if (_kbhit())//键盘消息获取
            ch = _getch();

        if (MouseHit())//鼠标消息获取
            m_msg = GetMouseMsg();

        putall();//绘制背景

        settextcolor(WHITE);//标题输出
        settextstyle(52, 0, L"黑体");
        outtextxy(140, 90, title);

        if (button(273, 300, L"回到主页"))//回到主页按钮
        {
            const wchar_t* text[10];
            text[0] = L"你确定你要回到主页吗？\n";
            button(273, 300, L"回到主页");
            if (putmessagebox(120, 165, 400, 150, L"回到主页", text, 1, MY_YESNO))//回到主页
对话框
            {
                return 1;
            }
        }
        FlushBatchDraw();
    }
    return 0;
}
```

ingame.cpp

```
#include "Game.h"
```

```
//入门
void Game::primary()
{
    Snake snake;
    Food food;
    Wall wall;
    food.create(snake,1);//正常出现食物
    while (1)
    {
        putall();//绘制背景
        if (snake.place[0].y > 22)//划分加速、减速区
        {
            Sleep(150);
            times += 0.15;//累加时间
        }
        else
        {
            Sleep(100);
            times += 0.1;
        }
        setfillcolor(RGB(100, 100, 100));//绘制游戏区
        solidrectangle(0, 0, 450, 450);

        ch = '#';// 键盘消息清空
        if (_kbhit())// 键盘消息获取
            if (keydown(snake.direction) == 5)//按键决定蛇方向和是否提前退出
                break;
        if (MouseHit())//鼠标消息获取
            m_msg = GetMouseMsg();

        snake.move();//蛇移动
        int foodlevel = snake.eatfood(food);//吃食物
        if (foodlevel>0&& foodlevel <6)//一般食物
        {
            score += foodlevel*10;
        }
        else if(foodlevel==6)//精灵果加分
        {
            score += 100;
        }
        else if (foodlevel == 7)//恶魔果死亡
        {
            break;
        }
        food.create(snake,1);//正常出现食物
        if (snake.kill_by_wall(wall) || snake.kill_by_snake())//判断撞墙、撞蛇
            break;

        wall.draw();//绘图及UI展示
        snake.draw();
        food.draw();
        show(1);
```

```
            FlushBatchDraw();
        }
        Sleep(3000);
        if (end())//每次游戏结束后产生对话框
            return;
    }
    //进阶
    void Game::middle()
    {
        Snake snake;
        Food food;
        Wall wall;
        food.create(snake,1);
        while (1)
        {
            putall();
            if (snake.place[0].y > 22)
            {
                Sleep(150);
                times += 0.15;
            }
            else
            {
                Sleep(100);
                times += 0.1;
            }
            setfillcolor(RGB(100, 100, 100));
            solidrectangle(0, 0, 450, 450);

            ch = '#';
            if (_kbhit())
                if (keydown(snake.direction) == 5)
                    break;
            if (MouseHit())
                m_msg = GetMouseMsg();

            snake.move();
            int foodlevel = snake.eatfood(food);
            if (foodlevel > 0 && foodlevel < 6)
            {
                score += foodlevel * 10;
            }
            else if (foodlevel == 6)
            {
                score += 100;
            }
            else if (foodlevel == 7)
            {
                break;
            }
            food.create(snake,1);
            if (snake.kill_by_wall(wall) || snake.kill_by_snake())//关键不同
            {
```

装 订 线

```
                Sleep(2000);
                snake.becomewall(wall);//蛇变成墙
                if (wall.full())//墙占满空间
                    break;
                else
                {
                    snake.create(wall,food);
                    food.create(snake,2);//重新出现食物
                }
            }

            wall.draw();
            snake.draw();
            food.draw();
            show(2);

            FlushBatchDraw();
        }
        Sleep(3000);
        if (end())
            return;
    }
    //高级
    void Game::senior()
    {
        Snake snake;
        Food food;
        Wall wall;
        snake.life = 5;
        food.create(snake,1);
        while (1)
        {
            putall();// 绘制背景
            if (snake.place[0].y > 22)
            {
                Sleep(150);
                times += 0.15;
            }
            else
            {
                Sleep(100);
                times += 0.1;
            }
            setfillcolor(RGB(100, 100, 100));
            solidrectangle(0, 0, 450, 450);

            ch = '#';
            if (_kbhit())
                if (keydown(snake.direction) == 5)
                    return;
            if (MouseHit())
                m_msg = GetMouseMsg();
```

```
            snake.move();
            int foodlevel = snake.eatfood(food);
            if (foodlevel > 0 && foodlevel < 6)
            {
                score += foodlevel * 10;
            }
            else if (foodlevel == 6)
            {
                score += 100;
            }
            else if (foodlevel == 7)
            {
                break;
            }
            food.create(snake,1);
            if (snake.kill_by_wall(wall) || snake.kill_by_snake())//关键不同
            {
                Sleep(2000);
                snake.becomefood(food);//蛇变成食物
                if (food.full())//食物占满空间
                    break;
                else
                {
                    snake.create(wall,food);
                    food.create(snake, 3);//重新出现食物
                    snake.life--;
                }
            }
            if (snake.life == 0)//蛇生命数为0
                break;

            wall.draw();
            snake.draw();
            food.draw();
            show(3);

            FlushBatchDraw();
        }
        Sleep(3000);
        if (end())
            return;
}
```

snake.cpp

```
#include "Game.h"
//构造函数
Snake::Snake()
{
    pos xy;
    xy = { 4,2 };
    place.push_back(xy);
    length = 1;
```

```
    color = RGB(100,149,237);
    direction = Right;
    next = { 0,0 };
    life = 1;
}
//生成蛇
void Snake::create(Wall wall,Food food)
{
    place.clear();//清空
    pos xy;
    while (1)
    {
        xy.x = rand() % 43 + 1;
        xy.y = rand() % 43 + 1;
        if (wall.place[xy.y][xy.x] == 0&&food.place[xy.y][xy.x]==0)
        {
            place.push_back(xy);
            break;
        }
    }
    length = 1;
    color = RGB(100, 149, 237);
    direction = rand() % 4 + 1;
    next = { 0,0 };
}
//蛇移动
void Snake::move()
{
    next = place[length - 1];
    for (int i =length-1;i>0;i--)//将除蛇头以外的节移动到它的前面一节
        place[i] = place[i - 1];
    switch (direction)//根据当前移动方向移动蛇头
    {
    case Right:
        place[0].x += 1;
        break;
    case Left:
        place[0].x -= 1;
        break;
    case up:
        place[0].y -= 1;
        break;
    case down:
        place[0].y += 1;
    }
}
// 吃食物
int Snake::eatfood(Food &food)
{
    int level = food.place[place[0].y][place[0].x];//吃到食物的级别：0没吃到，1到5正常，6精灵果，7恶魔果
    if (level!=0)
    {
```

```
            length++;
            place.push_back(next);//新增一个节到预留位置
            food.num--;
            food.place[place[0].y][place[0].x] = 0;
            return level;
        }
        return 0;
}
//撞墙
bool Snake::kill_by_wall(Wall wall)
{
        if (wall.place[place[0].y][place[0].x] == 1)
        {
            for (int i = 0; i <= length - 2; i++)//返回到撞墙之前的状态
                place[i] = place[i + 1];
            place[length - 1] = next;
            return true;
        }
        return false;
}
//撞到蛇
bool Snake::kill_by_snake()
{
        for (int i = 1; i < length; i++)
        {
            if (place[i].x == place[0].x && place[i].y == place[0].y)
                return true;
        }
        return false;
}
// 画蛇，部分UI
void Snake::draw()
{
        for (int i = 0; i < length; i++)
        {
            setfillcolor(color);
            fillrectangle(place[i].x*10, place[i].y*10, place[i].x*10 + 10, place[i].y*10 + 10);
        }

        settextstyle(20, 0, L"微软雅黑");
        settextcolor(WHITE);
        TCHAR s[10];
        outtextxy(480, 50, _T("长度"));
        _stprintf_s(s, _T("%d"), length);
        outtextxy(550, 50, s);
        outtextxy(480, 100, _T("生命"));
        _stprintf_s(s, _T("%d"), life);
        outtextxy(550, 100, s);
}
//蛇变成墙
void Snake::becomewall(Wall &wall)
{
        for (int i = 0; i < length; i++)
```

```
    {
        wall.place[place[i].y][place[i].x] = 1;
    }
}
//蛇变成食物
void Snake::becomefood(Food& food)
{
    for (int i = 0; i < length; i++)
    {
        food.place[place[i].y][place[i].x] = rand() % 5 + 1;;
    }
    food.num += length;
}
```

food.cpp

```
#include "Game.h"
//构造函数
Food::Food()
{
    num = 0;
    memset(place, 0, sizeof(place));
}
//出现食物
void Food::create(Snake snake,int n)
{
    int devilnum = 0;//恶魔果数
    for (int i = 0; i < 45; i++)
    {
        for (int j = 0; j < 45; j++)
        {
            if (place[i][j] == 7)
                devilnum++;
        }
    }

    int newnum = 0;//新加的食物数
    if (num == 0||num==devilnum||n==2||n==3)//没有食物、只有恶魔果、进阶:重新出现食物、高级:
重新出现食物
    {
        newnum = rand() % 5 + 1;
        if (n == 2)//进阶:重新出现食物
        {
            num = 0;
            memset(place, 0, sizeof(place));
        }
        int i = num;
        while (i < num+newnum)//产生食物
        {
            int x = rand() % 43 + 1;
            int y = rand() % 43 + 1;
            for (int j = 0; j < snake.length; j++)
            {
```

```
                if (snake.place[j].y==y&&snake.place[j].x==x||place[y][x]!=0)//有蛇或其他食物
                {
                    continue;
                }
            }
            place[y][x] = rand() % 7 + 1;
            i++;
        }
        num += newnum;
    }
}
//画出食物
void Food::draw()
{
    for (int i = 0; i < 45; i++)
    {
        for (int j = 0; j < 45; j++)
        {
            if (place[i][j] != 0)
            {
                if(place[i][j]==6)
                    setfillcolor(RGB(255, 255, 255));
                else if(place[i][j] == 7)
                    setfillcolor(RGB(0, 0, 0));
                else
                    setfillcolor(RGB(rand() % 256, rand() % 256, rand() % 256));
                fillrectangle(j * 10, i * 10, j * 10 + 10, i * 10 + 10);
            }
        }
    }
}
//判断是否还有空间生成蛇
bool Food::full()
{
    if (num > 2000)
        return true;
    return false;
}
```

wall.cpp

```
#include "Game.h"
//构造函数
Wall::Wall()
{
    memset(place, 0, sizeof(place));
    for (int i = 0; i < 45; i++)
    {
        place[0][i] = 1;
        place[44][i] = 1;
        place[i][0] = 1;
        place[i][44] = 1;
```

```
    }
}
//画出墙
void Wall::draw()
{
    setfillcolor(RGB(255,215,0));
    for (int i = 0; i < 45; i++)
    {
        for (int j = 0; j < 45; j++)
        {
            if(place[i][j] == 1)
                fillrectangle(j * 10, i * 10, j * 10 + 10, i * 10 + 10);
        }
    }
}
//判断是否还有空间生成蛇
bool Wall::full()
{
    int num = 0;
    for (int i = 0; i < 45; i++)
    {
        for (int j = 0; j < 45; j++)
        {
            if (place[i][j] == 1)
                num++;
        }
    }
    if (num > 2000)
        return true;
    return false;
}
```

装 订 线