贪吃蛇实验报告

国豪 06 班 2251079 隋建政

2022.6.22

1. 设计思路与功能描述

1.1 类和对象的设计思路

1.1.1 Snake 类

Snake 中的数据主要分为与绘图相关以及与蛇状态相关的两部分，绘图相关的主要有颜色与类型，状态相关的有位置、长度、速度、是否死亡以及是否暴走等。类中主要的函数为构造函数、绘制蛇的函数、进行蛇每一步移动的函数、判断蛇是否符合死亡条件和是否吃到食物的函数以及将蛇的状态由普通模式转变为暴走模式的函数。

蛇的对象在游戏开始时被初始化，由玩家选定相应的蛇类型并赋予相应的颜色，在游戏过程中，通过对上下左右键的读取，利用 move 函数，对蛇的下一步状态进行判断，然后在屏幕中进行输出。

1.1.2 Fruit 类

Fruit 中的数据与 Snake 类似，分为与绘图相关的颜色以及与状态相关的位置、是否存在以及效果（本来希望做出不同效果的 fruit 但时间条件实在是不允许因此最终只有一种效果的 fruit），类中的主要函数为创造果实的函数、绘制果实的函数以及删除果实的函数。按照游戏规则，每当屏幕中生成的五个果实被吃光就会创造新的五个果实并进行绘制，当果实被蛇吃掉时就会利用删除果实的函数将其清除。

1.1.3 Wall 类

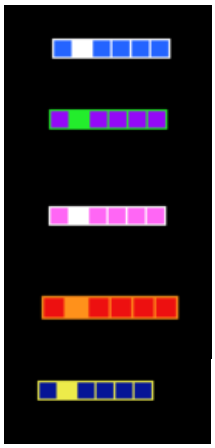
Wall 类只在后两个模式中使用到，Wall 中的数据仅有长度、位置以及是否存在三个，类中的主要函数为构造函数、析构函数以及绘制的函数。其中位置数是在程序运行过程中动态申请的 POINT 型一维数组（POINT 为自定义的描述坐标的结构体），当蛇死亡时便会进行初始化并进行绘制。

1.2 功能实现

1.2.1 游戏流程

首先进入游戏会有菜单栏，可以选择三种游玩模式，或读取历史记录，也可以直接进行退出。

当选择完游玩模式后会进行贪吃蛇皮肤的选择，共有 5 种可选（不同蛇之间除颜色外并无差别，特殊功能的蛇也因为时间问题没有完成）。



而不同的蛇在进入暴走模式时的颜色变化也是不同的，不同蛇暴走模式的演示放在 b 站，视频链接为： https://b23.tv/ZmURp3N。

选择完成后在进入游戏之前还有初始图片，是仿照 EVA 动画中的风格制作的，游戏结束时同样有类似的图片，按回车可以继续当前游戏模式，而按 q 则退出回主界面。

1.2.2 普通版

普通版便是最基础的功能，每次随机生成五个果实，当蛇全部吃完时会继续生成新一批的果实，直到蛇撞墙或者装在自己的身体上游戏就会结束。

1.2.3 进阶版

进阶版在普通版的基础上添加了蛇死后变为墙的功能（蛇死之后变为墙的颜色为橙色是致敬 EVA 中的橙汁 LCL 之海），同时蛇拥有五次生命值，其余均与普通版一致，区别是在进阶版中蛇每次生成的位置是随机的，而在普通版中每次的生成位置则是固定的。

1.2.4 高级版

高级版与进阶版类似，也是在普通版的基础上进行改动，添加了蛇死后身体变为果实的功能，同样拥有五次生命，而每次蛇的生成位置均为随机。

1.2.4 UI

ui 实时记录游玩时间、得分、记录（当当前得分超过记录时则会随当前记录不断刷新）、蛇的长度以及游戏速度，在后两个模式中还有蛇的剩余生命的显示，在最下方为蛇的状态，正常是现实绿色的正常，而当蛇进入暴走模式时则更改为红色的暴走。

1.2.5 游戏记录

首先各个模式的最高分会在游玩过程中的 ui 实时展示，当在主界面选择历史记录选项后，就可以看到迄今为止所有的游玩历史，记录了游戏模式、用户名最终得分以及游戏中所选择的贪吃蛇类型。

2. 遇到的问题及解决方案

2.1 贪吃蛇不同颜色皮肤呈现的问题

贪吃蛇的皮肤在游戏开始前由玩家选定，我的想法是将玩家选择的 type 作为蛇初始化函数的参数，因此蛇的颜色可以根据 type 进行初始化，而在之后的暴走模式中，也是根据 type 的不同，实现暴走模式副色变化不相同的效果。我的贪吃蛇并非每一节都是相同的，而是分为主色和副色，其中主色就是主体的颜色，而肤色是蛇身体轮廓以及第二节身体的颜色。将第二节身体的颜色做区分，一是为了美观，二是为了方便玩家区分贪吃蛇的头部与尾部，当游戏刚刚开始时，玩家只能延头朝向以及上下三个方向运动，而不能向尾部的方向运动，这样的颜色设计可以起到提示的作用，我认为算是一种良好的人机交互。

2.2 暴走模式的实现

暴走模式是本游戏的特殊机制，即每吃掉十个果实就会进入一次暴走模式，蛇的颜色发生改变，并且得分会相应的增加。但在暴走模式进入与结束的问题上经常出现问题。

首先进入暴走模式前要满足两个条件，累计吃掉的果实达到十个，且目前不处于暴走模式中，如果缺少该条件的判断，后续随着食物的增多，暴走模式的进入判断就会产生问题。且每次进入暴走模式时都将先前统计的果实数量清零。当暴走模式结束时，颜色变回原先的配色，并将 Snake 类中的判断变量更改为不暴走的状态，此时便回到了普通模式。

2.3 蛇卡住的问题

在刚开始调试普通版游戏时，我遇到了在蛇死后不断重复进行游戏时，蛇会卡死的问题，经过反复试验，我推测出应该是当上一局蛇死亡的情境下，如果情急之中按了键盘但蛇却没有读取，就会造成下一局游戏开始时蛇卡死的情况。因此在游戏结束后加入结束画面，并要求玩家按下回车才能继续下一把游戏后，这个问题就迎刃而解了。

2.4 说明

为使用 outtextxy 输出文字，我将默认的字符集修改为了多字节字符集，如果在运行过程中出现问题，烦请老师或助教哥哥姐姐们自行修改一下设置。

3. 心得体会

完成此次项目，我想最大的心得便是关于 oop 思维的应用，也许仅仅开发一个模式时感受不到 oop 的优势所在，但当模式和玩法不断延伸，oop 的优势便也有了明显的体现，当各个大类抽象完成时，尽管抽象时确实耗时很多，但在后续使用中确实有着十足

的简化。oop 的应用确实简化了面向过程设计中冗杂的流程设计。我想在面向对象编程时，初期的工程量是较大的，但在后期多种不同模式的开发和功能的延伸中，面向对象编程会使得程序更加简便，可读性更好。

但此次项目仍有些未完成的遗憾，如更多项目的开发，玩家操控 EVA 与 ai 的使徒进行对战，RPG 的故事模式等等，碍于时间的因素无法完成，其次就是 ui 的设计还可以更加美观，但精力有限还是没有完成，我想在之后的开发可以把这部分也利用 oop 进行使用，减少程序中段大量冗杂的代码，我想都是开发初期时间精力投入不足的问题，以后都会一一改进。

源代码

```
struct Point {
    int x;
    int y;
};

class Snake;
class Game {
public:
    int score;
    int life;
    Game();
};

Game::Game()
{
    score = 0;
    life = 5;
}

class Wall {
private:
public:
    int num;
    POINT* pos;
    bool ifexist;
    Wall() { num = 0; pos = NULL; ifexist = false; }
    ~Wall() { delete pos; }
    void create(Snake snake);
    void draw();
    void draw_senior();
    void Delete(int);
};
```

```
class Fruit {
private:
    Point pos;
    COLORREF linecolor;
    COLORREF fillcolor;
public:
    char effect;
    bool exist;
    Fruit();
    void create(Fruit*, Snake, int);
    void create(Fruit*, Snake, int, Wall*);
    void draw();
    int value_x();
    int value_y();
    void Delete();
};

class Snake {
private:
    Point headPos;
    Point tailPos;
    Point body[500];
    COLORREF color1;
    COLORREF color2;
    int len;
public:
    bool ifGod;
    int type;
    char headToward;
    bool death;
    int speed;
    Snake(int);
    Snake(int, Wall *);
    void initDraw();
    void draw();
    void check_death(Wall*);
    bool check_fruit(Fruit);
    bool check_fruit_senior(Wall*);
    void move(Fruit*, int, Game&);
    void move(Fruit*, int, Game&, Wall*);
    void move_senior(Fruit*, int, Game&, Wall*);
    int value_len();
    void turn_to_God();
```

```
    void turn_to_normal();
    Point value_body(int i);
};
void readKeyboard(Snake& snake, int& ifBegin)
{
    char pre_Toward = snake.headToward;
    if (_kbhit())
    {
        fflush(stdin);
        int ch;
        ch = _getch();
        ch = _getch();
        switch (ch) {
            case 72:
                snake.headToward = 'U';
                break;
            case 80:
                snake.headToward = 'D';
                break;
            case 75:
                snake.headToward = 'L';
                break;
            case 77:
                snake.headToward = 'R';
                break;
            default:
                break;
        }
    }

    if (pre_Toward == 'U' && snake.headToward == 'D')
        snake.headToward = 'U';
    if (pre_Toward == 'D' && snake.headToward == 'U')
        snake.headToward = 'D';
    if (pre_Toward == 'L' && snake.headToward == 'R')
        snake.headToward = 'L';
    if (pre_Toward == 'R' && snake.headToward == 'L')
        snake.headToward = 'R';
    if (pre_Toward == '0' && snake.headToward == 'R')
        snake.headToward = '0';
    if (snake.headToward != '0')
        ifBegin = 1;
}
void readMouse(int& mode)
```

```
{
    MOUSEMSG m{ 0 };
    bool flag = true;
    while (flag)
    {
        if (MouseHit())m = GetMouseMsg();
        if (m.uMsg == WM_LBUTTONDOWN && m.x >= 20 && m.x <= 260 && m.y >= 140 && m.y
<= 210) {
            mode = 1;
            flag = false;
        }
        if (m.uMsg == WM_LBUTTONDOWN && m.x >= 20 && m.x <= 260 && m.y >= 230 && m.y
<= 300) {
            mode = 2;
            flag = false;
        }
        if (m.uMsg == WM_LBUTTONDOWN && m.x >= 20 && m.x <= 260 && m.y >= 320 && m.y
<= 390) {
            mode = 3;
            flag = false;
        }
        if (m.uMsg == WM_LBUTTONDOWN && m.x >= 20 && m.x <= 340 && m.y >= 410 && m.y
<= 480) {
            mode = 4;
            flag = false;
        }
        if (m.uMsg == WM_LBUTTONDOWN && m.x >= 20 && m.x <= 180 && m.y >= 500 && m.y
<= 590) {
            mode = 5;
            flag = false;
        }
    }
}

int menu()
{
    int mode;
    initgraph(950, 650);
    IMAGE img_menu(950, 650);
    loadimage(&img_menu, _T("menu.jpg"));
    putimage(0, 0, &img_menu);
    settextstyle(70, 40, _T("黑体"));
    outtextxy(20, 140, "入门版");
    outtextxy(20, 230, "进阶版");
```

```
    outtextxy(20, 320, "高级版");
    outtextxy(20, 410, "历史记录");
    outtextxy(20, 500, "退出");
    setbkmode(TRANSPARENT);
    settextstyle(80, 60, _T("黑体"));
    settextcolor(WHITE);
    outtextxy(20, 20, "EVA SNAKE");
    readMouse(mode);
    switch (mode) {
        case 1:
            settextcolor(MY_LIGHT_BLUE);
            settextstyle(70, 40, _T("黑体"));
            outtextxy(20, 140, "入门版");
            break;
        case 2:
            settextcolor(MY_LIGHT_BLUE);
            settextstyle(70, 40, _T("黑体"));
            outtextxy(20, 230, "进阶版");
            break;
        case 3:
            settextcolor(MY_LIGHT_BLUE);
            settextstyle(70, 40, _T("黑体"));
            outtextxy(20, 320, "高级版");
            break;
        case 4:
            settextcolor(MY_LIGHT_BLUE);
            settextstyle(70, 40, _T("黑体"));
            outtextxy(20, 410, "历史记录");
            break;
        case 5:
            settextcolor(MY_LIGHT_BLUE);
            settextstyle(70, 40, _T("黑体"));
            outtextxy(20, 500, "退出");
            break;
        default:
            break;
    }
    Sleep(200);
    closegraph();
    return mode;
}
void readHistory()
{
    system("cls");
```

```
    char ch;
    file.open("history.txt", ios::in);
    while (!file.eof()) {
        ch = file.get();
        putchar(ch);
    }
    cout << "\n按enter退出" << endl;
    while (true) {
        ch = _getch();
        if (ch == '\r')
            break;
    }
}

int main()
{
    srand((unsigned int)(time(0)));
    bool flag = true;

    while (flag) {
        int mode;
        mode = menu();
        switch (mode)
        {
            case 1:
                starter_init();
                break;
            case 2:
                advanced_init();
                break;
            case 3:
                senior_init();
                break;
            case 4:
                readHistory();
                break;
            case 5:
                flag = false;
                break;
            default:
                break;
        }
    }
    system("cls");
```

```
    return 0;
}
```