哈爾濱工業大學（深圳）
Harbin Institute of Technology Shenzhen

《创新训练课 B》课程设计报告

学院：机电工程与自动化学院

题目：贪吃蛇小游戏

班级：18 级自动化 2 班

姓名：陈家苇

学号：180510113

教师：吴晓军

上交日期：2020.9.1

修订历史记录

日期	版本	说明	作者
2020/7/24	1.0	贪吃蛇小游戏开发	陈家苇

目 录

1 引言

1.1 编写目的

为了实现对经典小游戏“贪吃蛇”的基本功能如获取食物、躲避障碍物等功能以及 UI 界面的实现，设计此游戏。

这份报告对贪吃蛇小游戏做了简单的用户需求和功能需求分析，明确了程序设计应具有的设计了系统的总体架构，并对系统进行了模块划分，设计了程序流程，描述了主要功能的实现和具体的操作过程，记录了调试过程和调试结果，分析遇到的问题和解决方法，最后总结此课程设计作业的心得体会。

1.2 背景

1976 年，Gremlin 平台推出了一款经典街机游戏 Blockade。游戏中，两名玩家分别控制一个角色在屏幕上移动，所经之处砌起围栏。角色只能向左、右方向 90 度转弯，游戏目标保证让对方先撞上屏幕或围栏。两条每走一步都会长大的贪吃蛇比谁后结束游戏，玩家要做的就是避免撞上障碍物和越来越长的身体。Blockade 很受欢迎，类似的游戏先后出现在 Atari2600、TRS-80、苹果 2 等早期游戏机、计算机上。但真正让这种游戏形式红遍全球的还是 21 年后随诺基亚手机走向世界的贪吃蛇游戏——Snake。诺基亚为贪食蛇的推广做出了贡献，其贪食蛇的主要版本从“Snake”到 2008 年的第 8 版“Snakes Subsonic”，后来还推出了 3D 版本的贪吃蛇游戏。

1.3 定义

游戏定义：用游戏按钮或手柄把手的上下左右控制蛇的方向，寻找食物，每吃一口食物就能得到一定的积分，而且蛇的身子会越吃越长，身子越长玩的难度就越大，不能碰墙，不能咬到自己的身体，更不能咬自己的尾巴，直到游戏结束。

程序函数名词定义：

initSnake	drawSnake	moveSnake	keyDown
初始化蛇	画蛇	移动蛇	按键交互（控制蛇移动）
initFood	Initbarrier	drawFood	drawBarrier
初始化食物	初始化障碍物	画食物	画障碍物
eatFood	showgrade	gameOver	
判定蛇吃到食物	玩家得分显示	判定游戏结束	

1.4 参考资料

（1）《C++面向对象程序设计（第 2 版）》谭浩强 北京：清华大学出版社

（2）EasyX 在线文档

2 任务概述

主要目标为设计一款贪吃蛇小游戏。其游戏规则和功能介绍大致如下：

（1）用一个小矩形块表示蛇的一节身体，身体每长一节，增加一个矩形块，蛇头用两节表示（本程序设定为灰色），蛇的身体用金色的矩形块表示，游戏初始化蛇为三节（即两节灰色+一节金色）。

（2）移动时从蛇头开始，不能反向移动。如果不按任意键，蛇自行在当前方向上前移，但按下有效方向键后，蛇头朝着该方向移动，一步移动一节身体。

（3）每次蛇吃到食物，身体会增长一节矩形块，在游戏中的效果即在蛇尾增加一节金色的矩形块蛇身，玩家的游戏得分将会增加 10 分。

（4）每次游戏启动，将在地图中随机设置 4 个障碍物（本程序设定为朱红色），蛇头碰到四周墙壁、蛇身体、障碍物时游戏结束，因此蛇必须要避免撞墙、触碰到自己的身体，并注意绕开障碍物。

（5）为了提高游戏的奖励和娱乐性质，设定玩家每获得 100 分，地图中将会多出现一个食物，但受限于游戏界面画布的大小，设定食物总数量上限为 10 个，此外为游戏设置了背景音乐以提高玩家的游戏体验。

（6）游戏界面的设定：启动程序——欢迎界面——玩家手册——按任意键游戏开始——游戏结束，提示“Game over”和玩家游戏的最终得分。

3 需求分析

3.1 用户需求分析

一．实用需求

该贪吃蛇小游戏保持了经典贪吃蛇游戏的基本规则，为了提高游戏的娱乐性，有一些规则上的改变，让玩家既能体会到经典游戏又能有可玩性。

二．交互需求

该游戏界面十分简洁，背景颜色以及蛇体、食物、障碍物等配色感官舒适，不会让用户产生冗杂感。

游戏设置有背景音乐，可提高玩家的游戏体验。

游戏对于玩家的得分有明确的提示信息，便于玩家掌握游戏进度。

玩家每获得 100 分，地图中将会多出现一个食物，食物数量的提高可以加快玩家操作游戏的速度，加强游戏娱乐性。

三．系统操作可行性需求

1. 系统操作简单，说明具体清晰。在游戏的开始界面有欢迎界面，并附有《玩家须知》帮助玩家熟悉游戏规则，使游戏简单容易上手。

2. 为了满足不同玩家对键盘使用习惯的不同，本游戏的按键交互操作对于大小写字母形式的“WSAD”和“↑ ↓ ←→”两套“上下左右”按键均使用，玩家甚至可以左右手同时配合操作。

3. 容错能力。系统具有一定的容错和抗干扰能力，在非设备和硬件问题的情况下，基本能正常运行。针对蛇头方向快速移动变换可能造成的误触等问题进行

了解决，保证玩家舒适的游戏体验。

3.2 运行环境

1. 一台 586 以上的微机及兼容
2. 内存 16MB 以上
3. 彩色显示器一台
4. Windows98 以上操作系统

4 功能及操作介绍

4.1 操作

➢ // 初始化蛇

```
void initSnake()
{
    point xy;
    xy.x = 20;
    xy.y = 0;
    snake.xy.push_back(xy);
    snake.color.push_back(RGB(112, 128, 144));  //头部[0]颜色为石板灰
    xy.x = 10;
    xy.y = 0;
    snake.xy.push_back(xy);
    snake.color.push_back(RGB(112, 128, 144));  //头部[1]颜色为石板灰，两块[0][1]一起代表头部
    xy.x = 0;
    xy.y = 0;
    snake.xy.push_back(xy);
    snake.color.push_back(RGB(255, 215, 0));    // 蛇身体的颜色为金色
    snake.num = 3;
    snake.position = right;
}
```

➢ // 画蛇

```
void drawSnake()
{
    for (int i = 0; i < snake.num; i++)
    {
        setfillcolor(snake.color[i]);
        fillrectangle(snake.xy[i].x, snake.xy[i].y, snake.xy[i].x + 10, snake.xy[i].y + 10);
    }
}
```

➢ // 移动蛇

```
void moveSnake()
{
    // 将预留节设置为未移动前的尾节
```

```
    snake.next = snake.xy[snake.num - 1];
    // 将除蛇头以外的节移动到它的前面一节
    for (int i = snake.num - 1; i >= 1; i--)
        snake.xy[i] = snake.xy[i - 1];
    // 根据当前移动方向移动蛇头
    switch (snake.position)
    {
    case right:
        snake.xy[0].x += 10;
        break;
    case left:
        snake.xy[0].x -= 10;
        break;
    case up:
        snake.xy[0].y -= 10;
        break;
    case down:
        snake.xy[0].y += 10;
    }
}
```

➢ //按键交互（控制蛇移动）

```
void keyDown()
{
    char userKey = _getch();
    if (userKey == -32)             // 表明这是方向键
        userKey = -_getch();        // 获取具体方向，并避免与其他字母的 ASCII 冲突
    switch (userKey)
    {
    case 'w':
    case 'W':
    case -72:
        if (snake.position != down)
            snake.position = up;
        break;
    case 's':
    case 'S':
    case -80:
        if (snake.position != up)
            snake.position = down;
        break;
    case 'a':
    case 'A':
    case -75:
        if (snake.position != right)
```

```
            snake.position = left;
        break;
    case 'd':
    case 'D':
    case -77:
        if (snake.position != left)
            snake.position = right;
        break;
    }
}
```

➢ // 初始化食物

```
void initFood(int num /* 食物编号 */)
{
    food.fdxy[num].x = rand() % 80 * 10;
    food.fdxy[num].y = rand() % 60 * 10;
    for (int i = 0; i < snake.num; i++)
        if (food.fdxy[num].x == snake.xy[i].x && food.fdxy[num].y == snake.xy[i].y)
    // 避免食物生成在蛇身上
        {
            food.fdxy[num].x = rand() % 80 * 10;
            food.fdxy[num].y = rand() % 60 * 10;
        }
}
```

➢ //初始化障碍物（4个）

```
void initbarrier1()
{
    barrier1.brxy.x = rand() % 70 * 10;
    barrier1.brxy.y = rand() % 50 * 10;
    for (int i = 0; i < snake.num; i++)
        if (barrier1.brxy.x == snake.xy[i].x && barrier1.brxy.y == snake.xy[i].y ||
barrier1.brxy.x == food.fdxy[i].x && barrier1.brxy.y == food.fdxy[i].y)
            // 避免障碍生成在蛇上或者与食物重合或者障碍物彼此重合
        {
            barrier1.brxy.x = rand() % 70 * 10;
            barrier1.brxy.y = rand() % 50 * 10;
        }
}
void initbarrier2()
{
    barrier2.brxy.x = rand() % 70 * 10;
    barrier2.brxy.y = rand() % 50 * 10;
    for (int i = 0; i < snake.num; i++)
        if (barrier2.brxy.x == snake.xy[i].x && barrier2.brxy.y == snake.xy[i].y ||
barrier2.brxy.x == food.fdxy[i].x && barrier2.brxy.y == food.fdxy[i].y ||
```

```
            barrier2.brxy.x == barrier1.brxy.x && barrier2.brxy.y == barrier1.brxy.y)
            // 避免障碍生成在蛇上或者与食物重合或者障碍物彼此重合
        {
            barrier2.brxy.x = rand() % 70 * 10;
            barrier2.brxy.y = rand() % 50 * 10;
        }
}
void initbarrier3()
{
    barrier3.brxy.x = rand() % 70 * 10;
    barrier3.brxy.y = rand() % 50 * 10;
    for (int i = 0; i < snake.num; i++)
        if (barrier3.brxy.x == snake.xy[i].x && barrier3.brxy.y == snake.xy[i].y ||
barrier3.brxy.x == food.fdxy[i].x && barrier3.brxy.y == food.fdxy[i].y ||
            barrier3.brxy.x == barrier1.brxy.x && barrier3.brxy.y == barrier1.brxy.y ||
barrier3.brxy.x == barrier2.brxy.x && barrier3.brxy.y == barrier2.brxy.y)
            // 避免障碍生成在蛇上或者与食物重合或者障碍物彼此重合
        {
            barrier3.brxy.x = rand() % 70 * 10;
            barrier3.brxy.y = rand() % 50 * 10;
        }
}
void initbarrier4()
{
    barrier4.brxy.x = rand() % 70 * 10;
    barrier4.brxy.y = rand() % 50 * 10;
    for (int i = 0; i < snake.num; i++)
        if (barrier4.brxy.x == snake.xy[i].x && barrier4.brxy.y == snake.xy[i].y ||
barrier4.brxy.x == food.fdxy[i].x && barrier4.brxy.y == food.fdxy[i].y ||
            barrier4.brxy.x == barrier1.brxy.x && barrier4.brxy.y == barrier1.brxy.y ||
barrier4.brxy.x == barrier2.brxy.x && barrier4.brxy.y == barrier2.brxy.y ||
            barrier4.brxy.x == barrier3.brxy.x && barrier4.brxy.y == barrier3.brxy.y)
            // 避免障碍生成在蛇上或者与食物重合或者障碍物彼此重合
        {
            barrier4.brxy.x = rand() % 70 * 10;
            barrier4.brxy.y = rand() % 50 * 10;
        }
}
➢   // 画食物
void drawFood()
{
    for (int i = 0; i <= food.num - 1; i++)
    {
        setfillcolor(food.color = RGB(255, 215, 0));// 食物：金色
```

```
            fillrectangle(food.fdxy[i].x, food.fdxy[i].y, food.fdxy[i].x + 10, food.fdxy[i].y
+ 10);
        }
}
```

➢ //画障碍物

```
void drawBarrier()
{
    setfillcolor(barrier1.color = RGB(240, 65, 85)); // 障碍物：朱红色
    fillrectangle(barrier1.brxy.x, barrier1.brxy.y, barrier1.brxy.x + 10, barrier1.brxy.y
+ 10);
    setfillcolor(barrier2.color = RGB(240, 65, 85));
    fillrectangle(barrier2.brxy.x, barrier2.brxy.y, barrier2.brxy.x + 10, barrier2.brxy.y
+ 10);
    setfillcolor(barrier3.color = RGB(240, 65, 85));
    fillrectangle(barrier3.brxy.x, barrier3.brxy.y, barrier3.brxy.x + 10, barrier3.brxy.y
+ 10);
    setfillcolor(barrier4.color = RGB(240, 65, 85));
    fillrectangle(barrier4.brxy.x, barrier4.brxy.y, barrier4.brxy.x + 10, barrier4.brxy.y
+ 10);
}
```

➢ // 蛇头吃到食物

```
void eatFood()
{
    for (int i = 0; i <= food.num - 1; i++)
        if (snake.xy[0].x == food.fdxy[i].x && snake.xy[0].y == food.fdxy[i].y)
        {
            snake.num++;
            snake.xy.push_back(snake.next);          // 新增一个节到预留位置
            snake.color.push_back(food.color);       // 将新增节的颜色设置为当前吃掉食
物的颜色，即金色
            food.grade += 10;                //吃食物之后，分数加10分
            initFood(i);
            if (food.num < 10 && food.grade % 100 == 0 && food.grade != 0)
            {
                food.num++;       // 每得 100 分，增加一个食物，但食物总数不超过 10 个
                initFood(food.num - 1);              // 初始化新增加的食物
            }
            break;
        }
}
```

➢ // 分数显示

```
void showgrade()
{
    wchar_t grade[20] = L"";
```

```
    swprintf_s(grade, L"分数：%d", food.grade);
    outtextxy(650, 50, grade);                          //显示分数的位置
    settextcolor(RGB(0, 0, 0));                         //显示字符字体设为金色(255,215,0)
    settextstyle(20, 0, L"分数：%d");                    //设置分数显示的字符为高度20像素
    setbkmode(TRANSPARENT);                             // 设置文字输出模式为透明

}
```

➢ // 游戏结束的判断

```
bool gameOver()
{
    // 撞墙的情况，将墙向外扩展一圈，否则蛇无法到达地图（墙壁）边缘
    if (snake.xy[0].y <= -10 && snake.position == up)             return true;
    if (snake.xy[0].y + 10 >= 610 && snake.position == down)  return true;
    if (snake.xy[0].x <= -10 && snake.position == left)           return true;
    if (snake.xy[0].x + 10 >= 810 && snake.position == right)return true;
    // 蛇头撞自己的情况
    for (int i = 1; i < snake.num; i++)
    {
        if (snake.xy[0].x <= snake.xy[i].x + 10 && snake.xy[0].x >= snake.xy[i].x &&
snake.xy[0].y == snake.xy[i].y&&snake.position == left)
            return true;
        if (snake.xy[0].x + 10 >= snake.xy[i].x && snake.xy[0].x + 10 <= snake.xy[i].x +
10 && snake.xy[0].y == snake.xy[i].y && snake.position == right)
            return true;
        if (snake.xy[0].y <= snake.xy[i].y + 10 && snake.xy[0].y >= snake.xy[i].y &&
snake.xy[0].x == snake.xy[i].x && snake.position == up)
            return true;
        if (snake.xy[0].y + 10 >= snake.xy[i].y && snake.xy[0].y + 10 <= snake.xy[i].y +
10 && snake.xy[0].x == snake.xy[i].x && snake.position == down)
            return true;
    }
    //撞到障碍物1的情况
    if (snake.xy[0].x <= barrier1.brxy.x + 10 && snake.xy[0].x >= barrier1.brxy.x &&
snake.xy[0].y == barrier1.brxy.y&&snake.position == left)
        return true;
    if (snake.xy[0].x + 10 >= barrier1.brxy.x && snake.xy[0].x + 10 <= barrier1.brxy.x +
10 && snake.xy[0].y == barrier1.brxy.y && snake.position == right)
        return true;
    if (snake.xy[0].y <= barrier1.brxy.y + 10 && snake.xy[0].y >= barrier1.brxy.y &&
snake.xy[0].x == barrier1.brxy.x && snake.position == up)
        return true;
    if (snake.xy[0].y + 10 >= barrier1.brxy.y && snake.xy[0].y + 10 <= barrier1.brxy.y +
10 && snake.xy[0].x == barrier1.brxy.x && snake.position == down)
        return true;
```

```
    //撞到障碍物2
    //撞到障碍物3
    //撞到障碍物4
    //（上述三种情况与撞到障碍物1相同）
    return false;
}
```

➢ `//加载背景音乐`

```
    MCI_OPEN_PARMS mciOpen;
    mciOpen.lpstrDeviceType = _T("mpegvideo");
    mciOpen.lpstrElementName = _T("D://background.mp3");//背景音乐的路径
    MCIERROR mciError = mciSendCommand(0, MCI_OPEN, MCI_OPEN_TYPE | MCI_OPEN_ELEMENT,
(DWORD)&mciOpen);
    if (mciError)
    {
        TCHAR buf[128] = { 0 };
        mciGetErrorString(mciError, buf, 128);
        printf("%s/n", buf);
    }
    UINT DeviceID = mciOpen.wDeviceID;
    MCI_PLAY_PARMS mciPlay;
    mciError = mciSendCommand(DeviceID, MCI_PLAY, 0, (DWORD)&mciPlay);
    if (mciError)
    {
        printf("send MCI_PLAY command failed/n");
    }
```

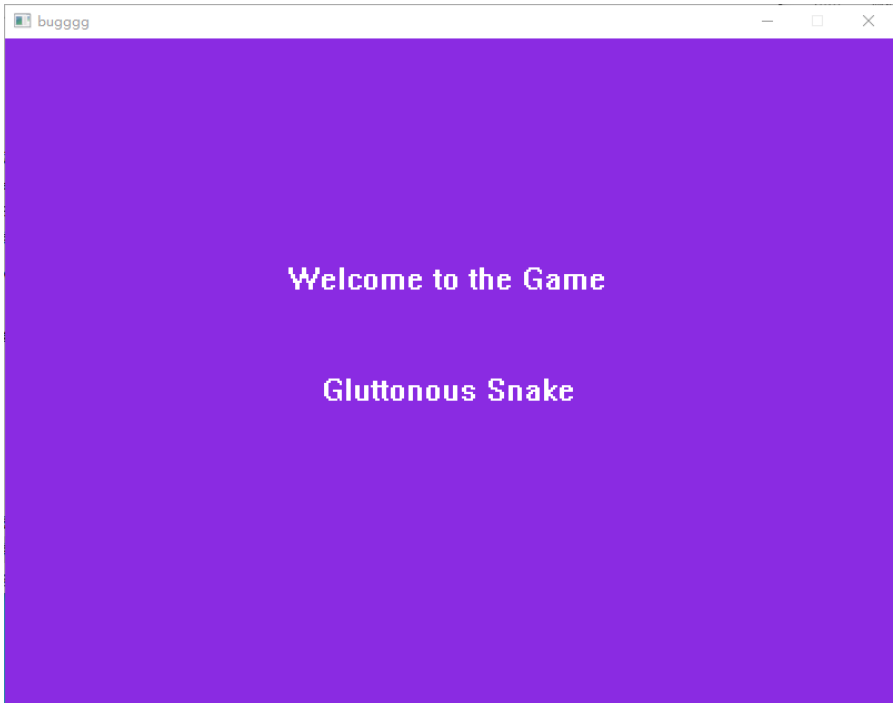
4.2 功能

本游戏只设计了服务于用户的部分，界面简洁，可实现玩家查看玩家须知，游戏得分、游戏操作的功能。

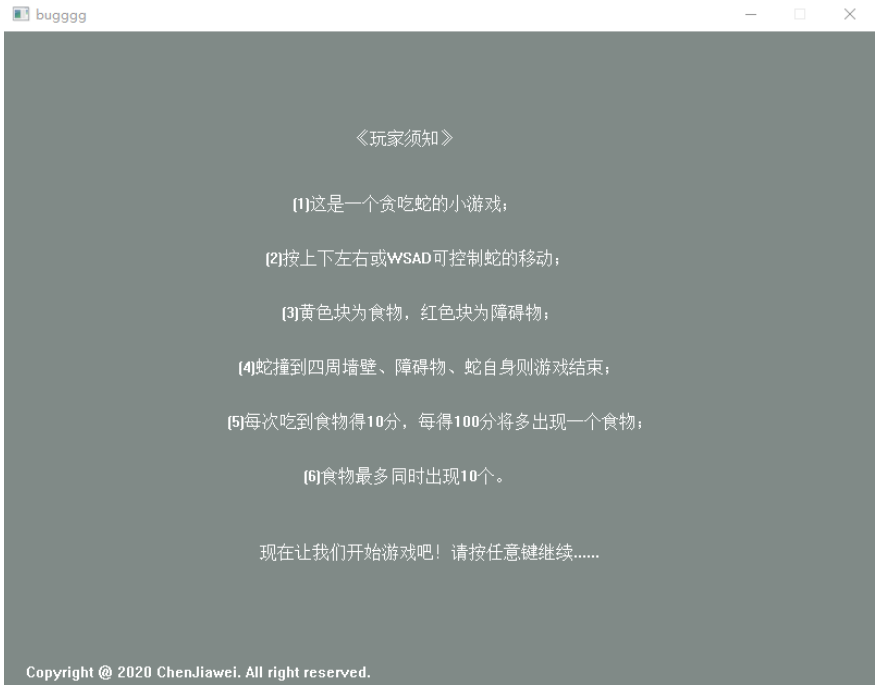
5 系统设计

5.1 总体架构设计

1.运行程序，首先进入欢迎界面，界面显示如下：



2. 接下来进入《玩家手册》提示界面，显示游戏规则和操作规范，玩家可以通过按任意键开始游戏。界面显示如下：



3. 玩家游戏操作界面。涉及食物的随机生成、障碍物的随机生成、分数的累计、达到一定分数时的食物增加与新食物的随机生成、食物的获取与蛇身体的增长、对玩家撞到墙壁/障碍物/自身身体的判定、背景音乐的加载等。

4. 游戏结束，提示结束语与玩家最终得分。界面显示如下：

5.2 模块分析与设计

- 界面菜单模块

```
//欢迎界面的背景颜色：浅紫色
setfillcolor(RGB(138, 43, 226));
solidrectangle(0, 0, 800, 600);
LOGFONT z;
gettextstyle(&z);                    // 获取当前字体设置
z.lfHeight = 30;                     // 设置字体高度30
settextstyle(&z);
setbkmode(TRANSPARENT);              // 设置字体为透明
outtextxy(255, 200, _T("Welcome to the Game"));
outtextxy(285, 300, _T("Gluttonous Snake"));
Sleep(1800);
cleardevice();

//玩家使用手册界面的背景颜色：浅灰
setfillcolor(RGB(128,138,135));
solidrectangle(0, 0, 800, 600);
LOGFONT p;
gettextstyle(&p);                    // 获取当前字体设置
p.lfHeight = 20;                     // 设置字体高度20
settextstyle(&p);
setbkmode(TRANSPARENT);                  // 设置字体为透明
outtextxy(320, 90, _T("《玩家须知》"));
outtextxy(265, 150, _T("(1)这是一个贪吃蛇的小游戏；"));
outtextxy(240, 200, _T("(2)按上下左右或WSAD可控制蛇的移动；"));
outtextxy(255, 250, _T("(3)黄色块为食物，红色块为障碍物；"));
```

```
outtextxy(215, 300, _T("(4)蛇撞到四周墙壁、障碍物、蛇自身则游戏结束；"));
outtextxy(205, 350, _T("(5)每次吃到食物得10分，每得100分将多出现一个食物；"));
outtextxy(275, 400, _T("(6)食物最多同时出现10个。"));
outtextxy(235, 470, _T("现在让我们开始游戏吧！请按任意键继续......"));
outtextxy(20, 580, _T("Copyright @ 2020 ChenJiawei. All right reserved."));
system("pause");
cleardevice();
```

- 游戏操作界面

```
// 初始化蛇
```

void initSnake()：对游戏开始时的蛇进行设置，设置蛇由三节组成，蛇头为两节，颜色为石板灰，蛇身为一节，颜色为金色，从画面的左上角位置出现，初始速度方向为向右（right）；

```
// 画蛇
```

void drawSnake()：对蛇结构体中蛇节数的每一段画一个正方形，像素为 10*10；

```
// 移动蛇
```

void moveSnake()：将预留节设置为未移动前的尾节，将除蛇头以外的节移动到它的前面一节，并根据当前移动方向移动蛇头（分为上下左右四种情况）；

```
// 按键交互（控制蛇的移动方向）
```

void keyDown()：通过_getch()获取键盘输入，在上下左右四种情况下分别获得键盘的方向控制输入，每种情况允许的输入有三种，即大、小写字母的键盘方向输入“WSAD”、“wsad”和键盘的“↑ ↓ ←→”键，判断方式是用这些键的 ASCII 码；

```
// 初始化食物
```

void initFood(int num)：函数中的参数 num 为食物编号，num 初始化为 1（即一个食物），随着游戏的进行，num 会增加，但总数不超过 10 个；食物通过随机函数的方式产生，且需要判断食物生成的位置，即食物不能与蛇身重合，否则将重新执行随机函数操作；

```
//初始化障碍物（4个）
```

void initbarrier1()、void initbarrier2()、void initbarrier3()、void initbarrier4()：

游戏默认定义四个障碍物，障碍物通过随机函数的方式出现，像素为 10*10 的矩形，且需要判断障碍物生成的位置，即避免障碍生成在蛇上、与食物重合或四个障碍物彼此间重合，若重合则重新执行随机函数操作；

```
// 画食物
```

void drawFood()：食物颜色设定为金色，RGB 值为(255, 215, 0)，每个食物为像素 10*10 的矩形；

```
//画障碍物
```

void drawBarrier()：障碍物颜色设定为朱红色，RGB 值为(240, 65, 85)，每个障碍物为像素 10*10 的矩形，

```
// 吃食物
```

void eatFood()：当蛇头的坐标与食物的坐标重合，即判断蛇吃到了食物，此时蛇的节数+1，得分+10，被吃掉的食物重新初始化（即更新位置）；此外需判断得分是否为 100 的整数倍且食物总数是否小于等于 10，如果是，则增加一个食物，且对此增加的食物作初始化操作；

```
// 分数显示
```

void showgrade()：在画面的右上角位置设置得分显示，高度为 20 像素，字形显示透明“TRANSPARENT”（防止干扰画面中物体观察）；

- 游戏结束

```
// 游戏结束的判断函数
```

bool gameOver()：下列情况下，均判断蛇死亡

```
// 撞墙
// 蛇头撞自己
//撞到障碍物1
//撞到障碍物2
//撞到障碍物3
//撞到障碍物4
```

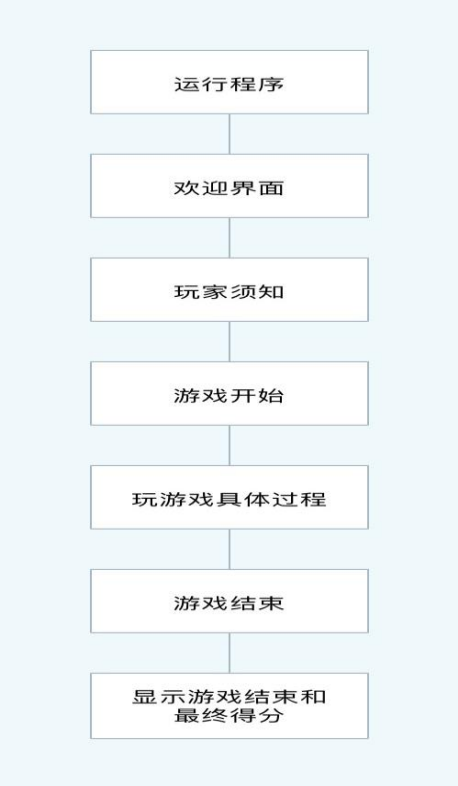
对于每一种情况，需要分为四种小情况，即蛇是从何方向而来，这样可以清楚的写出左边判断的条件；

- 游戏结束提示界面

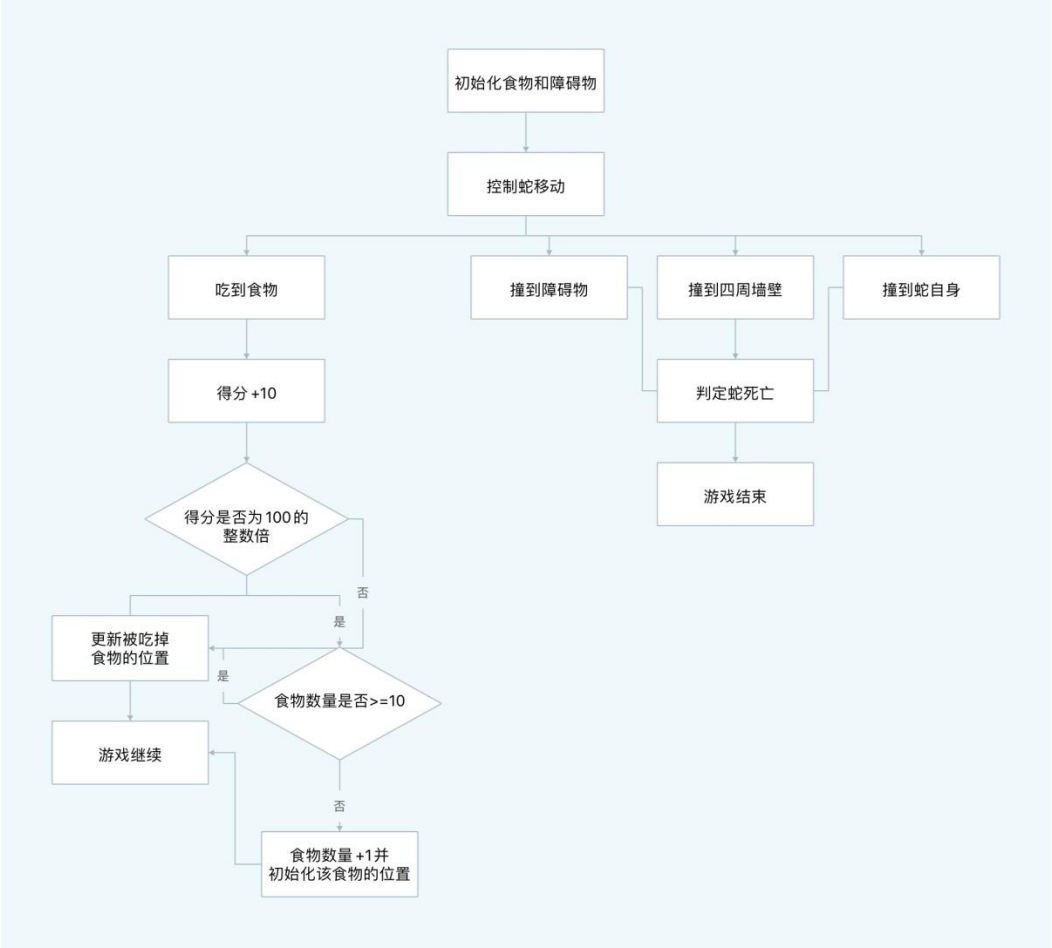
```
//游戏结束界面的背景颜色：绿色
setfillcolor(RGB(0, 153, 0));
solidrectangle(0, 0, 800, 600);
wchar_t overout[50] = L"";
//在画面的中间位置输出"Game over"提示语句和总得分
swprintf_s(overout, L"Game Over!您的最终得分为：%d", food.grade);
outtextxy(260, 250, overout);                    //显示结束语和得分的位置
FlushBatchDraw();                                //结束批量绘图
Sleep(1000);
// 按任意键退出并关闭图形窗口
_getch();
closegraph();
return 0;
```

5.3 软件结构（流程图）

（1）游戏主流程图（框架）：


运行程序
欢迎界面
玩家须知
游戏开始
玩游戏具体过程
游戏结束
显示游戏结束和
最终得分

（2）游戏进行中流程：


初始化食物和障碍物
控制蛇移动
吃到食物
撞到障碍物
撞到四周墙壁
撞到蛇自身
得分 +10
判定蛇死亡
游戏结束
得分是否为100的
整数倍
否
是
更新被吃掉
食物的位置
是
食物数量是否>=10
游戏继续
否
食物数量 +1并
初始化该食物的位置

6 调试与测试

6.1 调试过程

1、初始调试程序的过程中，更新蛇的位置，会出现画面闪烁，调试过程发现采取批量绘图＋及时清屏的操作可以避免闪烁，且清屏时间即反映蛇运动的速度（其实游戏难度可以设置为不同的等级，即难度越高蛇的运动速度越快，此程序由于在一周内完成所以没有进行此环节的设置），实现语句如下：

```
Sleep(100);                 //对于游戏蛇的运动速度的设置，数字越大速度越慢
BeginBatchDraw();           // 开始批量绘图，作用是避免闪烁
cleardevice();              // 清屏
```

2、对不同界面所显示背景颜色、字体改变高度等参数的调试：

由于欢迎界面、玩家须知界面、游戏结束界面的显示背景颜色、文字的设计不同，因此需要改变参数，否则一开始定义背景和文字参数之后，所有页面的输出背景和文字参数均相同。

经过调试之后，在每次需要改变背景和文字参数的界面显示的初始加上设置的语句即可，举例如下：

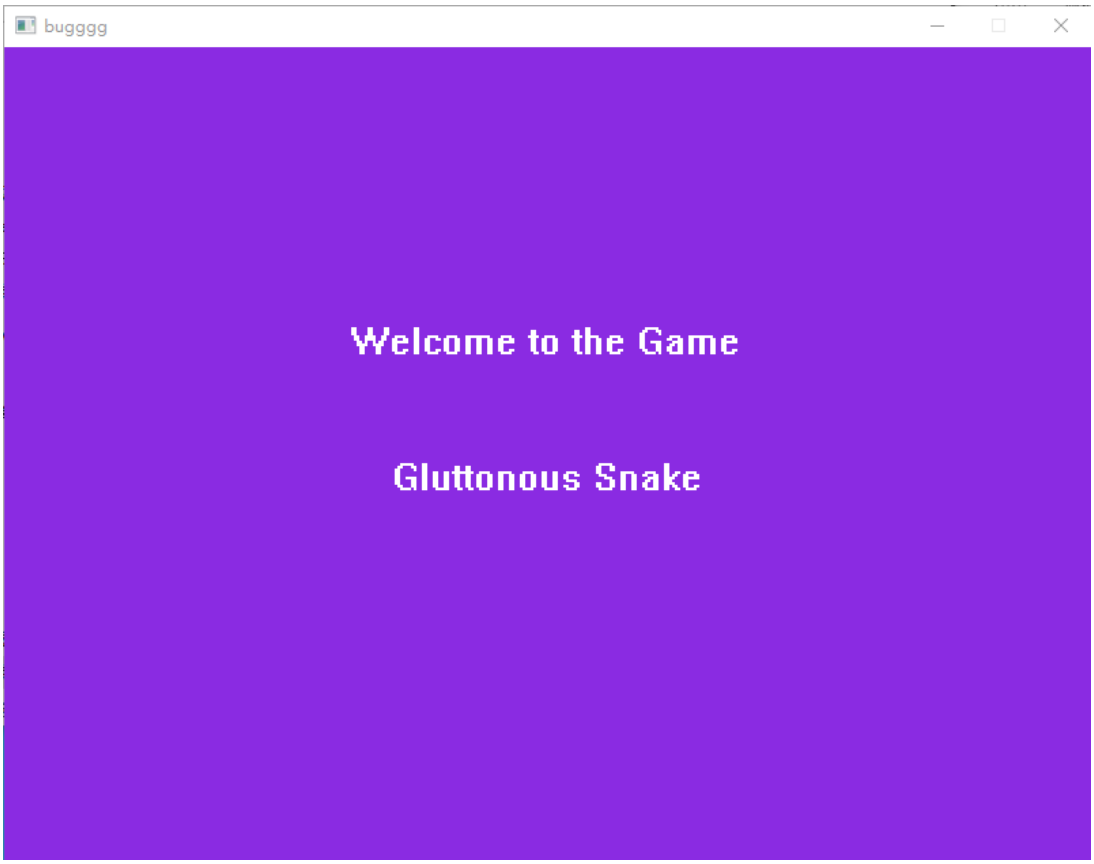
```
（1）setfillcolor(RGB(138, 43, 226));    //欢迎界面的背景颜色：浅紫色
    solidrectangle(0, 0, 800, 600);
    LOGFONT z;
    gettextstyle(&z);                   // 获取当前字体设置
    z.lfHeight = 30;                    // 设置字体高度
    settextstyle(&z);
    setbkmode(TRANSPARENT);             // 设置字体为透明
    outtextxy(255, 200, _T("Welcome to the Game"));
    outtextxy(285, 300, _T("Gluttonous Snake"));
    Sleep(1800);                        // 延迟显示1.8s
    cleardevice();                      // 清屏
（2）setfillcolor(RGB(128,138,135));    //玩家使用手册界面的背景颜色：浅灰
    solidrectangle(0, 0, 800, 600);
    LOGFONT p;
    gettextstyle(&p);                   // 获取当前字体设置
    p.lfHeight = 20;                    // 设置字体高度
    settextstyle(&p);
    setbkmode(TRANSPARENT);             // 设置字体为透明
    outtextxy(320, 90, _T("《玩家须知》"));
    ……
    system("pause");                    // “暂停”，按任意键继续下一语句
    cleardevice();                      // 清屏
（3）//游戏操作界面
    setbkcolor(RGB(100, 149, 237));     //整体游戏背景颜色为矢车菊蓝
    ……
（4）setfillcolor(RGB(0, 153, 0));      //游戏结束界面的背景颜色：绿色
```

```
    solidrectangle(0, 0, 800, 600);
    wchar_t overout[50] = L"";
    swprintf_s(overout, L"Game Over!您的最终得分为：%d", food.grade); //输出"Game over"提示语句和总得分
    outtextxy(260, 250, overout);       //显示结束语和得分的位置
    FlushBatchDraw();                   //结束批量绘图
    Sleep(1000);                        // 延迟
```

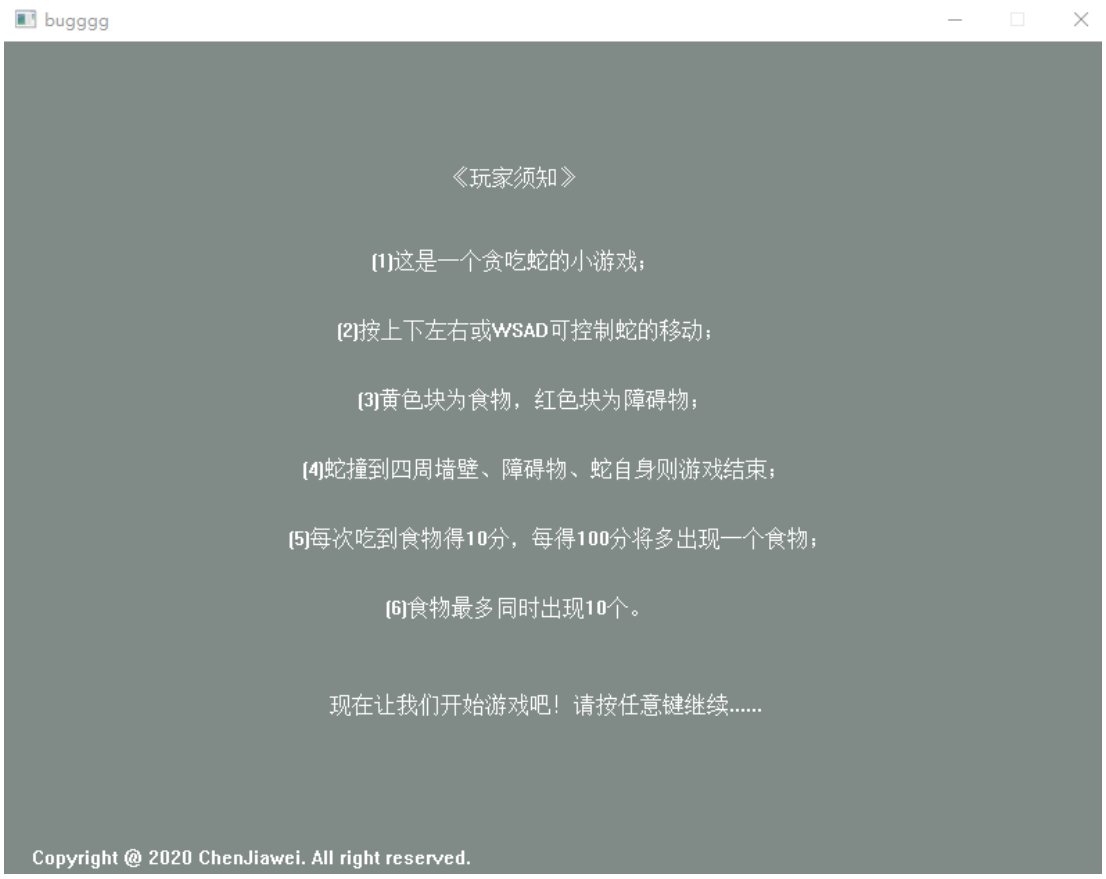
3、蛇体移动功能设置，避免玩家按下与当前蛇运动方向相反的按钮造成误判蛇死亡，经过调试此问题已解决。

6.2 测试结果

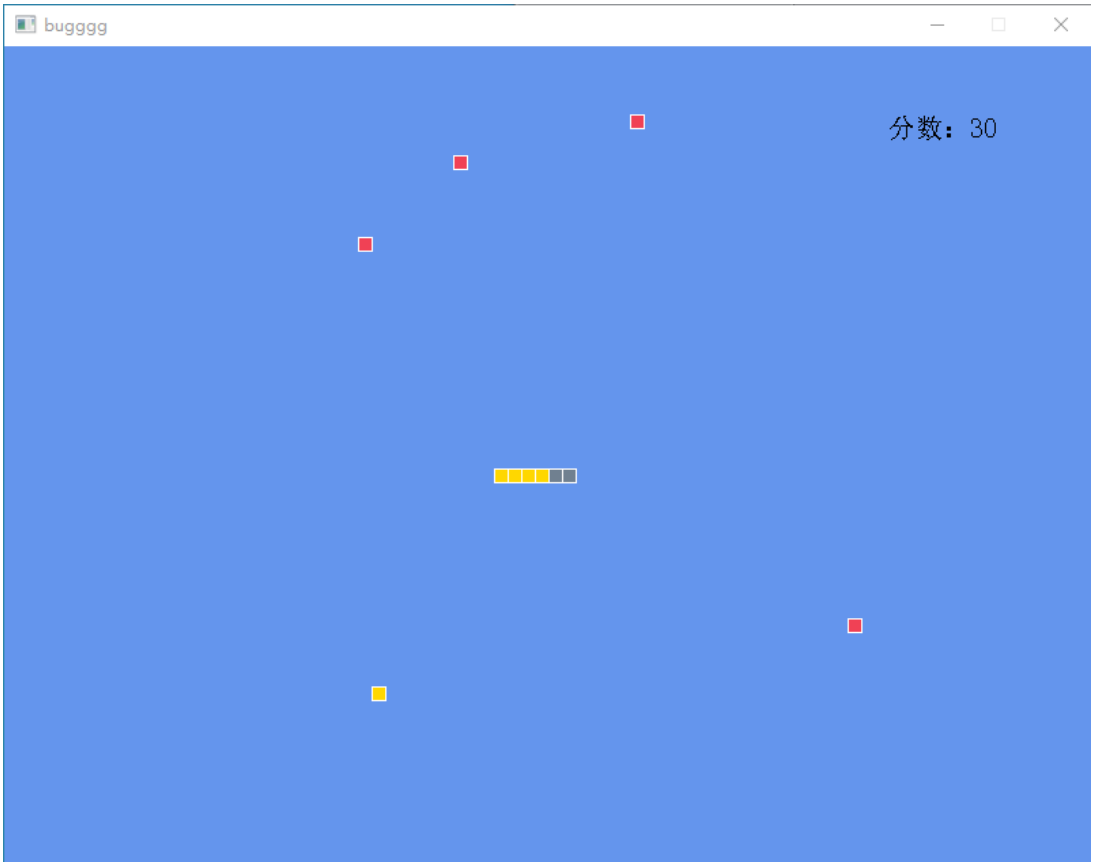
启动程序，进入游戏的欢迎界面：



玩家须知界面：

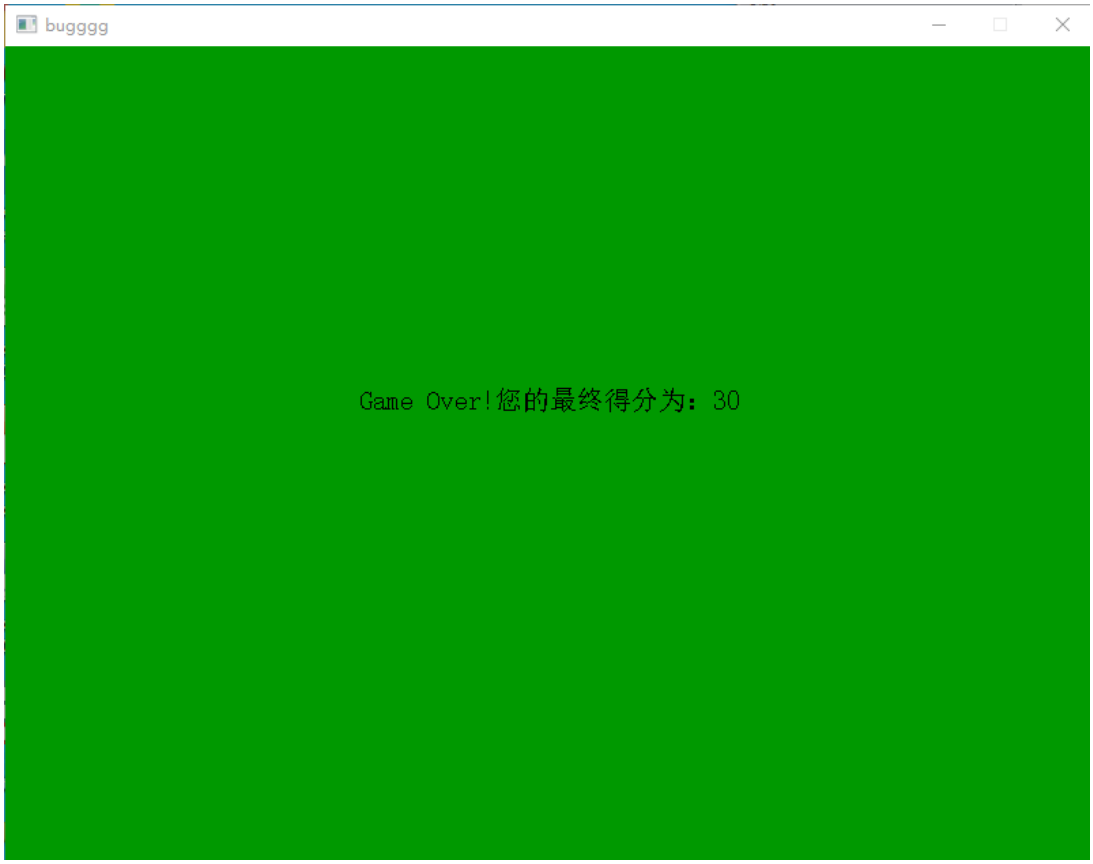


按任意键开始游戏，出现游戏操作界面，灰色蛇头，金色蛇身，红色障碍物，（图示为得分 30 分的情况，此时界面的食物只有一个）：



（图示为得分 110 分的情况，此时界面的食物有 2 个）：

蛇死亡后的游戏结束界面：



7 编程中遇到的问题

7.1 问题 1

对蛇撞到障碍物和自身的判断，解决方案是根据蛇运动的四种方向+坐标范围的方式判断，例如下面的程序：

```
// 蛇头撞自己
    for (int i = 1; i < snake.num; i++)
    {
        if (snake.xy[0].x <= snake.xy[i].x + 10 && snake.xy[0].x >= snake.xy[i].x &&
snake.xy[0].y == snake.xy[i].y&&snake.position == left)
            return true;
        if (snake.xy[0].x + 10 >= snake.xy[i].x && snake.xy[0].x + 10 <= snake.xy[i].x +
10 && snake.xy[0].y == snake.xy[i].y && snake.position == right)
            return true;
        if (snake.xy[0].y <= snake.xy[i].y + 10 && snake.xy[0].y >= snake.xy[i].y &&
snake.xy[0].x == snake.xy[i].x && snake.position == up)
            return true;
        if (snake.xy[0].y + 10 >= snake.xy[i].y && snake.xy[0].y + 10 <= snake.xy[i].y +
10 && snake.xy[0].x == snake.xy[i].x && snake.position == down)
            return true;
    }
```

程序判断语句中的“……&&snake.position == left”、 “……&&snake.position == right”、“……&&snake.position == up”、“……&&snake.position == down”即是对蛇四种运动方向的分解，在对应情况下，判断“碰撞”情况的坐标变得容易书写。

7.2 问题 2

撞到墙的判断（坐标范围问题），因为墙体的坐标不在画布800*600的范围内，因此其判断方式和撞到食物和蛇自身的方式有所区别，此处的解决方式是将墙壁向外扩展一圈，即810*610，这样蛇就能到达地图边缘，图像连接完整。

程序书写如下：

```
if (snake.xy[0].y <= -10 && snake.position == up)            return true;
if (snake.xy[0].y + 10 >= 610 && snake.position == down)     return true;
if (snake.xy[0].x <= -10 && snake.position == left)          return true;
if (snake.xy[0].x + 10 >= 810 && snake.position == right)    return true;
```

7.3 问题 3

mp3 格式的背景音乐的加载方法。一般情况下的音乐要使用 wav 格式，但是查询资料后，得到了加载 mp3 格式背景音乐的办法：

使用 msi 库，首先加入头文件：

```
#include "mmsystem.h"
#pragma comment(lib,"Winmm.lib")
```

执行程序如下：

```
//打开并播放背景音乐的程序
MCI_OPEN_PARMS mciOpen;
```

```
    mciOpen.lpstrDeviceType = _T("mpegvideo");
    mciOpen.lpstrElementName = _T("D://background.mp3");//背景音乐的路径
    MCIERROR mciError = mciSendCommand(0, MCI_OPEN, MCI_OPEN_TYPE | MCI_OPEN_ELEMENT,
(DWORD)&mciOpen);
    if (mciError)
    {
        TCHAR buf[128] = { 0 };
        mciGetErrorString(mciError, buf, 128);
        printf("%s/n", buf);
    }
    UINT DeviceID = mciOpen.wDeviceID;
    MCI_PLAY_PARMS mciPlay;
    mciError = mciSendCommand(DeviceID, MCI_PLAY, 0, (DWORD)&mciPlay);
    if (mciError)
    {
        printf("send MCI_PLAY command failed/n");
    }
```

7.4 问题 4

如何避免随机生成的食物/障碍物与蛇彼此位置出现重合：设置三种物体初始化的顺序，并将后面初始化的物体与前面已经初始化生成的物体的坐标进行判断，即判断语句越往后累加的越长，判断的思路也更清晰。

主函数中各物体的初始化顺序为：蛇——食物——障碍物 1——障碍物 2——障碍物 3——障碍物 4，因此，例如障碍物 2，其需要避免重合的物体为蛇、食物和障碍物 1，其程序语句如下：

```
void initbarrier2()
{
    barrier2.brxy.x = rand() % 70 * 10;
    barrier2.brxy.y = rand() % 50 * 10;
    for (int i = 0; i < snake.num; i++)
        if (barrier2.brxy.x == snake.xy[i].x && barrier2.brxy.y == snake.xy[i].y ||
        barrier2.brxy.x == food.fdxy[i].x && barrier2.brxy.y == food.fdxy[i].y ||
        barrier2.brxy.x == barrier1.brxy.x && barrier2.brxy.y == barrier1.brxy.y)
        // 避免障碍生成在蛇上或者与食物重合或者障碍物彼此重合
        {
            barrier2.brxy.x = rand() % 70 * 10;//重新执行随机生成函数操作
            barrier2.brxy.y = rand() % 50 * 10;
        }
    }
```

7.5 问题 5

如何避免蛇头反向运动的问题。程序设计原理中不允许因蛇反向移动导致游戏结束的情况，即例如蛇头正在向右运动，此时若玩家按下向左的控制按键，则蛇头会因为坐标与相邻的蛇身重合而导致符合“蛇头碰撞蛇身的情况”导致游戏结束，这样的情况属于玩家误触操作，不应该判定为游戏结束，即需要进行此类问题的避免。解决方案是，在“按键交互”函

数中，对每一种移动情况的适用范围添加限制条件，即例如蛇下一步被控制向左走，则蛇的上一刻的状态不能为向右走，否则“移动蛇”的程序不会执行，示例程序如下：

```
void keyDown()
{
    char userKey = _getch();
    if (userKey == -32)
        userKey = -_getch();
    switch (userKey)
    {
    ……
    case 'a':
    case 'A':
    case -75:
        if (snake.position != right)//蛇的上一刻的状态不能为向右走
            snake.position = left; //控制蛇下一步向左走
        break;
    ……
    }
}
```

8 分析总结与心得体会

通过此次课程设计，我巩固了编程方面的知识，对于面向对象思想和 UI 的设计过程有了更深的认识。在程序设计过程中，学习了利用 Visual C++ 2017+EasyX 开发工具，借助 C++语言和图形库的相关语句，开发一款小游戏，在此过程中拓展了知识面，也在解决编程问题的过程中更加熟悉了相关函数语句的功能、适用范围等知识。

此外，贪吃蛇是一款小时候就经常玩的游戏，通过此次课程设计，第一次真正了解了这款游戏的开发算法、游戏原理，这个过程感觉很奇妙。同时，调试程序的过程也相当于是自己体验游戏的过程，这使得调试 bug 的过程没有平日里那么枯燥。

总之，这一周的课程设计收获很多，也让我有了开发游戏的成就感和幸福感。