**同济大学计算机系**

**高级程序设计大作业报告**

**II**

****

**学 号 1950000**

**姓 名 一二三**

**专 业 计算机类**

**班 级 18级计算机二班**

**授课老师 陈宇飞**

**完成日期 2019/5/20**

贪吃蛇大作业

一、实验内容

本次实验的主要内容为使用C++编程语言，使用类的相关知识，构建出一个贪吃蛇小游戏，该小游戏应当具备有三种基础功能，并可根据OJ的提示，添加更多的加分项。

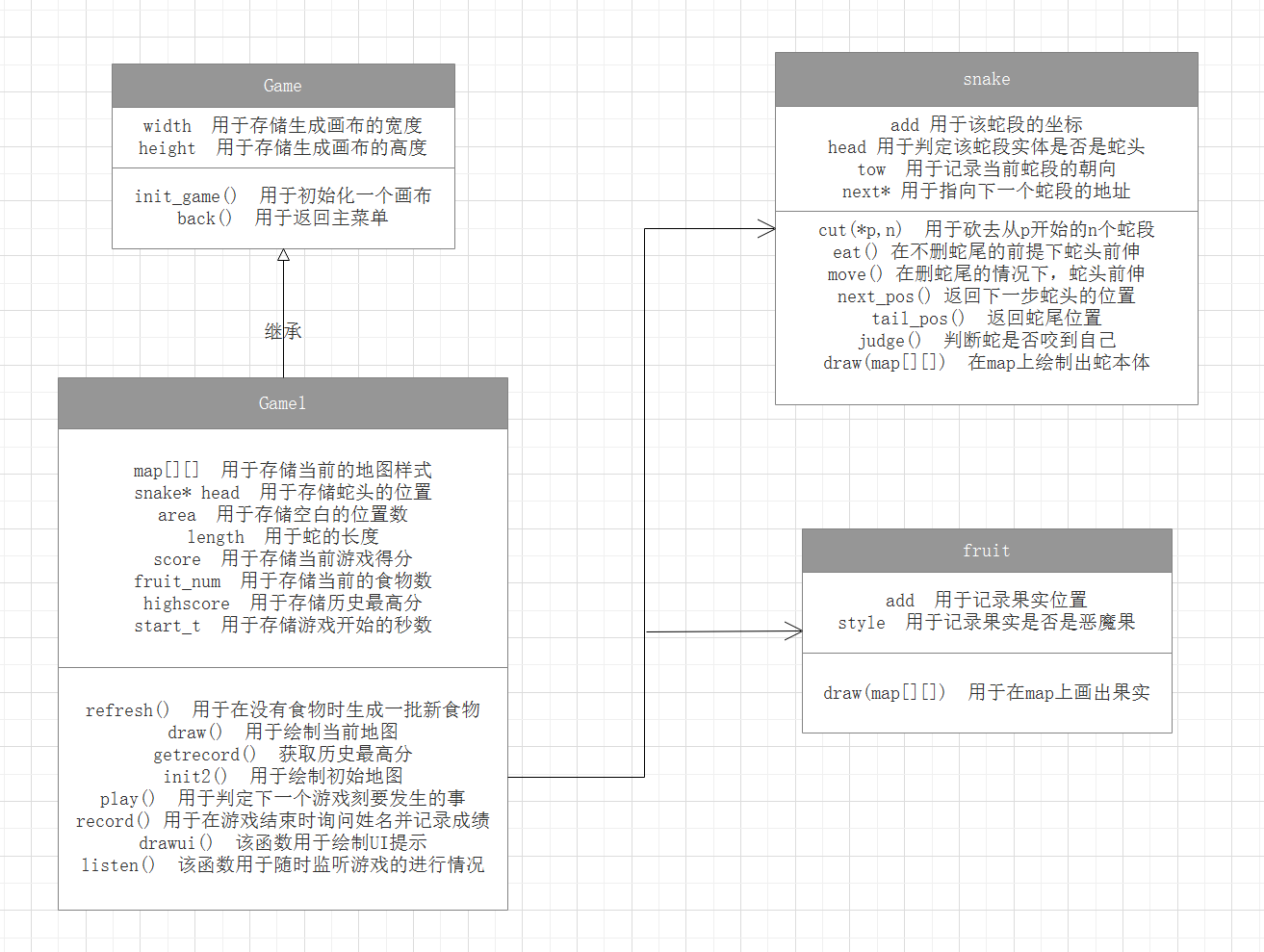
**二、设计思路与功能描述**

**2.1 设计思路说明**

**2.1.1 贪吃蛇游戏设计思路说明**

由于本次实验希望采用面向对象的思想来进行程序的编写，而贪吃蛇游戏本身作为程序的出题，故应当先绘制出该程序的类图，再在该类图的基础上进行进一步程序的编写。

由于不同玩法的贪吃蛇之间略有不同，故在此我们首先对“入门版”的贪吃蛇类图进行说明。



在本程序中，一场贪吃蛇游戏的进行应当是基于Game的某一子类展开的，在“入门版”中，这个子类为Game1。

通过生成一个Game1实体，依次调用该实体的init\_game()方法和init2()方法，可以在画布上画出初始的地图，并生成一个snake实体，同时藉由refresh()方式生成一批fruit实体。

在初始化结束后，游戏可以正式开始，在游戏进行的过程中，通过listen()方法对键盘进行实时监听，一旦用户执行了某个操作，就将操作值进行记录，在下一次调用play()方法时，判断该值对下一步结果的影响，最后调用draw()方法在画布上更新界面。

而当play()方法发现下一步蛇会死亡时，会弹出“游戏结束”字体，最后调用back()方法，回到初始菜单。

基于上述类图与大体流程描述，一下对各个类进行逐一详细描述：

***①Game类***

该类是所有游戏类的父类，其主要功能是为各个子类提供一个统一的初始与结束模板，使得各个游戏类的调用、结束显得较为统一。

该类中的属性有：

1） width

int类型，默认值为800。该属性是新建画布时，所建立的画布宽度，在本次程序设计中，这一初始值没有进行过改变。

2） length

int类型，默认值为600。该属性与width类似，是初始新建立画布的高度，在本次程序设计中，也未进行过改变。

该类中的方法有：

1） init\_game()

void返回类型。该方法是在初始化地图前必须进行的调用，作用是生成指定规格的画布。

2） back()

void返回类型。该方法是在游戏结束（包括按Q键退出、游戏失败录入姓名后退出等）后，生成了一个Menu类实体，重新回到主页面。

3）init2()、listen()

Game类作为一个大的父类，还额外定义了init2()和listen()这两个虚函数，用于在编写子类时记得实现这两个重要的方法。

***②Game1类***

该类是Game类的子类，也是其他Game类用以修改功能的基础，如何利用一个Game1类实体进行一场游戏已在前文叙述得较为清晰，故不在此赘述，这里将重点对Game1类中的众多属性和方法进行详细说明。

该类中的属性有：

1） map

char型二维数组，规格为[40][30]。这一大小对应了整个贪吃蛇游戏界面的40×30的规格，这一属性也是Game1类中众多方法实现的基础。map数组中存放的字符是游戏界面中，对应位置方格.jpg图片文件名的第二个字符。例如'0'表示这里是空草坪，'7'表示这里是食物等，该数组中可存放的字符会随着后续功能的拓展而发生变化。

2）mes

char型字符，默认值为'u'。mes用于记录用户上一次对贪吃蛇发出的控制指令，其中'u'表示向上，'d'表示向下，'l'表示向左，'r'表示向右。

3）head

snake型指针。由于贪吃蛇在本游戏中是以链表的形式存放的，即将每一个蛇段看作一个实体，所以为了能够对蛇进行控制，就需要掌握链表的表头——即蛇头的相关信息。

4） area

int类型，初始值为28×38。该值用于记录当前地图上还能放置水果的位置总数，以便随机生成新水果，并在后续功能中辅助判定地图是否已满。

5） length

int类型，初始值为3。该值用于记录当前蛇的长度，以便在UI界面进行显示。

6）life

int类型，初始值为1。该值用于记录蛇的生命数并在UI上显示，在Game1的游戏中这个值没有意义，但在后续加入的其他功能中，该值可以记录蛇的最多死亡次数。同时，当life变为0时，默认是游戏结束的标志，在这种情况下，play()方法将被禁用。

7） score

int类型，初始值为0。该值用于记录当前的得分并在UI上显示，同时该值也会在排行榜记录时被录入文件。

8） highscore

int类型。该值在init2()方法下的getrecord()方法中被赋值，用于记录当前所完游戏种类在排行榜中的最高分，当score高于highscore时，highscore在么一个游戏刻被刷新为score值。

该类中的方法有：

1） refresh()

void返回类型。该方法在初始化地图与fruit\_num为0时被调用，该方法可以在当前为空草坪（即map值为'0'）的位置随机生成1~5个fruit实体，并同时刷新fruit\_num值与area值。

2）draw()

void返回类型。该方法在每一次操作执行结束后进行调用，该方法会逐一遍历map数组，使用easyX自带的图形辅助函数将每一个数组元素所代表的图片样式绘制在画布的指定位置。

3） getrecord()

void返回类型。该方法在地图初始化时会被调用。该方法会通过遍历的方式逐行从record.txt中读取游戏记录，在游戏版本编号和当前游戏一致的记录中找寻得分最高的成绩赋值给highscore属性。

4） init2()

void返回类型。该方法是重写了父类Game的方法所得，会在新建好画布之后进行调用。该方法会对map属性进行赋值，将整个地图的边界都赋值为“硬墙”（map编号为'8'），同时调用refresh()方法，生成一批食物，并将食物所在位置赋值为“食物”（map编号为'7'）。

同时该方法还会新建一个snake实体，其长度为3，蛇头在[20][14]位置，蛇尾在[20][16]位置，并在map中对对应位置赋值。在完成了对map的初始化赋值后，调用draw()方法，绘制初始界面。

5） play()

bool返回类型。该方法用于处理在下一个游戏刻时游戏发生变化的逻辑，其过程是先通过snake类的next\_pos()方法，获取到下一个游戏刻蛇头的位置，然后对该位置进行判定。

如果该位置是蛇尾以外的蛇身、硬墙，则游戏结束，弹出提示语，同时将life归零，禁用play()方法，恒返回false；若该位置是果实，则调用该snake的eat()方法，同时让length属性自增1，返回true；若该位置是空草坪，则 调用snake的move()方法，返回true；

返回值适用于标识游戏是否能够进行下去的判断依据。

6） record()

void返回类型。该方法用于在游戏结束或用户自行退出时调用，弹出提示语要求用户输入姓名，同时对用户输入的内容进行实时显示。当用户输入完毕后，在record.txt的末尾追加本次游戏的版本编号、用户名、成绩。

7） drawui()

void返回类型。该方法用于在每一次draw方法调用后，紧接着调用该方法绘制页面下部的所有UI提示交互。其实现逻辑是使用int2char函数，将蛇的长度、生命数、分数等存储为int类型的信息与提示语一并合成为一个char数组。然后再在指定的位置输出这些提示字符串。

8） listen()

void返回类型。该方式重写了父类Game的listen方法。该方法可以实时对键盘内容进行实时监听，当有按键按下时，采用\_getch()函数获取按键值，并根据按键值对mes属性赋值。

当没有按键按下时，则按照系统的内部时钟，每隔一秒调用一次play()方法与draw()方法，重新绘制一遍界面。值得一提的是，当游戏结束后，play()被禁止，则同时停止调用draw()方法，只调用record方法，用于记录用户姓名。

***③snake类***

由于在本程序中采用链表的方式来存储贪吃蛇中的蛇，所以这里的snake类的实体其实是每一个蛇段，每个蛇段之间采用指针的方式前后相连，形成一条完整的蛇。以下，对sanke类中的属性和方法进行详细说明。

该类中的属性有：

1） add

pos类型。add用于存储当前蛇段在地图中的横纵坐标，便于后续绘制、判断等操作。

2） next

snake\*类型，默认值为NULL。next即蛇链表的后继指针，用于指向当前蛇段的下一个蛇段。蛇尾的next值为NULL。

3） head

bool类型，默认值为false。head属性用作一个标志位，用于表示当前这个蛇段是否是蛇头，如果是蛇头，则为true；其余均为false。

4）tow

char类型，默认值为'u'。tow用于记录该蛇段蛇头方向的指向，该值与其后继蛇段的tow值可以一同决定当前蛇段的形状、扭曲方向。

5）length

int类型，默认值为0。该值用于记录当前蛇的长度。值得一提的是，只有蛇头的该属性才是正确的，蛇段的length属性值无意义。

该类中的方法有：

1） cut(snake \*p, int n)

snake\*返回类型。该方法用于将从\*p所指向的蛇段开始，向后数连续n个蛇段进行删去，主要用于配合“高级版”的游戏实现。该方法是基于递归实现的，当n不为0时，则删除当前节点，并递归cut(p->next,n-1)。同时，为了便于操作，该方法还将返回删减后的头结点，用于替换原有的snake。

2） eat(char tow)

snake\*返回类型。该方法用于在判定到下一个游戏刻蛇头到达食物时调用，其方法是在tow方向上新建一个snake实体，并将该实体与原来的头部相连接，取而代之为新的头部，并将该头部返回为新的\*head。

3）move(char tow)

snake\*返回类型。该方法用于在判定到下一个游戏刻蛇头到达位置为草坪时调用。其流程为先删去蛇尾，并将倒数第二个蛇段的next置空，之后再调用eat方法，在tow方向上生成一个新的蛇头并返回该蛇头。

4）next\_pos(char tow)

pos返回类型。该方法返回如果沿tow方向走，蛇头在下一个游戏刻将要到达的位置，并返回该位置。该方法用于在play方法中辅助进行碰撞的判定。

5）tail\_pos()

pos返回类型。该方法将返回当前蛇的蛇尾坐标。

6） judge(snake\* s,pos p)

bool返回类型。该方法用于判定位置p是否在是蛇身上的某点，其实现采用了递归的方式来实现，遍历了蛇身上的每一个蛇段，依次判断该点是否在蛇的身上。

7） draw(char a[][])

void返回类型。该方法用于在a（即Game子类中的map属性）中在对应位置赋值上对应蛇段的形状标号。例如下拐左、右拐上等扭曲状态，便于Game子类在后续操作中便于画图。同时，为了满足后续idea实现过程中的需求，本程序在编写时还对draw方法根据a的规格进行了重载。

***④fruit类***

fruit类即在游戏过程中不断出现的食物抽象而成的类，这个类的方法与属性都较少，这与食物本身性质不多有很大的关系。其属性与方法的详细介绍如下：

该类中的属性有：

1） add

pos类型。add用于该食物在地图中的横纵坐标，便于后续绘制使用。

2） style

char 类型，默认为'g'。style用于记录当前食物的性质，当为'g'时说明该食物是正常水果；当为'b'时说明该食物是恶魔果。

该类中的方法仅有：

draw(char a[][])

void返回类型。该方法用于在a（即Game子类中的map属性）中在对应的位置根据该fruit实体的style赋上不同的值，以便在后期绘制过程中能够展示出该食物。

**2.1.2 其他Game子类思路简要说明**

①Game2

Game2实现的是OJ提到的“进阶版”内容，该内容要求实现蛇死去后身体变成边界，再生成一条新的蛇，直至地图被撑满。

该过程的实现主要在于修改判定到蛇死去的逻辑，加写一个stone()方法，在蛇死去后将其身体所在的所有位置赋值为“硬墙”。

②Game3

Game3实现的是“高级版”内容，该内容要求蛇死后，其身体变成实物，同时再生成一批新食物，直到死亡次数大于5。

该过程的实现与Game2的实现相仿，加写一个juice()方法，在蛇死后将其身体全部赋值为“食物”。同时还需要对死亡后的逻辑加以修改，使得蛇死后让其life值自减，life自减为0时游戏结束。

③Game4

Game4实现的是idea1的要求，即额外加入了“软墙”的概念。当蛇碰到软墙后，长度减2；碰到“硬墙”后，长度减半。

该过程的实现首先要对init()方法进行修改，在地图上加入软墙元素；其次就是要额外加入软墙的素材包，同时要修改碰撞判定之后的逻辑，即不再是碰到墙就游戏结束，而是要到蛇的长度小于一定值才结束。

④Game5

Game5实现的是idea2的要求，即额外加入“加速区”和“减速区”。

该过程的实现首先是init()方法的重写与素材包的更新，同时要对listen()方法进行修改，当蛇头在加速区时，将每隔1秒刷新一次更改为每隔0.5秒刷新一次；而当蛇头在减速区时，更改为每隔2秒刷新一次。

⑤Game6

Game6实现的是idea3的要求，即额外引入“体能槽”的概念，每次蛇的运动都会消耗体能，当体能清零时游戏结束。

该过程的实现首先要修改life属性的含义，将life属性视作体能，初始时为100，每次move()都会消耗体能；同时还要修改drawui()方法，将原来的字符串更改为能量条的形式；最后还要修改碰撞判定的逻辑，当life小于等于0时游戏结束。

⑥Game7

Game7实现的是idea4的要求，即额外引入“恶魔果”的概念，当蛇碰到恶魔果时直接死亡，游戏结束。

该过程的实现首先是要修改refresh()方法，在每次生成好果子的同时生成恶魔果，其次是要更新恶魔果的素材；同时要更改碰撞的判定逻辑，当碰到恶魔果时，游戏直接结束。

⑦Game8

Game8实现的是idea6的内容，即扩大地图的边界，每次只绘制蛇头在正中心的部分。（前提是不能绘制到边界以外的内容）

该过程的实现主要是要重新修改map[][]的属性，同时对snake类、fruit类中的相关函数进行重载，使之可以接受更大规格的map值传入。同时还要对draw()方法进行修改，确保能够满足题目的要求。

⑧Game9

Game9实现的是AI对战的内容，即添加了另一条蛇，由AI操控，当有一条蛇死亡时，另一方获胜。

该过程的实现是新添加了snake2类与ai类，其中sanke2类的定义与sanke类的定义相仿，只是对蛇段材质的内容进行了更新与修改；而ai类的实现将在后续进行详细说明。同时，还对listen()方法进行了修改，加写了aplay()等方法，加设了ames等属性。使得该Game类能够适应AI对战的相应需求。

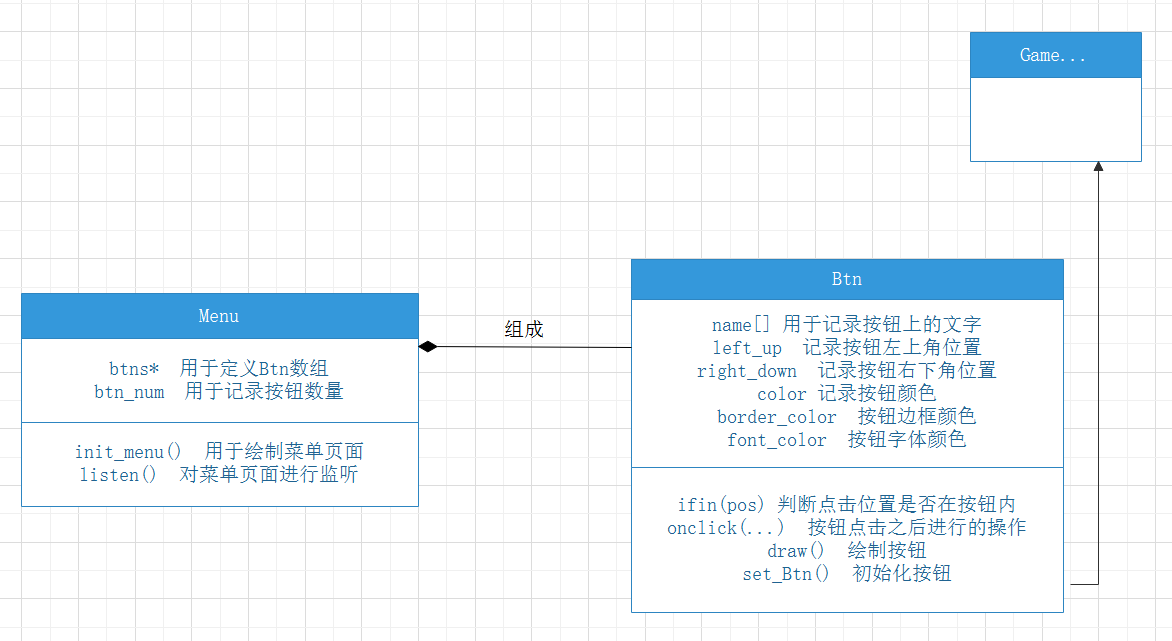
⑨Game10

Game10实现的是双人对战的内容，即可以实现对两条蛇的同时控制，当一条蛇死亡后，另一条蛇获胜。

该过程的实现主要是基于Game9，将其中的ai对象删除后，修改listen()方法使之能够同时监听两种键盘，实现对mes和ames属性的同时监控，从而实现同时控制的效果。

**2.1.3 菜单设计思路说明**

在本次程序设计中，菜单的设计也是基于面向对象的思想进行编写的，其类图为：



通过该类图不难看出其逻辑。在程序开始运行时，首先实例化一个Menu对象，该对象中包含12个Btn对象，这些Btn对象均会在被鼠标点击时生成对应的Game子类对象，由此可开始游戏。

由于Menu与Btn类的实现较为简单，且没有特别的亮点，故不在此赘述其具体属性与方法。

**2.1.4 AI设计**

在我初次拿到本次题目时，我首先想到的是从头部开始，对所有“草坪”单元格采用广度优先（BFS）的遍历算法，找到距离其最近的果实，而该路径的第一步就是AI此时会采取的策略。

在程序中3293~3367行注释掉的代码即为AI采用BFS的算法。但使用该AI时，我发现其一定不会作出错误的决定，导致玩家很难战胜这个AI。所以我将AI的策略“傻瓜化”，使得玩家可以获得一定的游戏体验。

其策略为，首先判断当前方向是否有食物，如果有则继续前进，没有则搜索左右两侧是否有食物，如果有则立刻拐弯。

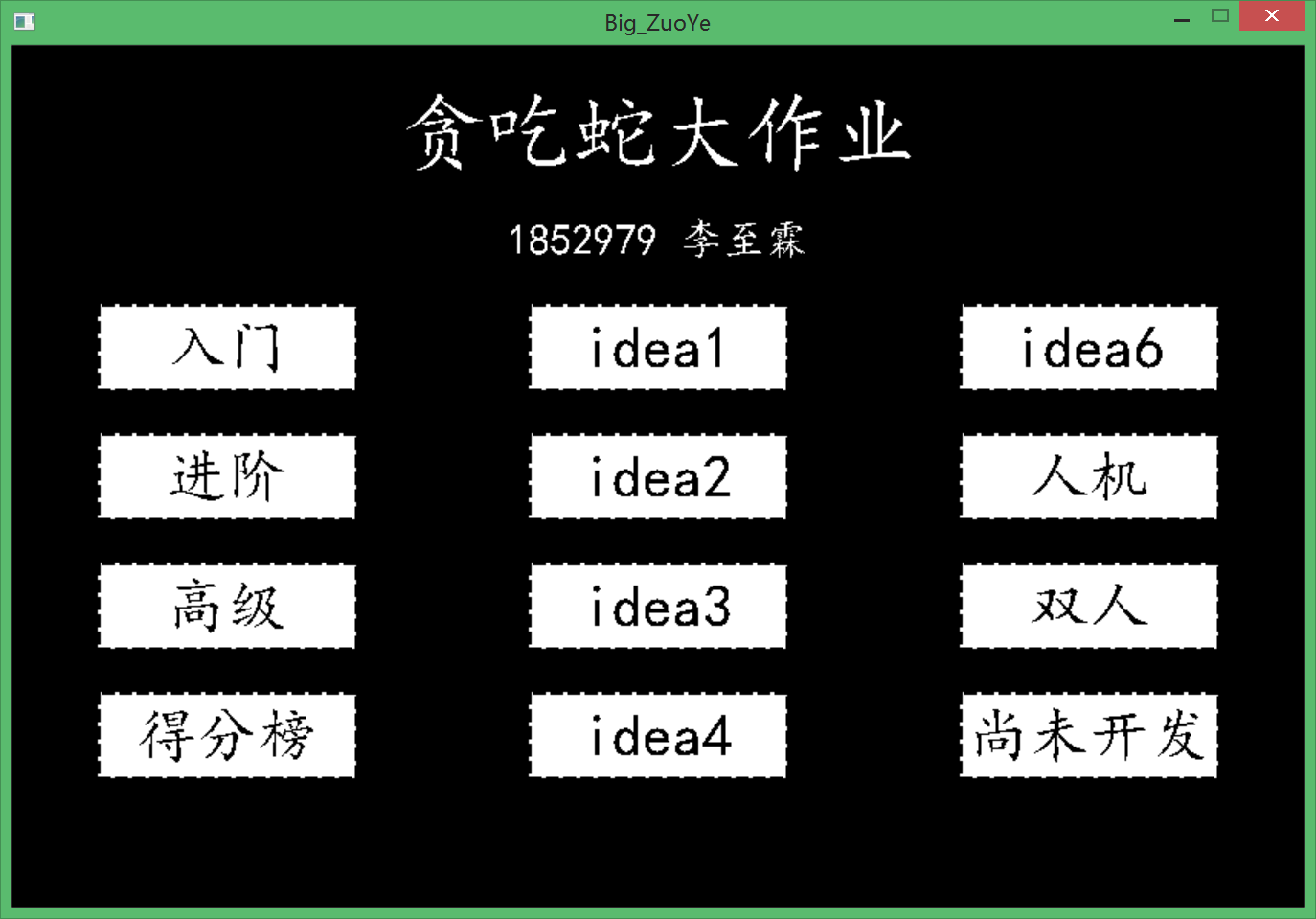
而当三个方向都没有食物时，会随机在地图内部生成一个“标记点”，然后让贪吃蛇作出尽可能向“标记点”的决策。

基于这样的AI设计，玩家可以看到AI作出很傻的决定，也能在一定程度下战胜AI，获得游戏快感。

**2.2 游戏功能描述**

**2.2.1 菜单功能展示**

该游戏的菜单界面由12个按钮与对应的提示语组成，鼠标点击对应的按钮即可进入相应的游戏：



**2.2.2 入门游戏展示**

在入门游戏中，玩家通过控制WASD四个键即可控制蛇的移动，按下Q键退出游戏，下方的字符串作为UI提示：



当按下Q键时，结束游戏，并弹出姓名输入框：



碰壁或是失败后，弹出“游戏结束”提示语，同时弹出姓名输入框。



**2.2.3 进阶版游戏展示**

贪吃蛇死后，其尸体变为边界：

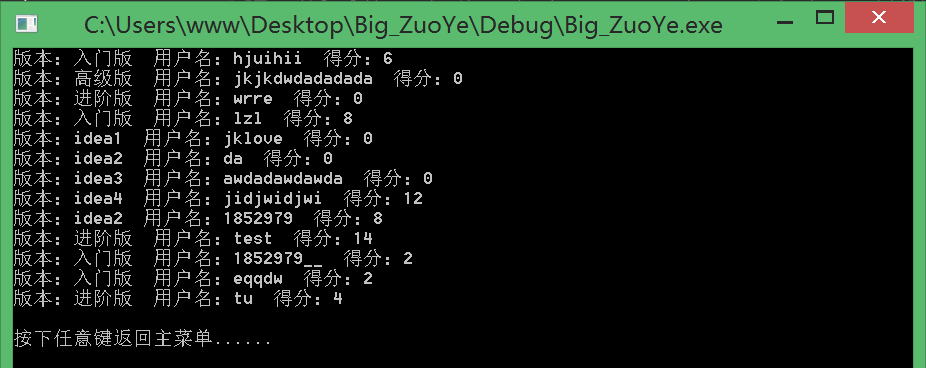


**2.2.4 高级版游戏展示**

贪吃蛇死后，尸体变成食物：



**2.2.5 得分榜展示**

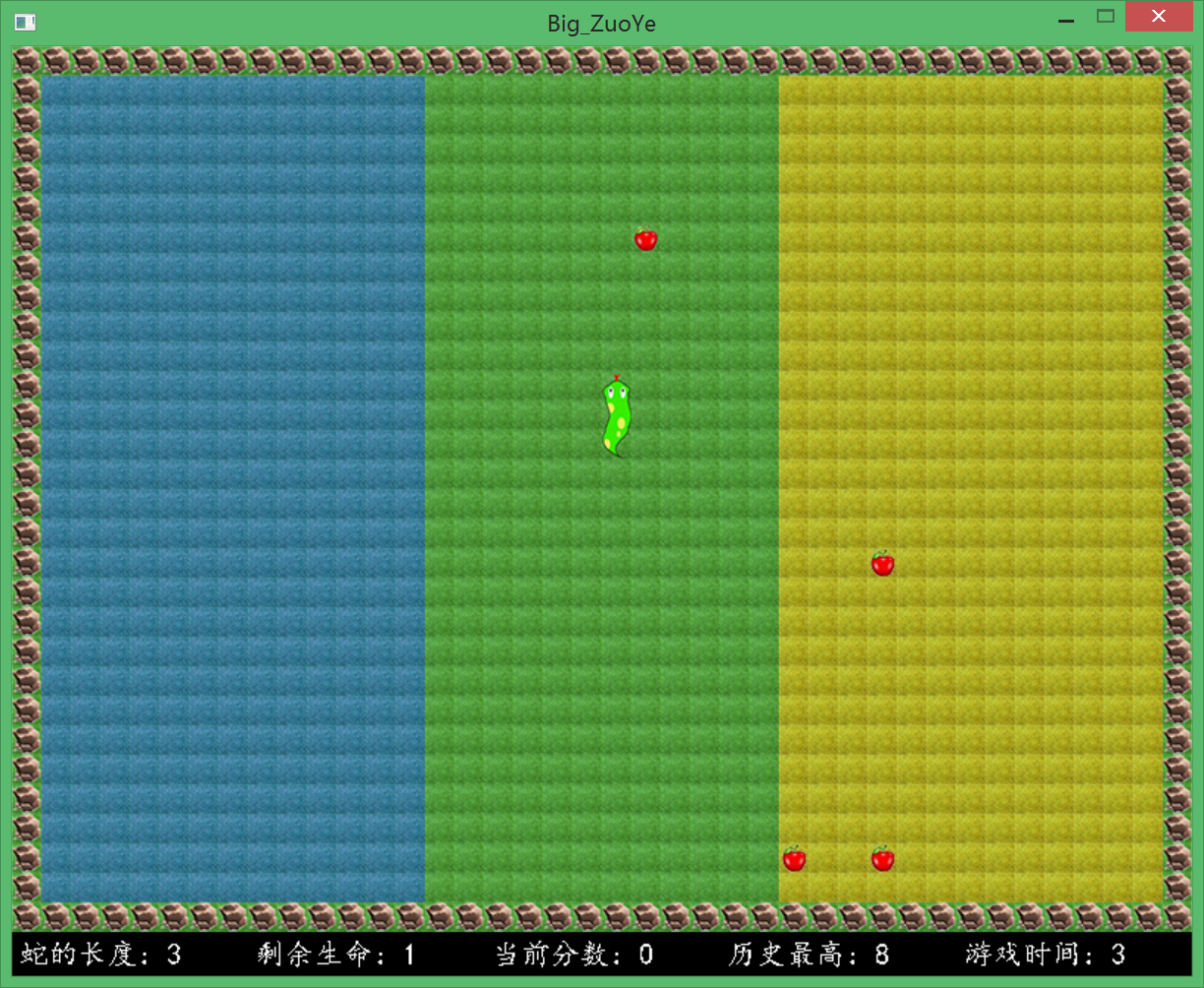
****

**2.2.6 idea1展示**



**2.2.7 idea2展示**

其中黄色块为加速区域，蓝色快为减速区域。

****

**2.2.8 idea3展示**

****

**2.2.9 idea4展示**

****

**2.2.10 idea6展示**

****

**2.2.11 人机模式展示**

****

**2.2.12 双人模式展示**

其中绿色蛇由WASD控制方向，紫色蛇由IJKL控制方向：



三、项目亮点与情况说明

**3.1 项目亮点**

①本次大作业的所有素材均为本人在PS中制作绘制而得，自认为很好地通过方块的拼合实现了贪吃蛇的蠕动效果。

②自认为本次大作业的完成度较高，完成了除多人对战、RPG模式、地图存储之外的所有功能。

③本次大作业实现了两种AI逻辑的编写。

**3.2 诸多情况说明**

①未对多人模式、RPG模式、地图存储的功能进行实现。

其中RPG模式与地图存储未能实现的原因在于，我个人认为这两个功能的难度不大，但需要耗费的精力会比较多，所以放弃了这两个模块的编写实现。

而对于多人模式，主要是由于我之前错误估计了本次大作业所要花费的时间，浪费了大量时间在绘制.jpg图片上，导致后来留给研究多人模式的时间不足。

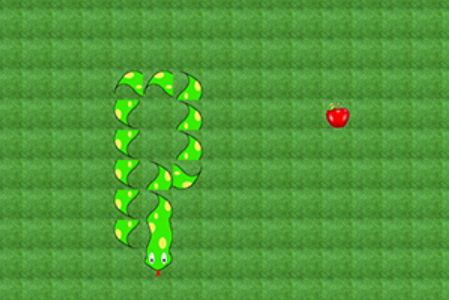
②采用了较为愚蠢的AI来实现。

正如前文所述，我发现采用BFS遍历搜索算法的AI所采取的的策略总是正确的，导致用户很难战胜电脑，而且测试时间也过长。所以我处于游戏性的考虑，最终选择了这个较为愚蠢的AI，采用BFS的AI逻辑我用代码进行了注释，TA可以取消注释，并对程序进行一定的修改来进行验证。

四、遇到的问题与解决方法

**4.1 遇到的问题**

在进行初级模式程序编写的过程中，我发现蛇总会在自己的身后留下一串尾巴，如下图所示：



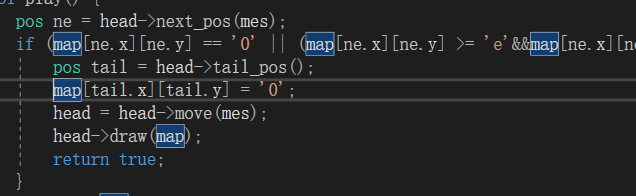
**4.2 解决方法**

这里问题出错的原因其实很容易查明——就是当我们在调用snake类的move()方法时，会删除掉蛇尾，但是这里的删除仅仅是在蛇链表中进行了删除，而未在map中修改该元素的信息，导致map中依然存储着这个尾巴的信息。

我在一开始，我的思路总是局限在要在snake类内部解决该问题，但是snake的draw()方法只能在map中绘制其占据的单元格，而不能更改其他位置的信息。于是我又试图重写了一个新的方法，专门用来去除该元素的信息。

但如此做之后，整个程序会显得十分冗余，甚至要为了这一个元素将整个map数组传递给snake的新方法。

后来，我意识到，我其实可以在Game类的play()方法中很轻易地解决这个问题，只要在移动钱获得尾巴的位置，再在移动后将尾巴的位置变成草坪即可，而不用再新建一个方法：



经过这一改写后，可以得到蛇的尾巴可以正常抹去：



五、心得体会

六、源代码

#include <iostream>

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <graphics.h>

#include <iostream>

#include <string.h>

#include <conio.h>

#include <ctime>

#include <vector>

#include <cstdlib>

#include <fstream>

#include <windows.h>

#include <time.h>

#define REC "record.txt"

using namespace std;

class Menu;

void int2char(int a, char \*s) {

char c[10];

int i = 0, temp = a, j = 0;

if (a == 0) {

c[0] = '0';

i = 1;

}

else {

for (; temp != 0;) {

c[i] = temp % 10 + '0';

i++;

temp /= 10;

}

}

i--;

for (; i >= 0; j++) {

s[j] = c[i];

i--;

}

s[j] = '\0';

}

struct pos {

int x;

int y;

};

class Game {

public:

int width = 800;

int height = 630;

void init\_game() {

initgraph(width, height);

}

void back();

virtual void init2() {

cerr << "错误！";

}

virtual void listen() {

cerr << "错误！";

}

};

class snake {

public:

pos add;

snake \*next = NULL;

bool head = false;

char tow = 'u';

int length = 0;

snake\* cut(snake \*p, int n) {

if (n == 0) {

return p;

}

snake \*k;

k = p->next;

delete p;

return cut(k, n - 1);

}

snake\* eat(char tow) {

this->head = false;

this->tow = tow;

snake \*p = new snake;

switch (tow) {

case 'u':

p->add.x = this->add.x;

p->add.y = this->add.y - 1;

break;

case 'l':

p->add.y = this->add.y;

p->add.x = this->add.x - 1;

break;

case 'd':

p->add.x = this->add.x;

p->add.y = this->add.y + 1;

break;

case 'r':

p->add.y = this->add.y;

p->add.x = this->add.x + 1;

break;

}

p->head = true;

p->tow = tow;

p->length = this->length + 1;

p->next = this;

return p;

}

snake\* move(char tow) {

this->head = false;

snake \*p, \*q;

for (p = this; p->next->next != NULL; p = p->next);

q = p->next;

delete q;

p->next = NULL;

return eat(tow);

}

pos next\_pos(char tow) {

pos ans;

switch (tow) {

case 'u':

ans.x = this->add.x;

ans.y = this->add.y - 1;

break;

case 'l':

ans.y = this->add.y;

ans.x = this->add.x - 1;

break;

case 'd':

ans.x = this->add.x;

ans.y = this->add.y + 1;

break;

case 'r':

ans.y = this->add.y;

ans.x = this->add.x + 1;

break;

}

return ans;

}

pos tail\_pos() {

snake \*p;

for (p = this; p->next != NULL; p = p->next);

return p->add;

}

bool judge(snake \*a, pos p) {

if (a->next == NULL) {

return true;

}

if (a->add.x == p.x&&a->add.y == p.y) {

return false;

}

return judge(a->next, p);

}

void draw(char a[40][30]) {

snake \*p = this;

char x;

switch (p->tow) {

case 'u':

x = 'a';

break;

case 'l':

x = 'b';

break;

case 'd':

x = 'c';

break;

case 'r':

x = 'd';

break;

}

a[p->add.x][p->add.y] = x;

p = p->next;

for (;;) {

if (p->next == NULL) {

switch (p->tow) {

case 'u':

x = 'g';

break;

case 'l':

x = 'h';

break;

case 'd':

x = 'e';

break;

case 'r':

x = 'f';

break;

}

a[p->add.x][p->add.y] = x;

break;

}

switch (p->tow) {

case 'u':

switch (p->next->tow) {

case 'u':

x = '2';

break;

case 'l':

x = '4';

break;

case 'r':

x = '3';

break;

}

break;

case 'l':

switch (p->next->tow) {

case 'u':

x = '6';

break;

case 'l':

x = '1';

break;

case 'd':

x = '3';

break;

}

break;

case 'd':

switch (p->next->tow) {

case 'l':

x = '5';

break;

case 'd':

x = '2';

break;

case 'r':

x = '6';

break;

}

break;

case 'r':

switch (p->next->tow) {

case 'u':

x = '5';

break;

case 'd':

x = '4';

break;

case 'r':

x = '1';

break;

}

break;

}

a[p->add.x][p->add.y] = x;

p = p->next;

}

}

void draw(char a[80][60]) {

snake \*p = this;

char x;

switch (p->tow) {

case 'u':

x = 'a';

break;

case 'l':

x = 'b';

break;

case 'd':

x = 'c';

break;

case 'r':

x = 'd';

break;

}

a[p->add.x][p->add.y] = x;

p = p->next;

for (;;) {

if (p->next == NULL) {

switch (p->tow) {

case 'u':

x = 'g';

break;

case 'l':

x = 'h';

break;

case 'd':

x = 'e';

break;

case 'r':

x = 'f';

break;

}

a[p->add.x][p->add.y] = x;

break;

}

switch (p->tow) {

case 'u':

switch (p->next->tow) {

case 'u':

x = '2';

break;

case 'l':

x = '4';

break;

case 'r':

x = '3';

break;

}

break;

case 'l':

switch (p->next->tow) {

case 'u':

x = '6';

break;

case 'l':

x = '1';

break;

case 'd':

x = '3';

break;

}

break;

case 'd':

switch (p->next->tow) {

case 'l':

x = '5';

break;

case 'd':

x = '2';

break;

case 'r':

x = '6';

break;

}

break;

case 'r':

switch (p->next->tow) {

case 'u':

x = '5';

break;

case 'd':

x = '4';

break;

case 'r':

x = '1';

break;

}

break;

}

a[p->add.x][p->add.y] = x;

p = p->next;

}

}

};

class snake2 {

public:

pos add;

snake2 \*next = NULL;

bool head = false;

char tow = 'u';

int length = 0;

snake2\* cut(snake2 \*p, int n) {

if (n == 0) {

return p;

}

snake2 \*k;

k = p->next;

delete p;

return cut(k, n - 1);

}

snake2\* eat(char tow) {

this->head = false;

this->tow = tow;

snake2 \*p = new snake2;

switch (tow) {

case 'u':

p->add.x = this->add.x;

p->add.y = this->add.y - 1;

break;

case 'l':

p->add.y = this->add.y;

p->add.x = this->add.x - 1;

break;

case 'd':

p->add.x = this->add.x;

p->add.y = this->add.y + 1;

break;

case 'r':

p->add.y = this->add.y;

p->add.x = this->add.x + 1;

break;

}

p->head = true;

p->tow = tow;

p->length = this->length + 1;

p->next = this;

return p;

}

snake2\* move(char tow) {

this->head = false;

snake2 \*p, \*q;

for (p = this; p->next->next != NULL; p = p->next);

q = p->next;

delete q;

p->next = NULL;

return eat(tow);

}

pos next\_pos(char tow) {

pos ans;

switch (tow) {

case 'u':

ans.x = this->add.x;

ans.y = this->add.y - 1;

break;

case 'l':

ans.y = this->add.y;

ans.x = this->add.x - 1;

break;

case 'd':

ans.x = this->add.x;

ans.y = this->add.y + 1;

break;

case 'r':

ans.y = this->add.y;

ans.x = this->add.x + 1;

break;

}

return ans;

}

pos tail\_pos() {

snake2 \*p;

for (p = this; p->next != NULL; p = p->next);

return p->add;

}

bool judge(snake2 \*a, pos p) {

if (a->next == NULL) {

return true;

}

if (a->add.x == p.x&&a->add.y == p.y) {

return false;

}

return judge(a->next, p);

}

void draw(char a[40][30]) {

snake2 \*p = this;

char x;

switch (p->tow) {

case 'u':

x = 't';

break;

case 'l':

x = 'u';

break;

case 'd':

x = 'v';

break;

case 'r':

x = 'w';

break;

}

a[p->add.x][p->add.y] = x;

p = p->next;

for (;;) {

if (p->next == NULL) {

switch (p->tow) {

case 'u':

x = 'z';

break;

case 'l':

x = '{';

break;

case 'd':

x = 'x';

break;

case 'r':

x = 'y';

break;

}

a[p->add.x][p->add.y] = x;

break;

}

switch (p->tow) {

case 'u':

switch (p->next->tow) {

case 'u':

x = 'o';

break;

case 'l':

x = 'q';

break;

case 'r':

x = 'p';

break;

}

break;

case 'l':

switch (p->next->tow) {

case 'u':

x = 's';

break;

case 'l':

x = 'n';

break;

case 'd':

x = 'p';

break;

}

break;

case 'd':

switch (p->next->tow) {

case 'l':

x = 'r';

break;

case 'd':

x = 'o';

break;

case 'r':

x = 's';

break;

}

break;

case 'r':

switch (p->next->tow) {

case 'u':

x = 'r';

break;

case 'd':

x = 'q';

break;

case 'r':

x = 'n';

break;

}

break;

}

a[p->add.x][p->add.y] = x;

p = p->next;

}

}

};

class fruit {

public:

pos add;

char style = 'g';

void draw(char a[40][30]) {

if (style == 'g') {

switch (a[add.x][add.y]) {

case '0':

a[add.x][add.y] = '7';

break;

case 'i':

a[add.x][add.y] = 'l';

break;

case 'j':

a[add.x][add.y] = 'm';

break;

}

}

else {

a[add.x][add.y] = 'k';

}

}

void draw(char a[80][60]) {

if (style == 'g') {

switch (a[add.x][add.y]) {

case '0':

a[add.x][add.y] = '7';

break;

case 'i':

a[add.x][add.y] = 'l';

break;

case 'j':

a[add.x][add.y] = 'm';

break;

}

}

else {

a[add.x][add.y] = 'k';

}

}

};

class Game1 :public Game {

public:

char map[40][30] = { '0' };

char mes = 'u';

snake \*head;

int area = 38 \* 28;

int length = 3;

int fruit\_num = 0;

int life = 1;

int score = 0;

int highscore;

clock\_t start\_t, now\_t, end\_t;

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* i, 20 \* j, &img);

}

}

}

void getrecord() {

fstream fp;

fp.open(REC, ios::in);

int a, c;

char b[100];

for (;;) {

if (fp.eof()) {

break;

}

fp >> a >> b >> c;

if (a != 1) {

continue;

}

if (c > highscore) {

highscore = c;

}

}

fp.close();

}

void init2() {

getrecord();

start\_t = clock();

snake \*p, \*q;

for (int i = 0; i < 40; i++) {

map[i][0] = '8';

map[i][29] = '8';

}

for (int j = 1; j < 29; j++) {

map[0][j] = '8';

map[39][j] = '8';

}

for (int i = 1; i < 39; i++) {

for (int j = 1; j < 29; j++) {

map[i][j] = '0';

}

}

refresh();

head = new snake;

head->head = true;

head->add.x = 20;

head->add.y = 14;

p = new snake;

head->next = p;

p->add.x = 20;

p->add.y = 15;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 20;

q->add.y = 16;

area -= 3;

head->draw(map);

draw();

}

bool play() {

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h')) {

pos tail = head->tail\_pos();

map[tail.x][tail.y] = '0';

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

if (!head->judge(head, ne)) {

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

return false;

}

if (map[ne.x][ne.y] == '8') {

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

return false;

}

}

void record() {

char sname[100];

int cp = 0;

char temp = '1';

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(20, 10, \_T("楷体"));

outtextxy(240, 340, "请输入您的英文名：");

INPUT\_RECORD keyRec;

HANDLE hIn = GetStdHandle(STD\_INPUT\_HANDLE);

DWORD state = 0, res;

while (temp != '\n') {

while (!\_kbhit()) {

}

ReadConsoleInput(hIn, &keyRec, 1, &res);

if (keyRec.Event.KeyEvent.wVirtualKeyCode == VK\_RETURN) {

break;

}

temp = \_getch();

if (temp == '\n') {

break;

}

sname[cp] = temp;

sname[cp + 1] = '\0';

cp++;

outtextxy(260, 370, sname);

}

fstream fp;

fp.open(REC, ios::app);

if (!fp.is\_open()) {

cerr << "出错！";

exit(-1);

}

fp << 1 << ' ' << sname << ' ' << score << '\n';

fp.close();

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char slen[30] = "蛇的长度：";

char slif[30] = "剩余生命：";

char ssco[30] = "当前分数：";

char smax[30] = "历史最高：";

char stim[30] = "游戏时间：";

highscore = highscore > score ? highscore : score;

int2char(length, &slen[10]);

int2char(life, &slif[10]);

int2char(score, &ssco[10]);

int2char(highscore, &smax[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, slen);

outtextxy(165, 605, slif);

outtextxy(325, 605, ssco);

outtextxy(485, 605, smax);

outtextxy(645, 605, stim);

}

void listen() {

struct tm t;

time\_t now, last;

time(&now);

localtime\_s(&t, &now);

bool flag = true;

char temp = 'u';

while (temp != 'q') {

while (!\_kbhit()) {

last = now;

time(&now);

if (last != now) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

}

}

record();

back();

}

};

class Game2 :public Game {

public:

char map[40][30] = { '0' };

char mes = 'u';

snake \*head;

int area = 38 \* 28;

int length = 3;

int fruit\_num = 0;

int life = 1;

int score = 0;

int highscore;

clock\_t start\_t, now\_t, end\_t;

void stone() {

snake \*p = head;

for (;;) {

map[p->add.x][p->add.y] = '8';

area--;

if (p->next != NULL) {

p = p->next;

}

else {

break;

}

}

}

bool new\_snake() {

mes = 'u';

int i, j;

snake \*p, \*q;

for (i = 20; i != 19; i++) {

for (j = 14; j != 13; j++) {

if (map[i][j] == '0'&&map[i][j + 1] == '0'&&map[i][j + 2] == '0') {

head = new snake;

head->head = true;

head->add.x = i;

head->add.y = j;

p = new snake;

head->next = p;

p->add.x = i;

p->add.y = j + 1;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = i;

q->add.y = j + 2;

return true;

}

if (j == 25) {

j = 0;

}

}

if (i == 38) {

i = 1;

}

}

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

return false;

}

void getrecord() {

fstream fp;

fp.open(REC, ios::in);

int a, c;

char b[100];

for (;;) {

if (fp.eof()) {

break;

}

fp >> a >> b >> c;

if (a != 2) {

continue;

}

if (c > highscore) {

highscore = c;

}

}

fp.close();

}

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* i, 20 \* j, &img);

}

}

}

void init2() {

getrecord();

start\_t = clock();

snake \*p, \*q;

for (int i = 0; i < 40; i++) {

map[i][0] = '8';

map[i][29] = '8';

}

for (int j = 1; j < 29; j++) {

map[0][j] = '8';

map[39][j] = '8';

}

for (int i = 1; i < 39; i++) {

for (int j = 1; j < 29; j++) {

map[i][j] = '0';

}

}

refresh();

head = new snake;

head->head = true;

head->add.x = 20;

head->add.y = 14;

p = new snake;

head->next = p;

p->add.x = 20;

p->add.y = 15;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 20;

q->add.y = 16;

area -= 3;

head->draw(map);

draw();

}

bool play() {

if (life == 0) {

return false;

}

bool flag = true;

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h')) {

pos tail = head->tail\_pos();

map[tail.x][tail.y] = '0';

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

if (!head->judge(head, ne)) {

stone();

score -= 10;

score = score < 0 ? 0 : score;

flag = new\_snake();

life = flag ? life : 0;

return flag;

}

if (map[ne.x][ne.y] == '8') {

stone();

score -= 10;

score = score < 0 ? 0 : score;

flag = new\_snake();

life = flag ? life : 0;

return flag;

}

}

void record() {

char sname[100];

int cp = 0;

char temp = '1';

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(20, 10, \_T("楷体"));

outtextxy(240, 340, "请输入您的英文名：");

INPUT\_RECORD keyRec;

HANDLE hIn = GetStdHandle(STD\_INPUT\_HANDLE);

DWORD state = 0, res;

while (temp != '\n') {

while (!\_kbhit()) {

}

ReadConsoleInput(hIn, &keyRec, 1, &res);

if (keyRec.Event.KeyEvent.wVirtualKeyCode == VK\_RETURN) {

break;

}

temp = \_getch();

if (temp == '\n') {

break;

}

sname[cp] = temp;

sname[cp + 1] = '\0';

cp++;

outtextxy(260, 370, sname);

}

fstream fp;

fp.open(REC, ios::app);

if (!fp.is\_open()) {

cerr << "出错！";

exit(-1);

}

fp << 2 << ' ' << sname << ' ' << score << '\n';

fp.close();

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char slen[30] = "蛇的长度：";

char slif[30] = "剩余生命：∞";

char ssco[30] = "当前分数：";

char smax[30] = "历史最高：";

char stim[30] = "游戏时间：";

highscore = highscore > score ? highscore : score;

int2char(length, &slen[10]);

int2char(score, &ssco[10]);

int2char(highscore, &smax[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, slen);

outtextxy(165, 605, slif);

outtextxy(325, 605, ssco);

outtextxy(485, 605, smax);

outtextxy(645, 605, stim);

}

void listen() {

struct tm t;

time\_t now, last;

time(&now);

localtime\_s(&t, &now);

bool flag = true;

char temp = 'u';

while (temp != 'q') {

while (!\_kbhit()) {

last = now;

time(&now);

if (last != now) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

}

}

record();

back();

}

};

class Game3 :public Game {

public:

char map[40][30] = { '0' };

char mes = 'u';

snake \*head;

int area = 38 \* 28;

int length = 3;

int fruit\_num = 0;

int life = 6;

int score = 0;

int highscore;

clock\_t start\_t, now\_t, end\_t;

void juice() {

snake \*p = head;

for (;;) {

map[p->add.x][p->add.y] = '7';

area--;

fruit\_num++;

if (p->next != NULL) {

p = p->next;

}

else {

break;

}

}

refresh();

}

bool new\_snake() {

mes = 'u';

int i, j;

snake \*p, \*q;

for (i = 20; i != 19; i++) {

for (j = 14; j != 13; j++) {

if (map[i][j] == '0'&&map[i][j + 1] == '0'&&map[i][j + 2] == '0') {

head = new snake;

head->head = true;

head->add.x = i;

head->add.y = j;

p = new snake;

head->next = p;

p->add.x = i;

p->add.y = j + 1;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = i;

q->add.y = j + 2;

return true;

}

if (j == 25) {

j = 0;

}

}

if (i == 38) {

i = 1;

}

}

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

return false;

}

void getrecord() {

fstream fp;

fp.open(REC, ios::in);

int a, c;

char b[100];

for (;;) {

if (fp.eof()) {

break;

}

fp >> a >> b >> c;

if (a != 3) {

continue;

}

if (c > highscore) {

highscore = c;

}

}

fp.close();

}

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* i, 20 \* j, &img);

}

}

}

void init2() {

getrecord();

start\_t = clock();

snake \*p, \*q;

for (int i = 0; i < 40; i++) {

map[i][0] = '8';

map[i][29] = '8';

}

for (int j = 1; j < 29; j++) {

map[0][j] = '8';

map[39][j] = '8';

}

for (int i = 1; i < 39; i++) {

for (int j = 1; j < 29; j++) {

map[i][j] = '0';

}

}

refresh();

head = new snake;

head->head = true;

head->add.x = 20;

head->add.y = 14;

p = new snake;

head->next = p;

p->add.x = 20;

p->add.y = 15;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 20;

q->add.y = 16;

area -= 3;

head->draw(map);

draw();

}

bool play() {

if (life == 0) {

return false;

}

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h')) {

pos tail = head->tail\_pos();

map[tail.x][tail.y] = '0';

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

if (!head->judge(head, ne)) {

life--;

score -= 10;

score = score < 0 ? 0 : score;

if (life == 0) {

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

return false;

}

else {

juice();

return new\_snake();

}

}

if (map[ne.x][ne.y] == '8') {

life--;

score -= 10;

score = score < 0 ? 0 : score;

if (life == 0) {

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

return false;

}

else {

juice();

return new\_snake();

}

}

}

void record() {

char sname[100];

int cp = 0;

char temp = '1';

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(20, 10, \_T("楷体"));

outtextxy(240, 340, "请输入您的英文名：");

INPUT\_RECORD keyRec;

HANDLE hIn = GetStdHandle(STD\_INPUT\_HANDLE);

DWORD state = 0, res;

while (temp != '\n') {

while (!\_kbhit()) {

}

ReadConsoleInput(hIn, &keyRec, 1, &res);

if (keyRec.Event.KeyEvent.wVirtualKeyCode == VK\_RETURN) {

break;

}

temp = \_getch();

if (temp == '\n') {

break;

}

sname[cp] = temp;

sname[cp + 1] = '\0';

cp++;

outtextxy(260, 370, sname);

}

fstream fp;

fp.open(REC, ios::app);

if (!fp.is\_open()) {

cerr << "出错！";

exit(-1);

}

fp << 3 << ' ' << sname << ' ' << score << '\n';

fp.close();

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char slen[30] = "蛇的长度：";

char slif[30] = "剩余生命：";

char ssco[30] = "当前分数：";

char smax[30] = "历史最高：";

char stim[30] = "游戏时间：";

highscore = highscore > score ? highscore : score;

int2char(length, &slen[10]);

int2char(score, &ssco[10]);

int2char(life, &slif[10]);

int2char(highscore, &smax[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, slen);

outtextxy(165, 605, slif);

outtextxy(325, 605, ssco);

outtextxy(485, 605, smax);

outtextxy(645, 605, stim);

}

void listen() {

struct tm t;

time\_t now, last;

time(&now);

localtime\_s(&t, &now);

bool flag = true;

char temp = 'u';

while (temp != 'q') {

while (!\_kbhit()) {

last = now;

time(&now);

if (last != now) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

}

}

record();

back();

}

};

class Game4 :public Game {

public:

char map[40][30] = { '0' };

char mes = 'u';

snake \*head;

int area = 38 \* 28;

int length = 3;

int fruit\_num = 0;

int life = 1;

int score = 0;

int highscore;

clock\_t start\_t, now\_t, end\_t;

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* i, 20 \* j, &img);

}

}

}

void getrecord() {

fstream fp;

fp.open(REC, ios::in);

int a, c;

char b[100];

for (;;) {

if (fp.eof()) {

break;

}

fp >> a >> b >> c;

if (a != 4) {

continue;

}

if (c > highscore) {

highscore = c;

}

}

fp.close();

}

void init2() {

getrecord();

start\_t = clock();

snake \*p, \*q;

for (int i = 0; i < 40; i++) {

map[i][0] = '8';

map[i][29] = '8';

}

for (int j = 1; j < 29; j++) {

map[0][j] = '8';

map[39][j] = '8';

}

for (int i = 1; i < 39; i++) {

for (int j = 1; j < 29; j++) {

map[i][j] = '0';

}

}

for (int i = 5; i <= 8; i++) {

for (int j = 5; j <= 8; j++) {

map[i][j] = '9';

map[i][j + 16] = '9';

map[i + 26][j] = '9';

map[i + 26][j + 16] = '9';

}

}

area -= 64;

refresh();

head = new snake;

head->head = true;

head->add.x = 20;

head->add.y = 14;

p = new snake;

head->next = p;

p->add.x = 20;

p->add.y = 15;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 20;

q->add.y = 16;

area -= 3;

head->draw(map);

draw();

}

void endit() {

life = 0;

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

}

void erase\_s(snake \*p, int n) {

if (n == 0) {

p->tow = p->next->tow;

switch (p->tow) {

case 'u':

map[p->add.x][p->add.y] = 'a';

break;

case 'r':

map[p->add.x][p->add.y] = 'd';

break;

case 'd':

map[p->add.x][p->add.y] = 'c';

break;

case 'l':

map[p->add.x][p->add.y] = 'b';

break;

}

return;

}

map[p->add.x][p->add.y] = '0';

erase\_s(p->next, n - 1);

}

bool play() {

if (life == 0) {

return false;

}

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h')) {

pos tail = head->tail\_pos();

map[tail.x][tail.y] = '0';

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

if (!head->judge(head, ne)) {

endit();

return false;

}

if (map[ne.x][ne.y] == '8') {

if (length <= 5) {

endit();

return false;

}

erase\_s(head, length - length / 2);

head = head->cut(head, length - length / 2);

length /= 2;

score -= 10;

score = score < 0 ? 0 : score;

return true;

}

if (map[ne.x][ne.y] == '9') {

if (length <= 4) {

endit();

return false;

}

erase\_s(head, 2);

head = head->cut(head, 2);

length -= 2;

score -= 10;

score = score < 0 ? 0 : score;

return true;

}

}

void record() {

char sname[100];

int cp = 0;

char temp = '1';

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(20, 10, \_T("楷体"));

outtextxy(240, 340, "请输入您的英文名：");

INPUT\_RECORD keyRec;

HANDLE hIn = GetStdHandle(STD\_INPUT\_HANDLE);

DWORD state = 0, res;

while (temp != '\n') {

while (!\_kbhit()) {

}

ReadConsoleInput(hIn, &keyRec, 1, &res);

if (keyRec.Event.KeyEvent.wVirtualKeyCode == VK\_RETURN) {

break;

}

temp = \_getch();

if (temp == '\n') {

break;

}

sname[cp] = temp;

sname[cp + 1] = '\0';

cp++;

outtextxy(260, 370, sname);

}

fstream fp;

fp.open(REC, ios::app);

if (!fp.is\_open()) {

cerr << "出错！";

exit(-1);

}

fp << 4 << ' ' << sname << ' ' << score << '\n';

fp.close();

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char slen[30] = "蛇的长度：";

char slif[30] = "剩余生命：";

char ssco[30] = "当前分数：";

char smax[30] = "历史最高：";

char stim[30] = "游戏时间：";

highscore = highscore > score ? highscore : score;

int2char(length, &slen[10]);

int2char(life, &slif[10]);

int2char(score, &ssco[10]);

int2char(highscore, &smax[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, slen);

outtextxy(165, 605, slif);

outtextxy(325, 605, ssco);

outtextxy(485, 605, smax);

outtextxy(645, 605, stim);

}

void listen() {

struct tm t;

time\_t now, last;

time(&now);

localtime\_s(&t, &now);

bool flag = true;

char temp = 'u';

while (temp != 'q') {

while (!\_kbhit()) {

last = now;

time(&now);

if (last != now) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

}

}

record();

back();

}

};

class Game5 :public Game {

public:

char map[40][30] = { '0' };

char mes = 'u';

snake \*head;

int area = 38 \* 28;

int length = 3;

int fruit\_num = 0;

int life = 1;

int score = 0;

int highscore;

clock\_t start\_t, now\_t, end\_t;

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0' && map[p->add.x][p->add.y] != 'i' && map[p->add.x][p->add.y] != 'j') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* i, 20 \* j, &img);

}

}

}

void getrecord() {

fstream fp;

fp.open(REC, ios::in);

int a, c;

char b[100];

for (;;) {

if (fp.eof()) {

break;

}

fp >> a >> b >> c;

if (a != 5) {

continue;

}

if (c > highscore) {

highscore = c;

}

}

fp.close();

}

void init2() {

getrecord();

start\_t = clock();

snake \*p, \*q;

for (int i = 0; i < 40; i++) {

map[i][0] = '8';

map[i][29] = '8';

}

for (int j = 1; j < 29; j++) {

map[0][j] = '8';

map[39][j] = '8';

}

for (int i = 1; i < 39; i++) {

for (int j = 1; j < 29; j++) {

map[i][j] = '0';

}

}

for (int i = 1; i <= 13; i++) {

for (int j = 1; j <= 28; j++) {

map[i][j] = 'i';

}

}

for (int i = 26; i <= 38; i++) {

for (int j = 1; j <= 28; j++) {

map[i][j] = 'j';

}

}

refresh();

head = new snake;

head->head = true;

head->add.x = 20;

head->add.y = 14;

p = new snake;

head->next = p;

p->add.x = 20;

p->add.y = 15;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 20;

q->add.y = 16;

area -= 3;

head->draw(map);

draw();

}

void endit() {

life = 0;

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

}

void remap(pos add) {

if (add.x <= 13) {

map[add.x][add.y] = 'i';

}

else if (add.x >= 26) {

map[add.x][add.y] = 'j';

}

else {

map[add.x][add.y] = '0';

}

}

bool play() {

if (life == 0) {

return false;

}

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h') || map[ne.x][ne.y] == 'i' || map[ne.x][ne.y] == 'j') {

pos tail = head->tail\_pos();

remap(tail);

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7' || map[ne.x][ne.y] == 'l' || map[ne.x][ne.y] == 'm') {

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

if (!head->judge(head, ne)) {

endit();

return false;

}

if (map[ne.x][ne.y] == '8') {

endit();

return false;

}

}

void record() {

char sname[100];

int cp = 0;

char temp = '1';

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(20, 10, \_T("楷体"));

outtextxy(240, 340, "请输入您的英文名：");

INPUT\_RECORD keyRec;

HANDLE hIn = GetStdHandle(STD\_INPUT\_HANDLE);

DWORD state = 0, res;

while (temp != '\n') {

while (!\_kbhit()) {

}

ReadConsoleInput(hIn, &keyRec, 1, &res);

if (keyRec.Event.KeyEvent.wVirtualKeyCode == VK\_RETURN) {

break;

}

temp = \_getch();

if (temp == '\n') {

break;

}

sname[cp] = temp;

sname[cp + 1] = '\0';

cp++;

outtextxy(260, 370, sname);

}

fstream fp;

fp.open(REC, ios::app);

if (!fp.is\_open()) {

cerr << "出错！";

exit(-1);

}

fp << 5 << ' ' << sname << ' ' << score << '\n';

fp.close();

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char slen[30] = "蛇的长度：";

char slif[30] = "剩余生命：";

char ssco[30] = "当前分数：";

char smax[30] = "历史最高：";

char stim[30] = "游戏时间：";

highscore = highscore > score ? highscore : score;

int2char(length, &slen[10]);

int2char(life, &slif[10]);

int2char(score, &ssco[10]);

int2char(highscore, &smax[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, slen);

outtextxy(165, 605, slif);

outtextxy(325, 605, ssco);

outtextxy(485, 605, smax);

outtextxy(645, 605, stim);

}

void listen() {

bool flag = true;

DWORD now, last;

char temp = 'u';

while (temp != 'q') {

now = GetTickCount();

last = now;

while (!\_kbhit()) {

now = GetTickCount();

if (head->add.x >= 26) {

if ((int)(now - last) >= 500) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

last = now;

}

}

else if (head->add.x <= 13) {

if ((int)(now - last) >= 2000) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

last = now;

}

}

else {

if ((int)(now - last) >= 1000) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

last = now;

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

}

}

record();

back();

}

};

class Game6 :public Game {

public:

char map[40][30] = { '0' };

char mes = 'u';

snake \*head;

int area = 38 \* 28;

int length = 3;

int fruit\_num = 0;

int life = 100;

int score = 0;

int highscore;

clock\_t start\_t, now\_t, end\_t;

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* i, 20 \* j, &img);

}

}

}

void getrecord() {

fstream fp;

fp.open(REC, ios::in);

int a, c;

char b[100];

for (;;) {

if (fp.eof()) {

break;

}

fp >> a >> b >> c;

if (a != 6) {

continue;

}

if (c > highscore) {

highscore = c;

}

}

fp.close();

}

void init2() {

getrecord();

start\_t = clock();

snake \*p, \*q;

for (int i = 0; i < 40; i++) {

map[i][0] = '8';

map[i][29] = '8';

}

for (int j = 1; j < 29; j++) {

map[0][j] = '8';

map[39][j] = '8';

}

for (int i = 1; i < 39; i++) {

for (int j = 1; j < 29; j++) {

map[i][j] = '0';

}

}

refresh();

head = new snake;

head->head = true;

head->add.x = 20;

head->add.y = 14;

p = new snake;

head->next = p;

p->add.x = 20;

p->add.y = 15;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 20;

q->add.y = 16;

area -= 3;

head->draw(map);

draw();

}

void endit() {

life = 0;

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

}

void erase\_s(snake \*p, int n) {

if (n == 0) {

p->tow = p->next->tow;

switch (p->tow) {

case 'u':

map[p->add.x][p->add.y] = 'a';

break;

case 'r':

map[p->add.x][p->add.y] = 'd';

break;

case 'd':

map[p->add.x][p->add.y] = 'c';

break;

case 'l':

map[p->add.x][p->add.y] = 'b';

break;

}

return;

}

map[p->add.x][p->add.y] = '0';

erase\_s(p->next, n - 1);

}

bool play() {

if (life <= 0) {

endit();

return false;

}

life--;

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h')) {

pos tail = head->tail\_pos();

map[tail.x][tail.y] = '0';

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

life += 30;

life = life > 100 ? 100 : life;

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

if (!head->judge(head, ne)) {

endit();

return false;

}

if (map[ne.x][ne.y] == '8') {

endit();

return false;

}

if (life <= 0) {

endit();

return false;

}

}

void record() {

char sname[100];

int cp = 0;

char temp = '1';

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(20, 10, \_T("楷体"));

outtextxy(240, 340, "请输入您的英文名：");

INPUT\_RECORD keyRec;

HANDLE hIn = GetStdHandle(STD\_INPUT\_HANDLE);

DWORD state = 0, res;

while (temp != '\n') {

while (!\_kbhit()) {

}

ReadConsoleInput(hIn, &keyRec, 1, &res);

if (keyRec.Event.KeyEvent.wVirtualKeyCode == VK\_RETURN) {

break;

}

temp = \_getch();

if (temp == '\n') {

break;

}

sname[cp] = temp;

sname[cp + 1] = '\0';

cp++;

outtextxy(260, 370, sname);

}

fstream fp;

fp.open(REC, ios::app);

if (!fp.is\_open()) {

cerr << "出错！";

exit(-1);

}

fp << 6 << ' ' << sname << ' ' << score << '\n';

fp.close();

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char slen[30] = "蛇的长度：";

char slif[30] = "体能：";

char ssco[30] = "当前分数：";

char smax[30] = "历史最高：";

char stim[30] = "游戏时间：";

highscore = highscore > score ? highscore : score;

int2char(length, &slen[10]);

int2char(score, &ssco[10]);

int2char(highscore, &smax[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, slen);

outtextxy(165, 605, slif);

outtextxy(325, 605, ssco);

outtextxy(485, 605, smax);

outtextxy(645, 605, stim);

setfillcolor(0xC1B6FF);

solidrectangle(220, 605, 220 + life, 625);

setfillcolor(0x555555);

solidrectangle(220 + life, 605, 320, 625);

}

void listen() {

struct tm t;

time\_t now, last;

time(&now);

localtime\_s(&t, &now);

bool flag = true;

char temp = 'u';

while (temp != 'q') {

while (!\_kbhit()) {

last = now;

time(&now);

if (last != now) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

}

}

record();

back();

}

};

class Game7 :public Game {

public:

char map[40][30] = { '0' };

char mes = 'u';

snake \*head;

int area = 38 \* 28;

int length = 3;

int fruit\_num = 0;

int life = 1;

int score = 0;

int highscore;

clock\_t start\_t, now\_t, end\_t;

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

num = rand() % 5 + 1;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

p->style = 'b';

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* i, 20 \* j, &img);

}

}

}

void getrecord() {

fstream fp;

fp.open(REC, ios::in);

int a, c;

char b[100];

for (;;) {

if (fp.eof()) {

break;

}

fp >> a >> b >> c;

if (a != 7) {

continue;

}

if (c > highscore) {

highscore = c;

}

}

fp.close();

}

void init2() {

getrecord();

start\_t = clock();

snake \*p, \*q;

for (int i = 0; i < 40; i++) {

map[i][0] = '8';

map[i][29] = '8';

}

for (int j = 1; j < 29; j++) {

map[0][j] = '8';

map[39][j] = '8';

}

for (int i = 1; i < 39; i++) {

for (int j = 1; j < 29; j++) {

map[i][j] = '0';

}

}

refresh();

head = new snake;

head->head = true;

head->add.x = 20;

head->add.y = 14;

p = new snake;

head->next = p;

p->add.x = 20;

p->add.y = 15;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 20;

q->add.y = 16;

area -= 3;

head->draw(map);

draw();

}

void endit() {

life = 0;

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

}

void erase\_s(snake \*p, int n) {

if (n == 0) {

p->tow = p->next->tow;

switch (p->tow) {

case 'u':

map[p->add.x][p->add.y] = 'a';

break;

case 'r':

map[p->add.x][p->add.y] = 'd';

break;

case 'd':

map[p->add.x][p->add.y] = 'c';

break;

case 'l':

map[p->add.x][p->add.y] = 'b';

break;

}

return;

}

map[p->add.x][p->add.y] = '0';

erase\_s(p->next, n - 1);

}

bool play() {

if (life == 0) {

return false;

}

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h')) {

pos tail = head->tail\_pos();

map[tail.x][tail.y] = '0';

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

if (!head->judge(head, ne)) {

endit();

return false;

}

if (map[ne.x][ne.y] == '8') {

endit();

return false;

}

if (map[ne.x][ne.y] == 'k') {

endit();

return false;

}

}

void record() {

char sname[100];

int cp = 0;

char temp = '1';

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(20, 10, \_T("楷体"));

outtextxy(240, 340, "请输入您的英文名：");

INPUT\_RECORD keyRec;

HANDLE hIn = GetStdHandle(STD\_INPUT\_HANDLE);

DWORD state = 0, res;

while (temp != '\n') {

while (!\_kbhit()) {

}

ReadConsoleInput(hIn, &keyRec, 1, &res);

if (keyRec.Event.KeyEvent.wVirtualKeyCode == VK\_RETURN) {

break;

}

temp = \_getch();

if (temp == '\n') {

break;

}

sname[cp] = temp;

sname[cp + 1] = '\0';

cp++;

outtextxy(260, 370, sname);

}

fstream fp;

fp.open(REC, ios::app);

if (!fp.is\_open()) {

cerr << "出错！";

exit(-1);

}

fp << 7 << ' ' << sname << ' ' << score << '\n';

fp.close();

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char slen[30] = "蛇的长度：";

char slif[30] = "剩余生命：";

char ssco[30] = "当前分数：";

char smax[30] = "历史最高：";

char stim[30] = "游戏时间：";

highscore = highscore > score ? highscore : score;

int2char(length, &slen[10]);

int2char(life, &slif[10]);

int2char(score, &ssco[10]);

int2char(highscore, &smax[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, slen);

outtextxy(165, 605, slif);

outtextxy(325, 605, ssco);

outtextxy(485, 605, smax);

outtextxy(645, 605, stim);

}

void listen() {

struct tm t;

time\_t now, last;

time(&now);

localtime\_s(&t, &now);

bool flag = true;

char temp = 'u';

while (temp != 'q') {

while (!\_kbhit()) {

last = now;

time(&now);

if (last != now) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

}

}

record();

back();

}

};

class Game8 :public Game {

public:

char map[80][60] = { '0' };

char mes = 'u';

snake \*head;

int area = 78 \* 58;

int length = 3;

int fruit\_num = 0;

int life = 1;

int score = 0;

int highscore;

clock\_t start\_t, now\_t, end\_t;

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 58; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

pos vec;

if (head->add.x < 20) {

vec.x = 0;

}

else if (head->add.x > 60) {

vec.x = 40;

}

else {

vec.x = head->add.x - 20;

}

if (head->add.y < 15) {

vec.y = 0;

}

else if (head->add.y > 45) {

vec.y = 30;

}

else {

vec.y = head->add.y - 15;

}

for (int i = vec.x; i < vec.x + 40; i++) {

for (int j = vec.y; j < vec.y + 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* (i - vec.x), 20 \* (j - vec.y), &img);

}

}

}

void getrecord() {

fstream fp;

fp.open(REC, ios::in);

int a, c;

char b[100];

for (;;) {

if (fp.eof()) {

break;

}

fp >> a >> b >> c;

if (a != 8) {

continue;

}

if (c > highscore) {

highscore = c;

}

}

fp.close();

}

void init2() {

getrecord();

start\_t = clock();

snake \*p, \*q;

for (int i = 0; i < 80; i++) {

map[i][0] = '8';

map[i][59] = '8';

}

for (int j = 1; j < 59; j++) {

map[0][j] = '8';

map[79][j] = '8';

}

for (int i = 1; i < 79; i++) {

for (int j = 1; j < 59; j++) {

map[i][j] = '0';

}

}

refresh();

head = new snake;

head->head = true;

head->add.x = 40;

head->add.y = 29;

p = new snake;

head->next = p;

p->add.x = 40;

p->add.y = 30;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 40;

q->add.y = 31;

area -= 3;

head->draw(map);

draw();

}

void endit() {

life = 0;

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "游戏结束");

}

void erase\_s(snake \*p, int n) {

if (n == 0) {

p->tow = p->next->tow;

switch (p->tow) {

case 'u':

map[p->add.x][p->add.y] = 'a';

break;

case 'r':

map[p->add.x][p->add.y] = 'd';

break;

case 'd':

map[p->add.x][p->add.y] = 'c';

break;

case 'l':

map[p->add.x][p->add.y] = 'b';

break;

}

return;

}

map[p->add.x][p->add.y] = '0';

erase\_s(p->next, n - 1);

}

bool play() {

if (life == 0) {

return false;

}

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h')) {

pos tail = head->tail\_pos();

map[tail.x][tail.y] = '0';

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

if (!head->judge(head, ne)) {

endit();

return false;

}

if (map[ne.x][ne.y] == '8') {

endit();

return false;

}

if (map[ne.x][ne.y] == 'k') {

endit();

return false;

}

}

void record() {

char sname[100];

int cp = 0;

char temp = '1';

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(20, 10, \_T("楷体"));

outtextxy(240, 340, "请输入您的英文名：");

INPUT\_RECORD keyRec;

HANDLE hIn = GetStdHandle(STD\_INPUT\_HANDLE);

DWORD state = 0, res;

while (temp != '\n') {

while (!\_kbhit()) {

}

ReadConsoleInput(hIn, &keyRec, 1, &res);

if (keyRec.Event.KeyEvent.wVirtualKeyCode == VK\_RETURN) {

break;

}

temp = \_getch();

if (temp == '\n') {

break;

}

sname[cp] = temp;

sname[cp + 1] = '\0';

cp++;

outtextxy(260, 370, sname);

}

fstream fp;

fp.open(REC, ios::app);

if (!fp.is\_open()) {

cerr << "出错！";

exit(-1);

}

fp << 8 << ' ' << sname << ' ' << score << '\n';

fp.close();

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char slen[30] = "蛇的长度：";

char slif[30] = "剩余生命：";

char ssco[30] = "当前分数：";

char smax[30] = "历史最高：";

char stim[30] = "游戏时间：";

highscore = highscore > score ? highscore : score;

int2char(length, &slen[10]);

int2char(life, &slif[10]);

int2char(score, &ssco[10]);

int2char(highscore, &smax[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, slen);

outtextxy(165, 605, slif);

outtextxy(325, 605, ssco);

outtextxy(485, 605, smax);

outtextxy(645, 605, stim);

}

void listen() {

struct tm t;

time\_t now, last;

time(&now);

localtime\_s(&t, &now);

bool flag = true;

char temp = 'u';

while (temp != 'q') {

while (!\_kbhit()) {

last = now;

time(&now);

if (last != now) {

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

}

}

record();

back();

}

};

class ai {

public:

char lac;

char map[40][30];

void copy\_map(char a[40][30]) {

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

map[i][j] = a[i][j];

}

}

}

/\*void find\_goal() {

dot temp = dots.front();

dots.pop();

if (map[temp.add.x][temp.add.y] == '7') {

goal = temp;

return;

}

map[temp.add.x][temp.add.y] = '\0';

pos p;

dot \*uhi;

p.x = temp.add.x + 1;

p.y = temp.add.y;

if (p.x < 39) {

if (map[p.x][p.y] == '0' || (map[p.x][p.y] <= '{'&&map[p.x][p.y] >= 'x') || map[p.x][p.y] == '7') {

uhi = new dot;

uhi->add = p;

uhi->a[uhi->num++] = 'r';

if (map[p.x][p.y] == '7') {

goal = \*uhi;

return;

}

dots.push(\*uhi);

}

}

p.x = temp.add.x - 1;

p.y = temp.add.y;

if (p.x > 0) {

if (map[p.x][p.y] == '0' || (map[p.x][p.y] <= '{'&&map[p.x][p.y] >= 'x') || map[p.x][p.y] == '7') {

uhi = new dot;

uhi->add = p;

uhi->a[uhi->num++] = 'l';

if (map[p.x][p.y] == '7') {

goal = \*uhi;

return;

}

dots.push(\*uhi);

}

}

p.x = temp.add.x;

p.y = temp.add.y + 1;

if (p.y < 29) {

if (map[p.x][p.y] == '0' || (map[p.x][p.y] <= '{'&&map[p.x][p.y] >= 'x') || map[p.x][p.y] == '7') {

uhi = new dot;

uhi->add = p;

uhi->a[uhi->num++] = 'd';

if (map[p.x][p.y] == '7') {

goal = \*uhi;

return;

}

dots.push(\*uhi);

}

}

p.x = temp.add.x;

p.y = temp.add.y - 1;

if (p.y > 0) {

if (map[p.x][p.y] == '0' || (map[p.x][p.y] <= '{'&&map[p.x][p.y] >= 'x') || map[p.x][p.y] == '7') {

uhi = new dot;

uhi->add = p;

uhi->a[uhi->num++] = 'u';

if (map[p.x][p.y] == '7') {

goal = \*uhi;

return;

}

dots.push(\*uhi);

}

}

find\_goal();

}\*/

char find\_goal(char a, snake2 \*hea) {

pos p;

p = hea->add;

int i;

int tempa, tempb;

switch (a) {

case'u':

for (i = p.y - 1; map[p.x][i] == '0'; i--);

if (map[p.x][i] == '7') {

lac = 'u';

break;

}

for (i = p.x - 1; map[i][p.y] == '0'; i--);

if (map[i][p.y] == '7') {

lac = 'l';

break;

}

for (i = p.x + 1; map[i][p.y] == '0'; i++);

if (map[i][p.y] == '7') {

lac = 'r';

break;

}

tempa = rand() % 30;

tempb = rand() % 40;

if (p.y > tempa) {

lac = 'u';

break;

}

if (p.x > tempb) {

lac = 'l';

break;

}

lac = 'r';

break;

case 'd':

for (i = p.y + 1; map[p.x][i] == '0'; i++);

if (map[p.x][i] == '7') {

lac = 'd';

break;

}

for (i = p.x - 1; map[i][p.y] == '0'; i--);

if (map[i][p.y] == '7') {

lac = 'l';

break;

}

for (i = p.x + 1; map[i][p.y] == '0'; i++);

if (map[i][p.y] == '7') {

lac = 'r';

break;

}

tempa = rand() % 30;

tempb = rand() % 40;

if (p.y <= tempa) {

lac = 'd';

break;

}

if (p.x > tempb) {

lac = 'l';

break;

}

lac = 'r';

break;

case 'l':

for (i = p.x - 1; map[i][p.y] == '0'; i--);

if (map[i][p.y] == '7') {

lac = 'l';

break;

}

for (i = p.y + 1; map[p.x][i] == '0'; i++);

if (map[p.x][i] == '7') {

lac = 'd';

break;

}

for (i = p.y - 1; map[p.x][i] == '0'; i--);

if (map[p.x][i] == '7') {

lac = 'u';

break;

}

tempa = rand() % 30;

tempb = rand() % 40;

if (p.x > tempb) {

lac = 'l';

break;

}

if (p.y <= tempa) {

lac = 'd';

break;

}

lac = 'u';

break;

case 'r':

for (i = p.x + 1; map[i][p.y] == '0'; i++);

if (map[i][p.y] == '7') {

lac = 'r';

break;

}

for (i = p.y + 1; map[p.x][i] == '0'; i++);

if (map[p.x][i] == '7') {

lac = 'd';

break;

}

for (i = p.y - 1; map[p.x][i] == '0'; i--);

if (map[p.x][i] == '7') {

lac = 'u';

break;

}

tempa = rand() % 30;

tempb = rand() % 40;

if (p.x <= tempb) {

lac = 'r';

break;

}

if (p.y <= tempa) {

lac = 'd';

break;

}

lac = 'u';

break;

}

return lac;

}

char make\_choice(char lc, snake2 \*head) {

char ans = find\_goal(lc, head);

return ans;

}

};

class Game9 :public Game {

public:

char map[40][30] = { '0' };

char mes = 'u';

snake \*head;

int area = 38 \* 28;

int length = 3;

int fruit\_num = 0;

int score = 0;

int life = 1;

clock\_t start\_t, now\_t, end\_t;

char ames = 'u';

snake2 \*ahead;

int alength = 3;

int ascore = 0;

int alife = 1;

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* i, 20 \* j, &img);

}

}

}

void init2() {

start\_t = clock();

snake \*p, \*q;

snake2 \*p2, \*q2;

for (int i = 0; i < 40; i++) {

map[i][0] = '8';

map[i][29] = '8';

}

for (int j = 1; j < 29; j++) {

map[0][j] = '8';

map[39][j] = '8';

}

for (int i = 1; i < 39; i++) {

for (int j = 1; j < 29; j++) {

map[i][j] = '0';

}

}

refresh();

head = new snake;

head->head = true;

head->add.x = 10;

head->add.y = 14;

p = new snake;

head->next = p;

p->add.x = 10;

p->add.y = 15;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 10;

q->add.y = 16;

area -= 3;

head->draw(map);

ahead = new snake2;

ahead->head = true;

ahead->add.x = 30;

ahead->add.y = 14;

p2 = new snake2;

ahead->next = p2;

p2->add.x = 30;

p2->add.y = 15;

q2 = new snake2;

p2->next = q2;

q2->next = NULL;

q2->add.x = 30;

q2->add.y = 16;

area -= 3;

ahead->draw(map);

draw();

}

void endit() {

life = 0;

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "您失败了");

}

void winit() {

alife = 0;

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "您胜利了");

}

bool play() {

if (life == 0) {

return false;

}

if (alife == 0) {

return false;

}

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h') || (map[ne.x][ne.y] >= 'x'&&map[ne.x][ne.y] <= '{')) {

pos tail = head->tail\_pos();

map[tail.x][tail.y] = '0';

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

else {

endit();

return false;

}

return false;

}

bool aplay() {

if (life == 0) {

return false;

}

if (alife == 0) {

return false;

}

pos ne = ahead->next\_pos(ames);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'x'&&map[ne.x][ne.y] <= '{') || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h')) {

pos tail = ahead->tail\_pos();

map[tail.x][tail.y] = '0';

ahead = ahead->move(ames);

ahead->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

ahead = ahead->eat(ames);

ahead->draw(map);

fruit\_num--;

alength++;

ascore += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

else {

winit();

return false;

}

return false;

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char ylen[30] = "你的长度：";

char ysco[30] = "你的分数：";

char alen[30] = "AI的长度：";

char asco[30] = "AI的分数：";

char stim[30] = "游戏时间：";

int2char(length, &ylen[10]);

int2char(score, &ysco[10]);

int2char(alength, &alen[10]);

int2char(ascore, &asco[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, ylen);

outtextxy(165, 605, ysco);

outtextxy(325, 605, alen);

outtextxy(485, 605, asco);

outtextxy(645, 605, stim);

}

void listen() {

struct tm t;

time\_t now, last;

time(&now);

localtime\_s(&t, &now);

bool flag = true;

bool afalg = true;

ai per;

char temp = 'u';

while (temp != 'q') {

while (!\_kbhit()) {

last = now;

time(&now);

if (last != now) {

if (afalg = aplay()) {

per.copy\_map(map);

ames = per.make\_choice(ames, ahead);

//draw();

}

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

}

}

back();

}

};

class Game10 :public Game {

public:

char map[40][30] = { '0' };

char mes = 'u';

snake \*head;

int area = 38 \* 28;

int length = 3;

int fruit\_num = 0;

int score = 0;

int life = 1;

clock\_t start\_t, now\_t, end\_t;

char ames = 'u';

snake2 \*ahead;

int alength = 3;

int ascore = 0;

int alife = 1;

void refresh() {

int num = rand() % 5 + 1;

int pl;

fruit \*p;

for (int uhi = 0; uhi < num; uhi++) {

p = new fruit;

pl = rand() % area;

for (p->add.x = 1;; p->add.x++) {

for (p->add.y = 1; p->add.y <= 28; p->add.y++) {

if (map[p->add.x][p->add.y] != '0') {

continue;

}

pl--;

if (pl == 0) {

break;

}

}

if (pl == 0) {

break;

}

area--;

}

p->draw(map);

fruit\_num++;

}

}

void draw() {

char s[7] = "\_?.jpg";

IMAGE img;

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 30; j++) {

s[1] = map[i][j];

loadimage(&img, s);

putimage(20 \* i, 20 \* j, &img);

}

}

}

void init2() {

start\_t = clock();

snake \*p, \*q;

snake2 \*p2, \*q2;

for (int i = 0; i < 40; i++) {

map[i][0] = '8';

map[i][29] = '8';

}

for (int j = 1; j < 29; j++) {

map[0][j] = '8';

map[39][j] = '8';

}

for (int i = 1; i < 39; i++) {

for (int j = 1; j < 29; j++) {

map[i][j] = '0';

}

}

refresh();

head = new snake;

head->head = true;

head->add.x = 10;

head->add.y = 14;

p = new snake;

head->next = p;

p->add.x = 10;

p->add.y = 15;

q = new snake;

p->next = q;

q->next = NULL;

q->add.x = 10;

q->add.y = 16;

area -= 3;

head->draw(map);

ahead = new snake2;

ahead->head = true;

ahead->add.x = 30;

ahead->add.y = 14;

p2 = new snake2;

ahead->next = p2;

p2->add.x = 30;

p2->add.y = 15;

q2 = new snake2;

p2->next = q2;

q2->next = NULL;

q2->add.x = 30;

q2->add.y = 16;

area -= 3;

ahead->draw(map);

draw();

}

void endit() {

life = 0;

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "紫色获胜");

}

void winit() {

alife = 0;

settextcolor(0xffffff);

settextcolor(0xffffff);

setbkcolor(0x555555);

settextstyle(60, 30, \_T("楷体"));

outtextxy(280, 270, "绿色获胜");

}

bool play() {

if (life == 0) {

return false;

}

if (alife == 0) {

return false;

}

pos ne = head->next\_pos(mes);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h') || (map[ne.x][ne.y] >= 'x'&&map[ne.x][ne.y] <= '{')) {

pos tail = head->tail\_pos();

map[tail.x][tail.y] = '0';

head = head->move(mes);

head->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

head = head->eat(mes);

head->draw(map);

fruit\_num--;

length++;

score += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

else {

endit();

return false;

}

return false;

}

bool aplay() {

if (life == 0) {

return false;

}

if (alife == 0) {

return false;

}

pos ne = ahead->next\_pos(ames);

if (map[ne.x][ne.y] == '0' || (map[ne.x][ne.y] >= 'x'&&map[ne.x][ne.y] <= '{') || (map[ne.x][ne.y] >= 'e'&&map[ne.x][ne.y] <= 'h')) {

pos tail = ahead->tail\_pos();

map[tail.x][tail.y] = '0';

ahead = ahead->move(ames);

ahead->draw(map);

return true;

}

else if (map[ne.x][ne.y] == '7') {

ahead = ahead->eat(ames);

ahead->draw(map);

fruit\_num--;

alength++;

ascore += 2;

if (fruit\_num == 0) {

refresh();

}

return true;

}

else {

winit();

return false;

}

return false;

}

void drawui() {

settextstyle(20, 10, \_T("楷体"));

setbkcolor(0x000000);

char ylen[30] = "绿方长度：";

char ysco[30] = "绿方分数：";

char alen[30] = "紫方长度：";

char asco[30] = "紫方分数：";

char stim[30] = "游戏时间：";

int2char(length, &ylen[10]);

int2char(score, &ysco[10]);

int2char(alength, &alen[10]);

int2char(ascore, &asco[10]);

int2char((int)((double)(now\_t - start\_t) / CLOCKS\_PER\_SEC), &stim[10]);

outtextxy(5, 605, ylen);

outtextxy(165, 605, ysco);

outtextxy(325, 605, alen);

outtextxy(485, 605, asco);

outtextxy(645, 605, stim);

}

void listen() {

struct tm t;

time\_t now, last;

time(&now);

localtime\_s(&t, &now);

bool flag = true;

bool afalg = true;

char temp = 'u';

while (temp != 'q') {

while (!\_kbhit()) {

last = now;

time(&now);

if (last != now) {

if (afalg = aplay()) {

draw();

}

if (flag = play()) {

now\_t = flag ? clock() : end\_t;

draw();

drawui();

}

}

}

temp = \_getch();

temp = flag ? temp : 'q';

if (!flag) {

end\_t = clock();

}

switch (temp) {

case 'w':

if (mes != 'd') {

mes = 'u';

}

break;

case 'a':

if (mes != 'r') {

mes = 'l';

}

break;

case 's':

if (mes != 'u') {

mes = 'd';

}

break;

case 'd':

if (mes != 'l') {

mes = 'r';

}

break;

case 'i':

if (ames != 'd') {

ames = 'u';

}

break;

case 'j':

if (ames != 'r') {

ames = 'l';

}

break;

case 'k':

if (ames != 'u') {

ames = 'd';

}

break;

case 'l':

if (ames != 'l') {

ames = 'r';

}

break;

}

}

back();

}

};

class Btn {

public:

char name[100];

pos left\_up, right\_down;

int color = 0xffffff;

int border\_color = 0x000000;

int font\_color = 0x000000;

bool ifin(pos a) {

if (a.x < left\_up.x || a.x > right\_down.x) {

return false;

}

if (a.y < left\_up.y || a.y > right\_down.y) {

return false;

}

return true;

}

void onclick(pos a, Game1 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void onclick(pos a, Game2 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void onclick(pos a, Game3 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void onclick(pos a, Game4 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void onclick(pos a, Game5 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void onclick(pos a, Game6 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void onclick(pos a, Game7 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void onclick(pos a, Game8 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void onclick(pos a, Game9 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void onclick(pos a, Game10 obj) {

if (ifin(a)) {

obj.init\_game();

obj.init2();

obj.listen();

}

}

void back();

void onclick(pos a) {

if (ifin(a)) {

closegraph();

fstream fp;

fp.open(REC);

if (!fp.is\_open()) {

cerr << "错误！";

exit(-1);

}

int a, c;

char b[100];

system("cls");

for (;;) {

fp >> a >> b >> c;

if (fp.eof()) {

break;

}

switch (a) {

case 1:

cout << "版本：入门版";

break;

case 2:

cout << "版本：进阶版";

break;

case 3:

cout << "版本：高级版";

break;

case 4:

cout << "版本：idea1";

break;

case 5:

cout << "版本：idea2";

break;

case 6:

cout << "版本：idea3";

break;

case 7:

cout << "版本：idea4";

break;

case 8:

cout << "版本：idea6";

break;

default:

a = -1;

}

if (a == -1) {

continue;

}

cout << " 用户名：" << b << " 得分：" << c << endl;

}

cout << "\n按下任意键返回主菜单......";

while (!\_kbhit());

\_getch();

back();

}

}

void draw() {

setfillstyle(BS\_SOLID);

setlinestyle(PS\_DASH | PS\_ENDCAP\_FLAT, 3);

setlinecolor(border\_color);

setfillcolor(color);

fillrectangle(left\_up.x, left\_up.y, right\_down.x, right\_down.y);

settextcolor(font\_color);

setbkcolor(color);

settextstyle((int)(0.7\*(right\_down.y - left\_up.y)), (int)(0.35\*(right\_down.y - left\_up.y)), \_T("楷体"));

double temp1, temp2;

temp1 = 0.5\*(left\_up.x + right\_down.x) - 0.5\*strlen(name)\*0.35\*(right\_down.y - left\_up.y);

temp2 = left\_up.y + 0.15\*(right\_down.y - left\_up.y);

outtextxy((int)temp1, (int)temp2, name);

}

void set\_Btn(const char name[100], int left, int up, int right, int down) {

strcpy(this->name, name);

left\_up.x = left;

left\_up.y = up;

right\_down.x = right;

right\_down.y = down;

}

};

class Menu {

public:

Btn \*btns;

int btn\_num = 12;

void init\_menu() {

initgraph(900, 600);

settextcolor(0xffffff);

settextstyle(60, 30, \_T("楷体"));

outtextxy(270, 30, "贪吃蛇大作业");

settextstyle(30, 15, \_T("楷体"));

outtextxy(345, 120, "1950000 一二三");

btns = new Btn[12];

btns[0].set\_Btn("入门", 60, 180, 240, 240);

btns[1].set\_Btn("进阶", 60, 270, 240, 330);

btns[2].set\_Btn("高级", 60, 360, 240, 420);

btns[3].set\_Btn("得分榜", 60, 450, 240, 510);

btns[4].set\_Btn("idea1", 360, 180, 540, 240);

btns[5].set\_Btn("idea2", 360, 270, 540, 330);

btns[6].set\_Btn("idea3", 360, 360, 540, 420);

btns[7].set\_Btn("idea4", 360, 450, 540, 510);

btns[8].set\_Btn("idea6", 660, 180, 840, 240);

btns[9].set\_Btn("人机", 660, 270, 840, 330);

btns[10].set\_Btn("双人", 660, 360, 840, 420);

btns[11].set\_Btn("尚未开发", 660, 450, 840, 510);

for (int i = 0; i < btn\_num; i++) {

btns[i].draw();

}

}

void listen() {

MOUSEMSG m{ 0 };

BOOL flag = TRUE;

pos temp;

Game1 g1;

Game2 g2;

Game3 g3;

Game4 g4;

Game5 g5;

Game6 g6;

Game7 g7;

Game8 g8;

Game9 g9;

Game10 g10;

while (flag)

{

if (MouseHit())m = GetMouseMsg(); // 获取鼠标信息

TCHAR s[20]; // EasyX设置文字需要的字符串变量类型

switch (m.uMsg)

{

case WM\_LBUTTONDOWN:

temp.x = m.x;

temp.y = m.y;

btns[0].onclick(temp, g1);

btns[1].onclick(temp, g2);

btns[2].onclick(temp, g3);

btns[3].onclick(temp);

btns[4].onclick(temp, g4);

btns[5].onclick(temp, g5);

btns[6].onclick(temp, g6);

btns[7].onclick(temp, g7);

btns[8].onclick(temp, g8);

btns[9].onclick(temp, g9);

btns[10].onclick(temp, g10);

break;

}

}

}

};

void Game::back() {

Menu haha;

haha.init\_menu();

haha.listen();

}

void Btn::back() {

Menu haha;

haha.init\_menu();

haha.listen();

}

int main() {

srand(time(0));

Menu haha;

haha.init\_menu();

haha.listen();

}

//int main() {

//

// initgraph(800, 600); // 初始化窗口

// MOUSEMSG m{ 0 }; // 鼠标信息结构体

// BOOL flag = TRUE;

// while (flag)

// {

// if (MouseHit())m = GetMouseMsg(); // 获取鼠标信息

// TCHAR s[20]; // EasyX设置文字需要的字符串变量类型

// switch (m.uMsg)

// {

// case WM\_LBUTTONDOWN:

// // 左键按下，在当下位置画一个圆

// setfillcolor(RED);

// solidcircle(m.x, m.y, 20);

// initgraph(600, 800);

// break;

// case WM\_RBUTTONDOWN:

// // 右键按下，在当下位置画一个椭圆

// setfillcolor(RGB(255, 0, 255));

// solidellipse(m.x - 40, m.y - 20, m.x + 40, m.y + 20);

// break;

// case WM\_MOUSEMOVE:

// // 鼠标移动，在窗口左上角显示当前坐标

// setfillcolor(BLACK); // 黑色矩形覆盖上次坐标记录

// solidrectangle(0, 0, 75, 20);

// \_stprintf\_s(s, \_T("[%d,%d]"), m.x, m.y); // 格式化字符串

// outtextxy(0, 0, s);

// break;

// case WM\_LBUTTONDBLCLK:

// // 左键双击退出循环

// flag = FALSE;

// break;

// }

// }

//

// closegraph();

//

// return 0;

//}