同济大学
计算机科学与技术系

贪吃蛇大作战实验报告

学　　号　1750562

姓　　名　张博

专　　业　信息安全

授课老师　陈宇飞

日　　期　2019 年 5 月 10 日

一、设计思路与功能描述

本次实验主要完成了三种游戏模式的设计。对应的就是三个关于蛇的类，其中第一类蛇是最基础的蛇即对应入门版，其中的数据成员和函数成员都是 public 关系，这样在后面的进阶版和高级版可以用于继承。

首先是选择数据成员结构的设计，对蛇主体类中，数据成员选择：长度、坐标组、方向、得分。对于食物结构体选择：数量、坐标组。这两个数据成员的选择都很自然，都是必要的因素。考虑完数据成员后，下考虑蛇内封装哪些函数：总体上看，每一版游戏的进行实际上就是一个大循环，直到最终循环跳出，即为游戏结束。因此在类中存在顶层子函数 loop,通过 loop 这个无限循环调用各个函数实现游戏功能。游戏的进行主要依赖四个点：蛇的移动、蛇的进食、蛇的死亡、蛇的显示。蛇的移动是在一个棋盘上进行，棋盘为一个二维数组蛇的每一节对应一个坐标，蛇的移动实际上就是坐标的改变。通过玩了多次贪吃蛇游戏后我发现蛇的移动就是除了蛇头在方向指引下前进一格，其余蛇身均移动为前一节蛇身上。知道了这个原理后实现移动就比较简单，SnakeMove 函数先将除蛇头外的蛇身坐标均赋值为前一节坐标，再通过判定当前方向来移动蛇头。蛇在移动中可能遇到进食和碰壁。当蛇头碰到食物时，判定为蛇吃到食物：“length++、score+=2”。这就决定了关于蛇头的坐标判定是要时刻进行的，要在每次移动时都检测是否蛇将碰壁或者蛇将进食。当蛇碰到墙壁时判定为 GameOver,确认游戏失败之后需要将相关信息输入进入文件中，由于事先存在游戏记录，先从文件中得到相应版本的历史最高记录，若本次游戏突破历史记录，则进行更新否则直接退出即可。于是就设计了三个关于文件的函数：PrintFile 用于将文件内容输出到屏幕上，ReadFile 用于将本版本所需的历史最高记录读取出来，WriteFile 用于将新的记录写入文本。这样下来就定出了顶层函数 loop 中所需要调用的基本函数。但是只有这些函数还不够，还需要决定函数的出现顺序以及该如何显示图像。图像的装载很简单，但是清屏的位置以及蛇各种行为判定的先后顺序还要考虑。蛇先移动，再判定，判定完成后进行数据改写。

对于进阶版，在继承入门版蛇的基础上做了几个函数的重写，但是整体的 loop 结构并没有很大变化。由于进阶版的特性当蛇挂掉后选择用 wall_x，与 wall_y 数组将其蛇身坐标记下，利用重写的 Displaywall(),选择将新生成的墙壁和边界墙壁同时出现。且在重写的判断 GameOver 函数中也要加入对蛇是否碰新生成墙壁的判定。由于墙的产生对游戏区域有了改变，当蛇死后随机生成的蛇以及食物的位置也要改变。当蛇头与墙壁重合时重新生成，当食物与墙壁都重合时也要重新生成。除此之外，GameOver 函数还要添加新的失败判定。当全部的可用坐标只剩下 2 个时，游戏判定为失败。

对于高级版，在继承入门版后，由于高级版对特性，关于食物的函数得到了重写。在蛇死亡后要随机生成的食物要避开所有蛇尸体变成的食物，同时在和进阶版一样判定是否游戏结束时利用了判定空间和死亡次数判定。吃不同的食物加分也不一样。

总体设计是这样，下面是每个具体函数的功能由于进阶版的蛇和高级版的蛇都是在继承入门版蛇的基础上重写了部分函数，就不一一贴出了源代码中有更为详细的注释：

```
class SNAKE_1{
```

```
    //初始化游戏界面
    void initGame()
    //加载蛇头、蛇身绘出蛇
    void SnakeDisplay()
    //控制蛇的移动
    void SnakeMove()
    //在蛇死亡时退回死之前位置
    void ReSnakeMove()
    //方向键控制蛇的方向改变
    void SnakeDirect()
    //查找是否坐标与蛇身重合
    bool find_in_snake(int x, int y)
    //生成食物坐标
    virtual void CreateFOOD()
    //判定吃到食物
    virtual void EatFOOD()
    //游戏结束输出
    virtual void EndGame(int flag)
    //判断游戏结束
    virtual  int GameOver()
    //打印食物
    virtual void DisplayFOOD()
    //打印文件内容
    void PrintFile()
    //取出版本历史最高数据
    virtual void ReadFile(int &k)
    //更新历史最高记录
    void WriteFile(int score)
    //显示墙
    virtual void DisplayWall()
    //游戏循环
    virtual  void loop()
    //等待按键
    void wait_for_enter()
};
    //菜单函数
    void Menu()
```

二、在实验中遇到的问题和方法

①：实验中遇到由于清屏太多，闪动的问题，加入了BeginFlush和EndFlush之后解决了问题。且由于当碰到墙壁时蛇死掉，但是这样就会丧失碰到时尾巴的坐标，因为这个一直无法还原进阶版中的墙壁，因为总是会少一节蛇的部分，最后建立新的变量将每一次移动时的尾巴坐标记下，在还原墙壁时进行调用完成了

问题。

②：在进阶版和高级版中会遇到判定空间不足后无法正确显示最终状态的问题。在最后一次没有空间用于随机产生蛇和食物时，各项数据坐标被更新，但是直接跳转到EndGame时并未显示，解决方式是在最后结束游戏时将各项数据再次清屏打印一次。

③：在处理高级版时一度遇见蛇在正常情况下在别处莫名闪出蛇身然后蛇撞上导致死亡次数加一。多方检查代码都没有问题，最后发现问题出现iter迭代器。由于在蛇死后产生的新的食物：糖果是用STL容器存储。在判定吃到糖果时我用for循环直接判定是否蛇头与之重合，重合则直接erase擦除。这是不对的。原因在于我擦除了之后，本身的距离可能已经发生了改变，与这时的i值并不相符

```
/*
for (unsigned int i = 0; i < food_x.size(); i++)
{
    if (s_pos[0].x == food_x[i]
        && s_pos[0].y == food_y[i])
    {
        length++;                           //长度加1
        score += 1;
        food_x.erase(food_x.begin()+i);
        food_y.erase(food_y.begin()+i);
    }
}*/
```

后经过订正，利用迭代器将擦除后的指针返回值记下，再继续往后判断才消除了这个bug。

```
    vector<int>::iterator iter_x = food_x.begin();
    vector<int>::iterator iter_y = food_y.begin();
    for ( ;iter_x != food_x.end(); )
    {
        if (*iter_x == s_pos[0].x && (*iter_y) == s_pos[0].y)       //如果重合
        {
            iter_x = food_x.erase(iter_x);                          //擦去同时记下该处
            iter_y = food_y.erase(iter_y);                          //避免出现值相同时被忽略的问题
            length++;
            score++;
        }
        else
        {
            iter_x++;
            iter_y++;
        }
    }
}
```

三、心得体会

本次实验综合性比较强，需要了解贪吃蛇的移动、吃食物、碰壁判断各种操作如何实现。算法设计方面并没有特别大的困难，但在细节处出现很多错误。例如随机数种子的选取、清屏和显示的先后顺序、在临界条件处对于游戏是否失败的判定等。设计时碰到了诸多问题，极大考验了耐力，但是最后也都解决了。主要考验了面向对象编程的实现，在仔细理解了贪吃蛇的原理之后构建这样的一个框架也不算非常复杂。但是遇到了来自各方面的问题，这次实验让我知道了自己在文件方面和面向对面编程方面还有许多需要学习的知识。同时也觉得自己在代码简洁性方面需要有比较大的改变，写出的代码太多拉杂，不够精简。看了别的同学写的版本更让我觉得差距太大，今后一定下更多的功夫来学习高程，毕竟是拿来吃饭的家伙！upup

四、源代码

以下为程序源代码：

```
//1750562 zhangbo
#define _CRT_SECURE_NO_WARNINGS
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<time.h>
#include<stdlib.h>
#include<windows.h>
#include<vector>
#include<iostream>
#include<fstream>
#include<string>
//#include<algorithm>
using namespace std;
#define IMAGE_LENGTH 400
#define IMAGE_WIDTH    400
#define MAX    200              //蛇的最大长度
#define SIZE 40                 //蛇的宽度,移动速度
#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77
IMAGE head_up, head_down, head_left, head_right, s_body, s_food, s_wall,s_food_0;

struct POS                      //坐标
{
    int x;
    int y;
    bool eat;                   //food 用
};

struct FOOD
{
    int num;            //记录现存食物数量
    POS f_pos[5];           //食物坐标
};
//墙壁的坐标
vector<int>wall_x;
vector<int>wall_y;
//由尸体产生的食物坐标
```

```
vector<int>food_x;
vector<int>food_y;

//入门版
class SNAKE_1
{
    //private:
public:
    int length;              //蛇的节数
    POS s_pos[MAX];          //蛇每节的坐标
    char ch;             //蛇的方向
    FOOD food;
    int score = 0;               //得分
    int final[2];        //记录最后一节蛇的位置
public:
    //初始化游戏界面
    void initGame()
    {
        initgraph(IMAGE_LENGTH + SIZE * 2, IMAGE_WIDTH + SIZE * 2);
                        //窗口宽度，高度

        setbkcolor(WHITE);                                  //背景颜色调至白色
        cleardevice();                                      //清屏

        //加载图片
        loadimage(&head_up, L"蛇头上.jpg", SIZE, SIZE);
        loadimage(&head_down, L"蛇头下.jpg", SIZE, SIZE);
        loadimage(&head_left, L"蛇头左.jpg", SIZE, SIZE);
        loadimage(&head_right, L"蛇头右.jpg", SIZE, SIZE);
        loadimage(&s_body, L"蛇身.jpg", SIZE, SIZE);
        loadimage(&s_food, L"西瓜.jpg", SIZE, SIZE);
        loadimage(&s_wall, L"墙.jpg", SIZE, SIZE);
        loadimage(&s_food_0, L"糖果.jpg", SIZE, SIZE);
        length = 1;                                         //蛇的节数初始化
        srand((unsigned)time(NULL));
        s_pos[0].x = rand() % 10 * SIZE + SIZE;
    //蛇的坐标
        s_pos[0].y = rand() % 10 * SIZE + SIZE;
        int direction = rand() % 4 + 1;
        switch (direction)
        {
        case 1:ch = UP; break;
        case 2:ch = DOWN; break;
        case 3:ch = LEFT; break;
```

```
        case 4:ch = RIGHT; break;
        default:break;
        }                                                   //蛇的方向

        food.num = 0;                                       //未生成食物
        for (int i = 0; i < 5; i++)
            food.f_pos[i].eat = 1;                          //食物未出现（已吃掉）
        for (int j = 0; j < IMAGE_WIDTH + SIZE * 2; j += SIZE)
        {
            putimage(0, j, &s_wall);
            putimage(IMAGE_LENGTH + SIZE, j, &s_wall);
        }

        for (int i = SIZE; i <= IMAGE_LENGTH; i += SIZE)
        {
            putimage(i, 0, &s_wall);
            putimage(i, IMAGE_WIDTH + SIZE, &s_wall);
        }
}

//加载蛇头、蛇身绘出蛇
void SnakeDisplay()
{
        //根据方向绘出蛇头
        switch (ch)
        {
        case UP:
            putimage(s_pos[0].x, s_pos[0].y, &head_up);             //三元光栅缺省
            break;
        case DOWN:
            putimage(s_pos[0].x, s_pos[0].y, &head_down);
            break;
        case LEFT:
            putimage(s_pos[0].x, s_pos[0].y, &head_left);
            break;
        case RIGHT:
            putimage(s_pos[0].x, s_pos[0].y, &head_right);
            break;
        default:
            break;
        }
        //绘制蛇身
        for (int i = length - 1; i > 0; i--)
        {
```

```
            putimage(s_pos[i].x, s_pos[i].y, &s_body);
        }
    }

    //控制蛇的移动
    void SnakeMove()
    {
        //移动时除蛇头外，后面每节蛇身均前移一位，直接对数组操作
        final[0] = s_pos[length - 1].x;
        final[1] = s_pos[length - 1].y;
        for (int i = length - 1; i > 0; i--)
        {
            s_pos[i].x = s_pos[i - 1].x;
            s_pos[i].y = s_pos[i - 1].y;
        }
        switch (ch)                    //根据方向设定蛇头运动
        {
        case UP:
            s_pos[0].y -= SIZE;
            break;
        case DOWN:
            s_pos[0].y += SIZE;
            break;
        case LEFT:
            s_pos[0].x -= SIZE;
            break;
        case RIGHT:
            s_pos[0].x += SIZE;
            break;
        default:
            break;
        }
    }

    //在蛇死亡时退回死之前位置
    void ReSnakeMove()
    {
        for (int i = 0; i < length - 1; i++)
        {
            s_pos[i].x = s_pos[i + 1].x;
            s_pos[i].y = s_pos[i + 1].y;
        }
        s_pos[length - 1].y = final[1];
        s_pos[length - 1].x = final[0];
```

```
}

//方向键控制蛇的方向改变
void SnakeDirect()
{
	switch (_getch())
	{
	case 72:
		if (ch != DOWN)			//非下即可
			ch = UP;
		break;
	case 80:				//非上
		if (ch != UP)
			ch = DOWN;
		break;
	case 75:				//非右
		if (ch != RIGHT)
			ch = LEFT;
		break;
	case 77:				//非左
		if (ch != LEFT)
			ch = RIGHT;
		break;
	default:
		break;
	}
}

//查找是否坐标与蛇身重合
bool find_in_snake(int x, int y)
{
	for (int i = 0; i < length; i++)
		if (s_pos[i].x == x && s_pos[i].y == y)
			return TRUE;
	return FALSE;
}
//生成食物坐标
virtual void CreateFOOD()
{
	//随机数种子
	int num = 0, k = 0;
	unsigned int t = 0;
	srand((unsigned)time(NULL));
	while (k == num)
```

```
        {
            t++;
            k = 0;
            //由其注意这个时间为种子会出现 OVERFLOW 出现死循环
            num = rand() % 5 + 1;                                    //随机产生 n 个
食物
            food.num = num;                                          // 记 录 食 物 个
数，为 0 时补充食物
            for (int i = 0; i < num; i++)
            {
                food.f_pos[i].x = rand() % 10 * SIZE + SIZE;          //根据地图
大小定随机数范围
                food.f_pos[i].y = rand() % 10 * SIZE + SIZE;
                food.f_pos[i].eat = 0;                               //未被吃
            }
            for (int i = 0; i < num; i++)
                if (food.f_pos[i].eat == 1)
                    k++;
        }
        return;
    }

    //判定吃到食物
    virtual void EatFOOD()
    {
        //int num = food.num;
        for (int i = 0; i < 5; i++)             //5 次判定
        {
            if (s_pos[0].x == food.f_pos[i].x
                &&s_pos[0].y == food.f_pos[i].y
                &&food.f_pos[i].eat == 0)              //坐标重合且食物存在
            {
                length++;                               //长度加 1
                food.num--;                             //食物数量减一
                food.f_pos[i].eat = 1;             //食物被吃
                score += 2;
            }
        }
    }

    //游戏结束输出
    virtual void EndGame(int flag)
    {
        if (flag == -1)
```

```
        return;
    cleardevice();
    settextcolor(LIGHTRED);
    settextstyle(50, 0, L"宋体");
    int X = IMAGE_LENGTH / 4;
    int Y = IMAGE_WIDTH / 4;
    if (flag == 0)
    {
        outtextxy(X - SIZE, Y, L"蛇咬到了自己！");

    }
    else if (flag == 1)
    {
        outtextxy(X - SIZE, Y, L"蛇咬到了墙壁！");

    }
    else if (flag == 2)
    {
        outtextxy(X - SIZE, Y, L"你可真厉害！没空间啦！");
    }
    outtextxy(X - SIZE, Y + 50, L"最终得分为:");
    char temp[50];
    _itoa(score, temp, 10);
    int n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
    wchar_t *str = new wchar_t[n];
    MultiByteToWideChar(0, 0, temp, -1, str, n);
    outtextxy(X + 300, Y + 50, str);                    //出现问题

    int k;
    ReadFile(k);
    if (score <= k)
        outtextxy(X - SIZE, Y + 100, L"请再接再厉！");
    else
    {
        outtextxy(X - SIZE, Y + 100, L"恭喜你打破纪录！");
        WriteFile(score);
    }
    char ch;
    ch = _getch();


    PrintFile();
    exit(0);
}
```

```
//判断游戏结束
virtual    int GameOver()
{
    //蛇头撞到墙壁
    if (s_pos[0].x < SIZE || s_pos[0].x >= IMAGE_LENGTH + SIZE ||
        s_pos[0].y < SIZE || s_pos[0].y >= IMAGE_WIDTH + SIZE)
    {
        return(1);
    }
    //蛇头撞到了蛇身
    for (int i = length - 1; i > 0; i--)
    {
        if (s_pos[0].x == s_pos[i].x
            &&s_pos[0].y == s_pos[i].y)
        {
            return(0);
        }
    }
    return -1;                  //未失败
}

//打印食物
virtual void DisplayFOOD()
{
    for (int i = 0; i < 5; i++)
    {
        if (food.f_pos[i].eat == 0)
            putimage(food.f_pos[i].x, food.f_pos[i].y, &s_food);
    }
    settextcolor(BLACK);
    settextstyle(15, 0, L"宋体");
    outtextxy(SIZE, IMAGE_WIDTH + SIZE, L"当前得分为：");
    char temp[50];
    _itoa(score, temp, 10);
    int n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
    wchar_t *str = new wchar_t[n];
    MultiByteToWideChar(0, 0, temp, -1, str, n);
    outtextxy(SIZE * 3 + 15, IMAGE_WIDTH + SIZE, str);                  // 出 现
问题


    outtextxy(SIZE, IMAGE_WIDTH + SIZE + 15, L"蛇的长度为：");
    //char temp[50];
```

```
	_itoa(length, temp, 10);
	n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
	str = new wchar_t[n];
	MultiByteToWideChar(0, 0, temp, -1, str, n);
	outtextxy(SIZE * 3 + 15, IMAGE_WIDTH + SIZE + 15, str);
//出现问题


	outtextxy(SIZE * 4, IMAGE_WIDTH + SIZE, L"当前版本最高分数为：");
	//char temp[50];
	//dai gao
	int k;
	ReadFile(k);
	_itoa(k, temp, 10);
	n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
	str = new wchar_t[n];
	MultiByteToWideChar(0, 0, temp, -1, str, n);
	outtextxy(SIZE * 8, IMAGE_WIDTH + SIZE, str);                    //出现问题
}

//打印文件内容
void PrintFile()
{
	ifstream in("record.txt", ios::binary);
	string x[12];
	for (int i = 0; i < 12; i += 3)
	{
		in >> x[i] >> x[i + 1] >> x[i + 2];
		cout << x[i] << " " << x[i + 1] << " " << x[i + 2] << endl;
	}
	_getch();
	in.close();
}

//取出版本历史最高数据
virtual void ReadFile(int &k)
{

	//对于 snake_1
	ifstream in("record.txt", ios::binary);
	string x[3];
	char temp;
	int i;
	in >> x[0] >> x[1];
```

```
        for (i = 0; i < 6; i++)
            in >> temp;
        in >> k;
    }

    void replaceString(string &origin, string old_value, string new_value) {
        //int k = 0;
        /* 该函数用于对 origin 全文查找出现的 old_value 并用 new_value 替换
*/
        for (string::size_type pos(0); pos != string::npos; pos += new_value.length())
{
            if (((pos = origin.find(old_value, pos)) != string::npos))
                origin.replace(pos, old_value.length(), new_value);
            else break;
        }
    }
    //更新历史最高记录
    void WriteFile(int score)
    {
        ifstream out("record.txt", ios::binary);
        //string str;

        istreambuf_iterator<char> beg(out), end;                    // 设置两个文件
指针，指向开始和结束，以 char(一字节)为步长
        string content(beg, end);                                    // 将文件全部
读入 string 字符串
        out.close();
        int k;
        char temp_k[10], temp_score[10];
        ReadFile(k);
        _itoa(k, temp_k, 10);
        _itoa(score, temp_score, 10);
        replaceString(content, temp_k, temp_score);
        ofstream fout("record.txt", ios::binary);
        fout << content;
        fout.close();
    }
    //显示墙
    virtual void DisplayWall()
    {
        for (int j = 0; j < IMAGE_WIDTH + SIZE * 2; j += SIZE)
        {
            putimage(0, j, &s_wall);
            putimage(IMAGE_LENGTH + SIZE, j, &s_wall);
```

```
    }

    for (int i = SIZE; i <= IMAGE_LENGTH; i += SIZE)
    {
        putimage(i, 0, &s_wall);
        putimage(i, IMAGE_WIDTH + SIZE, &s_wall);
    }
}

//游戏循环
virtual   void loop()
{
    CreateFOOD();
    DisplayFOOD();          //显示食物
    SnakeDisplay();
    _getch();
    while (1)
    {
        //_getch();
        while (!_kbhit())          //当按下按键退出循环
        {
            BeginBatchDraw();
            if (food.num == 0)
                CreateFOOD();
            cleardevice();              //刷新屏幕
            DisplayWall();
            DisplayFOOD();
            SnakeMove();              //蛇的移动
            EatFOOD();
            int flag = GameOver();
            if (flag != -1)
                ReSnakeMove();
            SnakeDisplay();
            EndBatchDraw();
            Sleep(200);
            EndGame(flag);
        }
        SnakeDirect();
    }
}
//等待按键
void wait_for_enter()
{
    cout << endl << "按回车键继续";
```

```
        while (_getch() != '\r')
            ;
        cout << endl << endl;
    }


};


//进阶版
class SNAKE_2 :public SNAKE_1
{
public:
    void ReadFile(int &k)
    {

        //对于 snake_2
        ifstream in("record.txt", ios::binary);
        string x[3];
        char temp;
        int i;
        in >> x[0] >> x[1] >> x[2] >> x[0] >> x[1];
        for (i = 0; i < 6; i++)
            in >> temp;
        in >> k;
    }
    bool find_in_wall(int x, int y)
    {
        //查找是否新生成的坐标与墙壁重合
        for (unsigned int i = 0; i < wall_x.size(); i++)
            if (wall_x[i] == x && wall_y[i] == y)
                return TRUE;
        return FALSE;
    }

    void ReInit(int flag) //flag= 0 1  时执行操作
    {
        //若蛇挂掉，将蛇身变成墙壁,将蛇身添加进入 wall 数组
        if (flag == 0 || flag == 1)
        {
            //记录墙壁坐标
            for (int i = 0; i < length; i++)
            {
                wall_x.push_back(s_pos[i].x);
```

```
            wall_y.push_back(s_pos[i].y);
        }


        //随机生成新蛇与新果实（避开墙壁）
        unsigned int t = 0;

        while (find_in_wall(s_pos[0].x, s_pos[0].y))          //重新生成
        {
            t++;
            srand((unsigned)time(NULL) + t);
            s_pos[0].x = rand() % 10 * SIZE + SIZE;
            s_pos[0].y = rand() % 10 * SIZE + SIZE;
        }
        //随即生成蛇的方向
        int direction = rand() % 4 + 1;
        switch (direction)
        {
        case 1:ch = UP; break;
        case 2:ch = DOWN; break;
        case 3:ch = LEFT; break;
        case 4:ch = RIGHT; break;
        default:break;
        }
        food.num = 0;
        length = 1;
    }
}
//判断游戏结束   需要修改
//游戏结束输出
void EndGame(int flag)              //-2 时执行操作
{
    if (flag != -2)
    {
        return;
    }
    BeginBatchDraw();
    cleardevice();                  //刷新屏幕
    DisplayFOOD();
    SnakeDisplay();
    DisplayWall();
    EndBatchDraw();
    Sleep(100);
    cleardevice();
```

```
    settextcolor(LIGHTRED);
    settextstyle(SIZE, 0, L"宋体");
    int X = 1 * SIZE;
    int Y = 4 * SIZE;
    outtextxy(X - SIZE, Y + 50, L"最终得分为:");
    char temp[50];
    _itoa(score, temp, 10);
    int n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
    wchar_t *str = new wchar_t[n];
    MultiByteToWideChar(0, 0, temp, -1, str, n);
    outtextxy(X + 300, Y + 50, str);                    //出现问题

    int k;
    ReadFile(k);
    if (score <= k)
        outtextxy(X - SIZE, Y + 100, L"请再接再厉！ ");
    else
    {
        outtextxy(X - SIZE, Y + 100, L"恭喜你打破纪录！ ");
        WriteFile(score);
    }
    char ch;
    ch = _getch();


    PrintFile();
    exit(0);

}

int GameOver()        //撞墙  1  撞自己  0  不撞  -1  结束游戏-2
{
    int flag = -1;
    //蛇头撞到墙壁
    if (s_pos[0].x < SIZE || s_pos[0].x >= IMAGE_LENGTH + SIZE ||
        s_pos[0].y < SIZE || s_pos[0].y >= IMAGE_WIDTH + SIZE ||
        find_in_wall(s_pos[0].x, s_pos[0].y))
    {
        flag = 1;
    }
    //蛇头撞到了蛇身
    else
    {
        for (int i = length - 1; i > 0; i--)
```

```
            {
                if (s_pos[0].x == s_pos[i].x
                    &&s_pos[0].y == s_pos[i].y)
                {
                    flag = 0;
                    break;
                }
            }
        }
        unsigned int temp = (IMAGE_LENGTH / SIZE) * (IMAGE_WIDTH / SIZE);
        if (temp == wall_x.size())                                          //当没有空间产生果实和蛇头时结束游戏
            flag = -2;
        return flag;
    }

    void CreateFOOD()
    {
        //随机数种子
        int num = 0, k = 0, t = 0;
        while (k == num)
        {
            t++;
            k = 0;
            srand((unsigned)time(NULL) + t);                    //由其注意这个时间为种子会出现 OVERFLOW 出现死循环
            num = rand() % 5 + 1;                               //随机产生 n 个食物
            food.num = num;                                     //记录食物个数，为 0 时补充食物
            for (int i = 0; i < num; i++)
            {
                food.f_pos[i].x = rand() % 10 * SIZE + SIZE;            //根据地图大小定随机数范围
                food.f_pos[i].y = rand() % 10 * SIZE + SIZE;
                food.f_pos[i].eat = 0;                          //未被吃
                if (find_in_wall(food.f_pos[i].x, food.f_pos[i].y))         //记录下与墙重叠的果实（包括重复的果实）
                {
                    food.f_pos[i].eat = 1;
                }
            }
            for (int i = 0; i < num; i++)
```

```
                if (food.f_pos[i].eat == 1)
                    k++;
        }
        return;
    }

    void DisplayWall()
    {
        for (int j = 0; j < IMAGE_WIDTH + SIZE * 2; j += SIZE)
        {
            putimage(0, j, &s_wall);
            putimage(IMAGE_LENGTH + SIZE, j, &s_wall);
        }

        for (int i = SIZE; i <= IMAGE_LENGTH; i += SIZE)
        {
            putimage(i, 0, &s_wall);
            putimage(i, IMAGE_WIDTH + SIZE, &s_wall);
        }
        //记录每一次死亡后蛇身的坐标用于显示墙
        for (unsigned int i = 0; i < wall_x.size(); i++)
        {
            putimage(wall_x[i], wall_y[i], &s_wall);                    //加载墙的图片
        }


        settextcolor(BLACK);
        settextstyle(15, 0, L"宋体");
        outtextxy(SIZE, IMAGE_WIDTH + SIZE, L"当前得分为：");
        char temp[50];
        _itoa(score, temp, 10);
        int n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
        wchar_t *str = new wchar_t[n];
        MultiByteToWideChar(0, 0, temp, -1, str, n);
        outtextxy(SIZE * 3 + 15, IMAGE_WIDTH + SIZE, str);                  // 出 现
问题


        outtextxy(SIZE, IMAGE_WIDTH + SIZE + 15, L"蛇的长度为：");
        //char temp[50];
        _itoa(length, temp, 10);
        n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
        str = new wchar_t[n];
        MultiByteToWideChar(0, 0, temp, -1, str, n);
```

```
    outtextxy(SIZE * 3 + 15, IMAGE_WIDTH + SIZE + 15, str);
//出现问题

    outtextxy(SIZE * 4, IMAGE_WIDTH + SIZE, L"当前版本最高分数为：");
    //char temp[50];
    //dai gao
    int k;
    ReadFile(k);
    _itoa(k, temp, 10);
    n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
    str = new wchar_t[n];
    MultiByteToWideChar(0, 0, temp, -1, str, n);
    outtextxy(SIZE * 8, IMAGE_WIDTH + SIZE, str);                    //出现问题
}

void loop()
{
    CreateFOOD();
    DisplayFOOD();            //显示食物
    SnakeDisplay();
    _getch();
    //int k = 0;
    while (1)
    {
        //_getch();
        while (!_kbhit())            //当按下按键退出循环
        {
            BeginBatchDraw();
            if (food.num == 0)
                CreateFOOD();
            cleardevice();                //刷新屏幕
            DisplayFOOD();
            SnakeMove();                //蛇的移动

            //k++;
            EatFOOD();
            int flag = GameOver();
            if (flag != -1)
                ReSnakeMove();
            ReInit(flag);
            SnakeDisplay();
            DisplayWall();
            EndBatchDraw();
```

```
                Sleep(200);
                EndGame(flag);
            }
            SnakeDirect();
        }

    }

};
```

//高级版
//思路：从进阶版进化而来，在挂掉的时候记录下蛇身坐标然后生成食物当食物再次被吃完的时候再次随即生成食物。不断检测撞墙次数，到达五的时候结束调用 EndGame

```
class SNAKE_3 :public SNAKE_1
{
public:
    void ReadFile(int &k)
    {

        //对于 snake_1
        ifstream in("record.txt", ios::binary);
        string x[3];
        char temp;
        int i;
        in >> x[0] >> x[1] >> x[2]
            >> x[0] >> x[1] >> x[2]
            >> x[0] >> x[1];
        for (i = 0; i < 6; i++)
            in >> temp;
        in >> k;
    }
    bool find_in_food(int x, int y)
    {
        //查找是否新生成的坐标与墙壁重合
        for (unsigned int i = 0; i < food_x.size(); i++)
            if (food_x[i] == x && food_y[i] == y)
                return TRUE;
        return FALSE;
    }
```

```
void ReInit(int flag)
{
    //若蛇挂掉，将蛇身变成墙壁,将蛇身添加进入 wall 数组
    if (flag == 0 || flag == 1)
    {
        //记录新生成果子坐标
        for (int i = 0; i < length; i++)
        {
            food_x.push_back(s_pos[i].x);
            food_y.push_back(s_pos[i].y);
        }
        //随机生成新蛇与新果实（避开墙壁）
        //unsigned int t = 0;
        srand((unsigned)time(NULL));
        while (find_in_food(s_pos[0].x, s_pos[0].y))          //重新生成
        {


            s_pos[0].x = rand() % 10 * SIZE + SIZE;
            s_pos[0].y = rand() % 10 * SIZE + SIZE;
        }
        //随即生成蛇的方向

        //随即生成蛇的方向
        int direction = rand() % 4 + 1;
        switch (direction)
        {
        case 1:ch = UP; break;
        case 2:ch = DOWN; break;
        case 3:ch = LEFT; break;
        case 4:ch = RIGHT; break;
        default:break;
        }
        food.num = 0;
        length = 1;
    }
}

void CreateFOOD()
{
    //随机数种子
    int num = 0, k = 1, t = 0;
    srand((unsigned)time(NULL) );
    while (k != 0)
```

```
        {
            //t++;
            k = 0;
                                        // 由 其 注 意 这 个 时 间 为 种 子 会 出 现
OVERFLOW 出现死循环
            num = rand() % 5 + 1;                                   //随机产生 n 个
食物
            food.num = num;                                         // 记 录 食 物 个
数，为 0 时补充食物
            for (int i = 0; i < num; i++)
            {
                food.f_pos[i].x = rand() % 10 * SIZE + SIZE;            //根据地图
大小定随机数范围
                food.f_pos[i].y = rand() % 10 * SIZE + SIZE;
                food.f_pos[i].eat = 0;                                  //未被吃
                if (find_in_food(food.f_pos[i].x, food.f_pos[i].y))         //记录下与
尸体重叠的果实（包括重复的果实）
                {
                    food.f_pos[i].eat = 1;
                }
            }
            for (int i = 0; i < num; i++)
                if (food.f_pos[i].eat == 1)
                    k++;
        }
        return;
    }
    void EatFOOD()
    {
        for (int i = 0; i < 5; i++)             //5 次判定
        {
            if (s_pos[0].x == food.f_pos[i].x
                &&s_pos[0].y == food.f_pos[i].y
                &&food.f_pos[i].eat == 0)               //坐标重合且食物存在
            {
                length++;                               //长度加 1
                food.num--;                             //食物数量减一
                food.f_pos[i].eat = 1;              //食物被吃
                score += 2;
            }
        }
        /*
        for (unsigned int i = 0; i < food_x.size(); i++)
        {
```

```
            if (s_pos[0].x == food_x[i]
                && s_pos[0].y == food_y[i])
            {
                length++;                               //长度加 1
                score += 1;
                food_x.erase(food_x.begin()+i);
                food_y.erase(food_y.begin()+i);
            }
        }*/
        vector<int>::iterator iter_x = food_x.begin();
        vector<int>::iterator iter_y = food_y.begin();
        for ( ;iter_x != food_x.end(); )
        {
            if (*iter_x == s_pos[0].x && (*iter_y) == s_pos[0].y)          //如果重合
            {
                iter_x = food_x.erase(iter_x);                             //擦去同时
记下该处
                iter_y = food_y.erase(iter_y);                             //避免出现
值相同时被忽略的问题
                length++;
                score++;
            }
            else
            {
                iter_x++;
                iter_y++;
            }
        }
    }
    //打印食物
    void DisplayFOOD()
    {
        for (int i = 0; i < 5; i++)
        {
            if (food.f_pos[i].eat == 0)
                putimage(food.f_pos[i].x, food.f_pos[i].y, &s_food);
        }
        for (unsigned int i = 0; i < food_x.size(); i++)
        {
            putimage(food_x[i], food_y[i], &s_food_0);
        }
        settextcolor(BLACK);
        settextstyle(15, 0, L"宋体");
        outtextxy(SIZE, IMAGE_WIDTH + SIZE, L"当前得分为：");
```

```
        char temp[50];
        _itoa(score, temp, 10);
        int n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
        wchar_t *str = new wchar_t[n];
        MultiByteToWideChar(0, 0, temp, -1, str, n);
        outtextxy(SIZE * 3 + 15, IMAGE_WIDTH + SIZE, str);                  // 出 现
问题


        outtextxy(SIZE, IMAGE_WIDTH + SIZE + 15, L"蛇的长度为：");
        //char temp[50];
        _itoa(length, temp, 10);
        n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
        str = new wchar_t[n];
        MultiByteToWideChar(0, 0, temp, -1, str, n);
        outtextxy(SIZE * 3 + 15, IMAGE_WIDTH + SIZE + 15, str);
    //出现问题


        outtextxy(SIZE * 4, IMAGE_WIDTH + SIZE, L"当前版本最高分数为：");
        //char temp[50];
        //dai gao
        int k;
        ReadFile(k);
        _itoa(k, temp, 10);
        n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
        str = new wchar_t[n];
        MultiByteToWideChar(0, 0, temp, -1, str, n);
        outtextxy(SIZE * 8, IMAGE_WIDTH + SIZE, str);                  //出现问题
    }
    int Space()
    {
        unsigned int temp = (IMAGE_LENGTH / SIZE) * (IMAGE_WIDTH / SIZE)
- 2;
        if (temp == food_x.size())                                                  //
当没有空间产生果实和蛇头时结束游戏
            return 0;
        return 1;
    }
    void EndGame(int Death)
    {
        if (Death != 5 && Space())              //未达到失败次数且仍有空间
        {
            return;
```

```
    }
    cleardevice();
    settextcolor(LIGHTRED);
    settextstyle(SIZE, 0, L"宋体");
    int X = 1 * SIZE;
    int Y = 4 * SIZE;


    if (!Space())              //无多余空间
    {
        outtextxy(X, Y, L"你也忒猛了！没空间了！");
    }
    else
    {
        outtextxy(X, Y, L"You Suck！撞了五次墙！");
    }
    outtextxy(X, Y + SIZE, L"最终得分为:");
    char temp[50];
    _itoa(score, temp, 10);
    int n = MultiByteToWideChar(0, 0, temp, -1, NULL, 0);
    wchar_t *str = new wchar_t[n];
    MultiByteToWideChar(0, 0, temp, -1, str, n);
    outtextxy(X + 6 * SIZE, Y + SIZE, str);                    //出现问题

    int k;
    ReadFile(k);
    if (score <= k)
        outtextxy(X - SIZE, Y + 100, L"请再接再厉！");
    else
    {
        outtextxy(X - SIZE, Y + 100, L"恭喜你打破纪录！");
        WriteFile(score);
    }
    char ch;
    ch = _getch();
    PrintFile();
    exit(0);

}
void loop()
{
    CreateFOOD();
    DisplayFOOD();             //显示食物
    SnakeDisplay();
```

```
        _getch();
        int Death = 0;
        while (1)
        {
            //_getch();
            while (!_kbhit())            //当按下按键退出循环
            {
                BeginBatchDraw();
                if (food.num == 0)
                    CreateFOOD();
                cleardevice();              //刷新屏幕
                DisplayWall();
                SnakeMove();                //蛇的移动
                //k++;
                EatFOOD();
                int flag = GameOver();
                if (flag != -1)             //撞到墙或者自己
                {
                    Death++;
                    ReSnakeMove();
                }
                ReInit(flag);
                SnakeDisplay();
                DisplayFOOD();              //有问题
                EndBatchDraw();
                Sleep(200);
                EndGame(Death);
            }
            SnakeDirect();
        }

    }

};
//菜单函数
void Menu()
{
    SetConsoleTitleA("游戏：2048");                                    //设置控制台
标题
    HANDLE handle_out = GetStdHandle(STD_OUTPUT_HANDLE);          //获
取标准输入设备句柄
    SetConsoleTextAttribute(handle_out, FOREGROUND_GREEN);              //设
置打印颜色为绿色
    cout << "                    ----------------------------------------\n";
```

```
    cout                                        <<                                  "
**********************************************\n";
    SetConsoleTextAttribute(handle_out,          FOREGROUND_RED          |
FOREGROUND_GREEN);          //设置打印颜色为黄色,用三原色搭配（自行查阅）
    cout << "                                        菜单\n";
    cout << "                                    a.入门版\n";
    cout << "                                    b.进阶版\n";
    cout << "                                    c.高级版\n";
    cout << "                                    d.游戏规则\n";
    cout << "                                    e.退出游戏\n";
    SetConsoleTextAttribute(handle_out,  FOREGROUND_GREEN);          //设
置打印颜色为绿色
    cout                                        <<                                  "
**********************************************\n";
    cout << "                ----------------------------------------\n";
    cout << "\n 请输入你的选择(a-e):";
}

void wait_for_enter()
{
    cout << endl << "按回车键继续";
    while (_getch() != '\r')
        ;
    cout << endl << endl;
}

void Help()
{
    system("cls");
    HANDLE  handle_out  =  GetStdHandle(STD_OUTPUT_HANDLE);          //获
取标准输入设备句柄
    CONSOLE_SCREEN_BUFFER_INFO csbi;                                  //定
义窗口缓冲区信息结构体
    GetConsoleScreenBufferInfo(handle_out,  &csbi);          //获得窗口缓冲
区信息
    SetConsoleTextAttribute(handle_out,          FOREGROUND_RED          |
FOREGROUND_GREEN);          //设置打印颜色为黄色
    cout << "                ----------------------------------------------------\n";
    cout                                        <<                                  "
**********************************************************\n\n";
    cout << "                操作说明：\n                ↓：下    ←：左    ↑：上
→：右   ESC 键：退出\n\n";
    cout  << "                游戏介绍：\n                每次选择一个方向作为蛇
```

```
的移动方向\n";
    cout << "                进阶版为：蛇死后变成墙壁，再随机产生新的蛇和食
物，直到空间不足为止\n";
    cout << "                高级版为：蛇死后变成食物，再随机产生新的蛇和食
物，直到撞墙五次或空  \n";
    cout << "                间不足为止\n\n";
    cout                                    <<                                  "
**********************************************************\n";
    cout << "                ----------------------------------------------------\n\n";
    wait_for_enter();
}
int main()
{

    char choice, ch;

    while (1)
    {
        system("CLS");
        Menu(); //调用菜单显示函数
        cin >> choice;
        if (choice == 'e') //选择退出
        {
            cout << "\n 确定退出吗?" << endl;
            cin >> ch;
            if (ch == 'y' || ch == 'Y')
            {
                cout << "那好吧..886";
                break;
            }
            else
                continue;
        }
        SNAKE_1 snake1;
        SNAKE_2 snake2;
        SNAKE_3 snake3;
        switch (choice)
        {
        case 'a': snake1.initGame(); snake1.loop(); break;
        case 'b': snake2.initGame(); snake2.loop(); break;
        case 'c':snake3.initGame(); snake3.loop(); break;
        case 'd':Help();
        default:
            cout << "\n 输入错误，请重新输入" << endl;
```

```
                wait_for_enter();
            }
        }
        return 0;
}
```