ZMC

班级：12  
学号：2151765  
姓名：张铭宸  
完成日期：2022.6.1

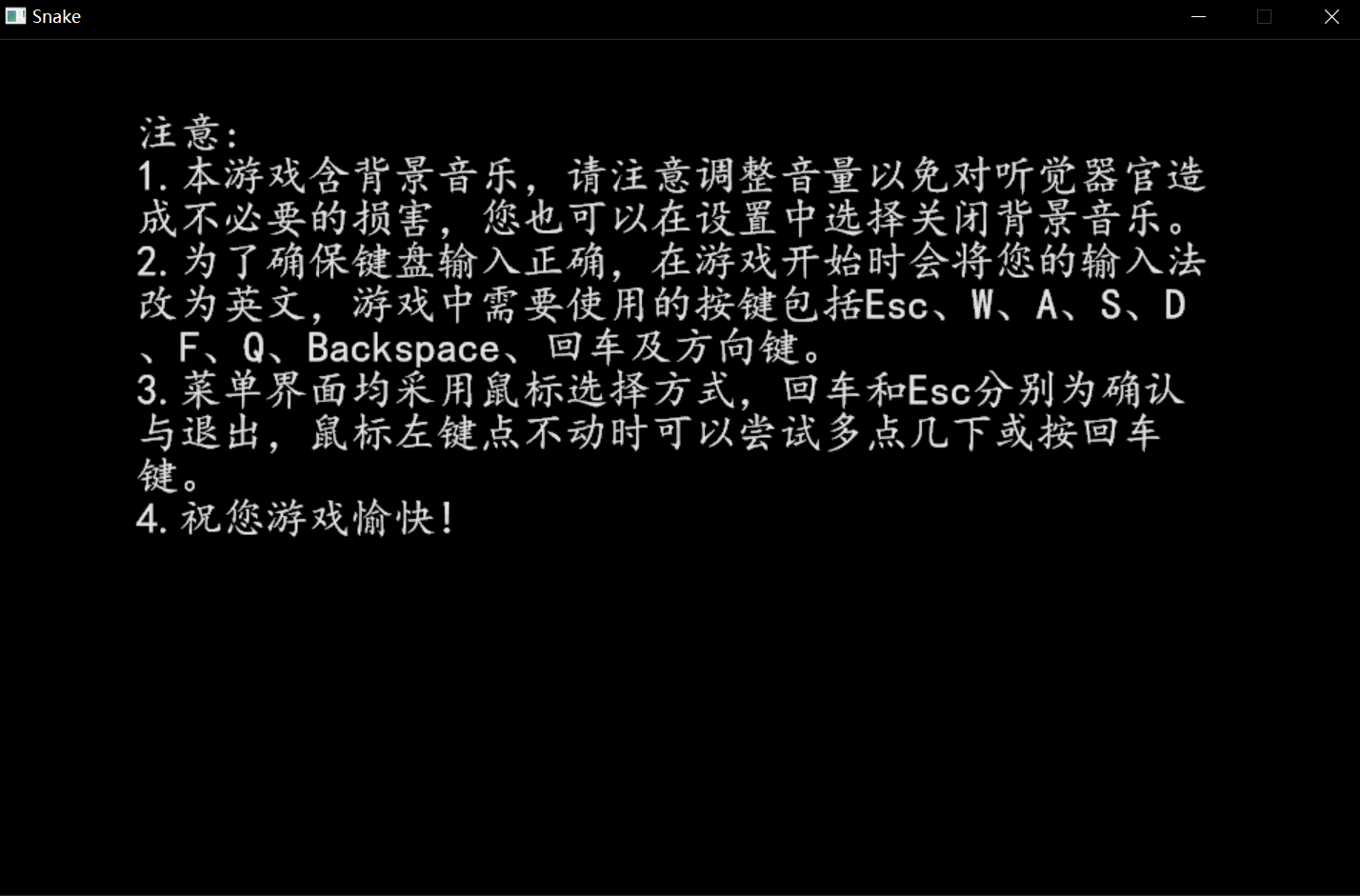
贪吃蛇

大作业实验报告

1. 功能描述

1-0.VS中运行Debug x86模式

1-1.启动程序时显示提示界面并逐渐淡出，随后显示标题界面，按任意键继续。





1-2.主界面6个选项：标准模式、故事模式、双人对决、设置、本地记录、退出，鼠标选择，左键或回车确认，Esc退出。



1-3.标准模式子菜单，选择三种版本，分别为入门版(原始规则)、进阶版（蛇死后变为墙壁）、高级版（蛇死后变为食物）。



1-4.标准模式游玩，右侧UI显示关卡名称、生命数、血量、分数（显示最近一次加分）、用时、当前模式最高分数、道具介绍。生命值归零时游戏结束，并记录本次游戏信息到本地文件。



1-5.游戏结束界面，可以选择返回主菜单、重开一把或直接退出程序。



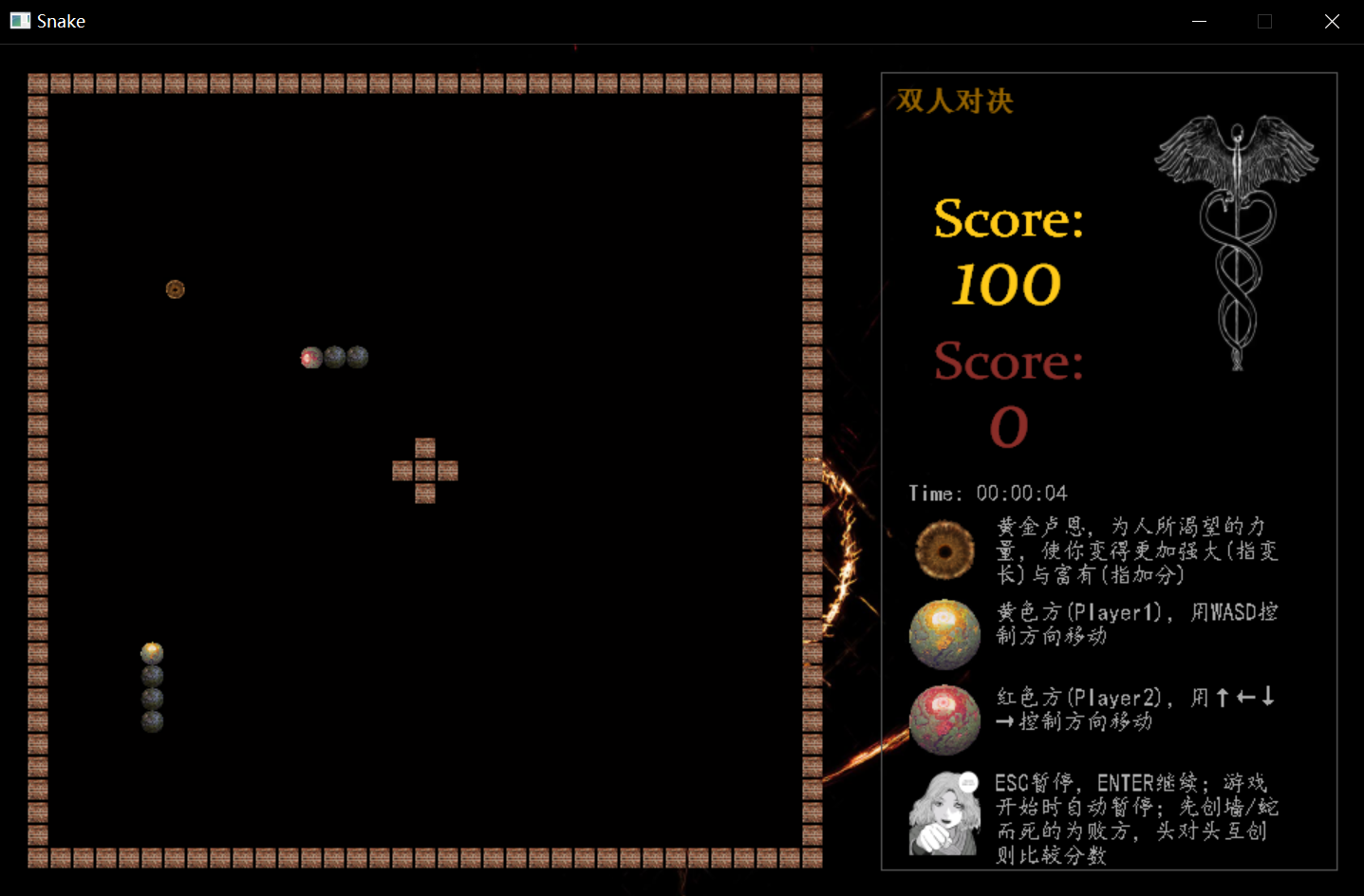
1-6.故事模式（RPG）子菜单，可选择5关，每关关前有小剧情，有不同通关目标（第五关为boss战，boss掌握攻击手段），每关通关后可选择返回菜单、重玩本关或进入下一关。



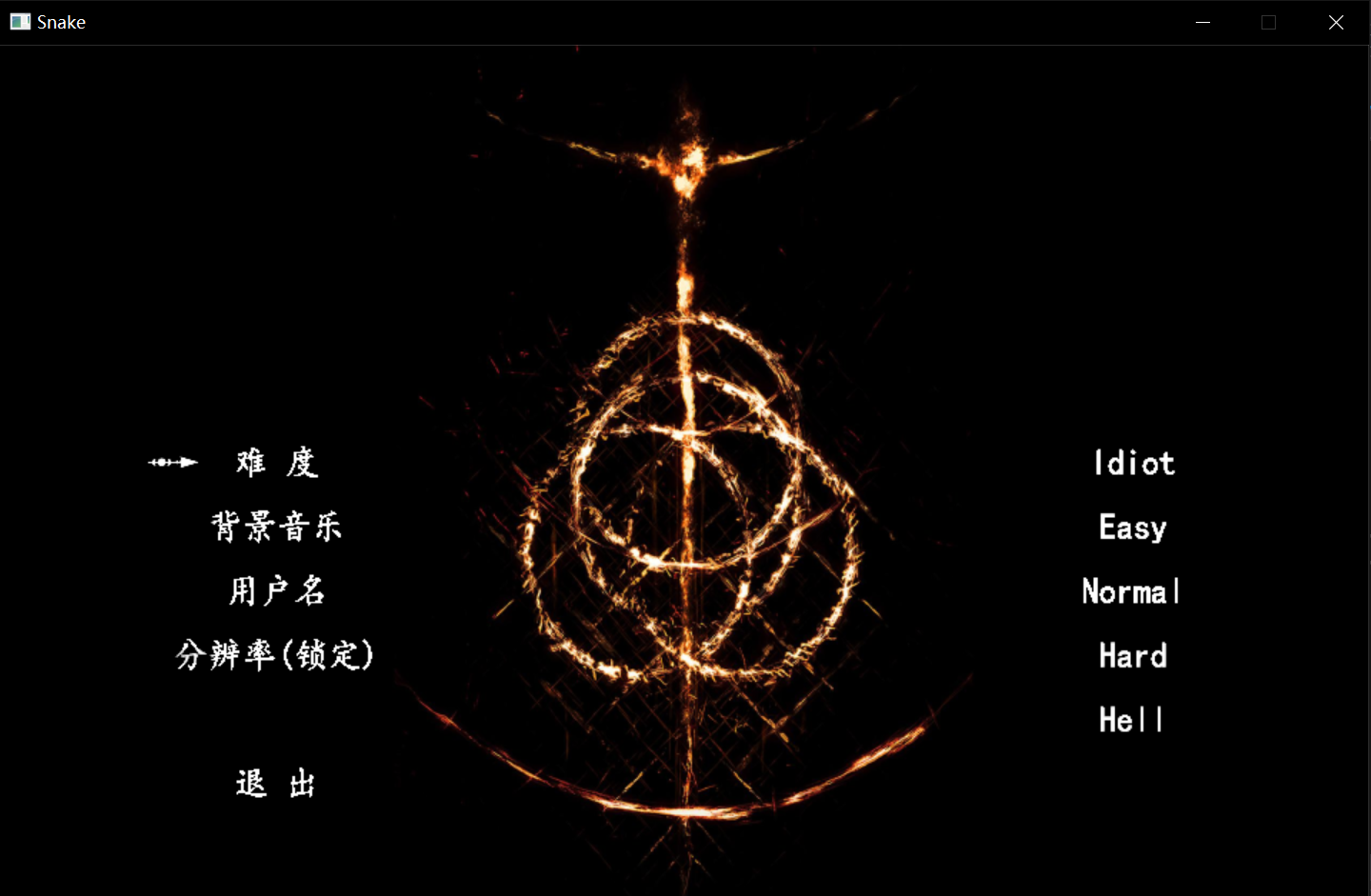




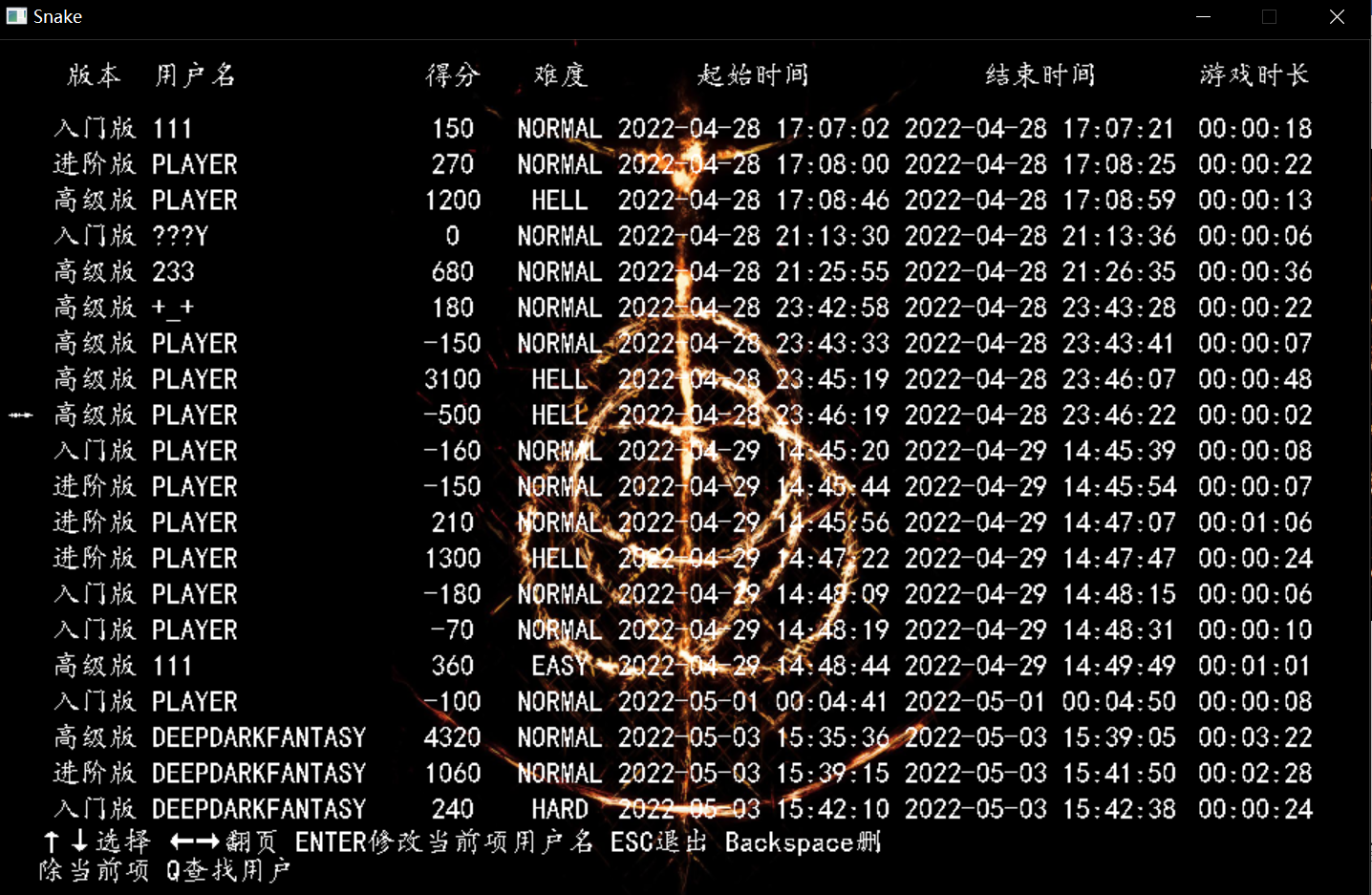
1-7.双人模式界面。一方使用WASD控制，另一方使用方向键控制，头对头相撞时比较得分。

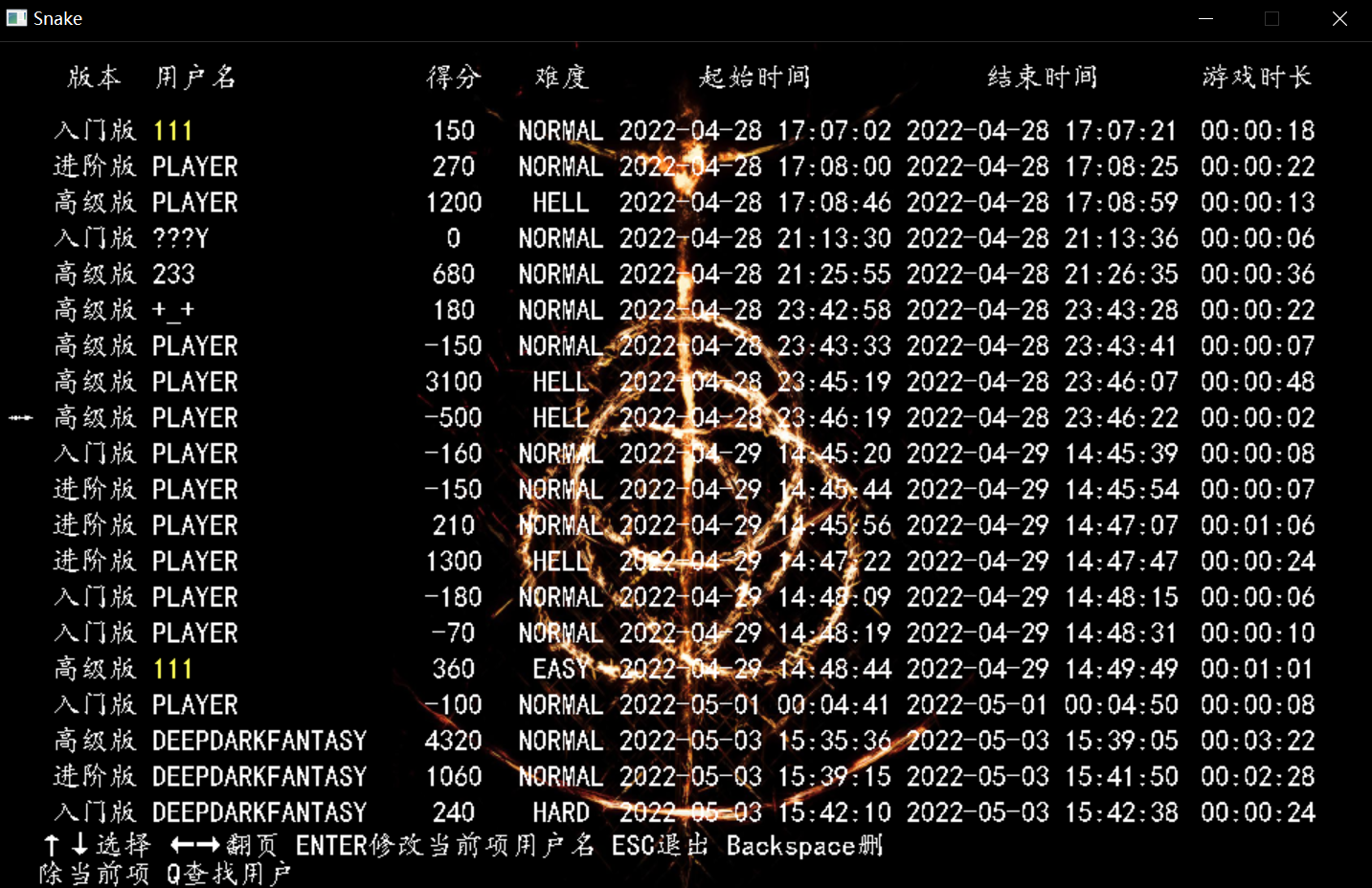


1-8.设置界面，可以更改标注模式游戏难度、用户名，以及设置BGM开关。



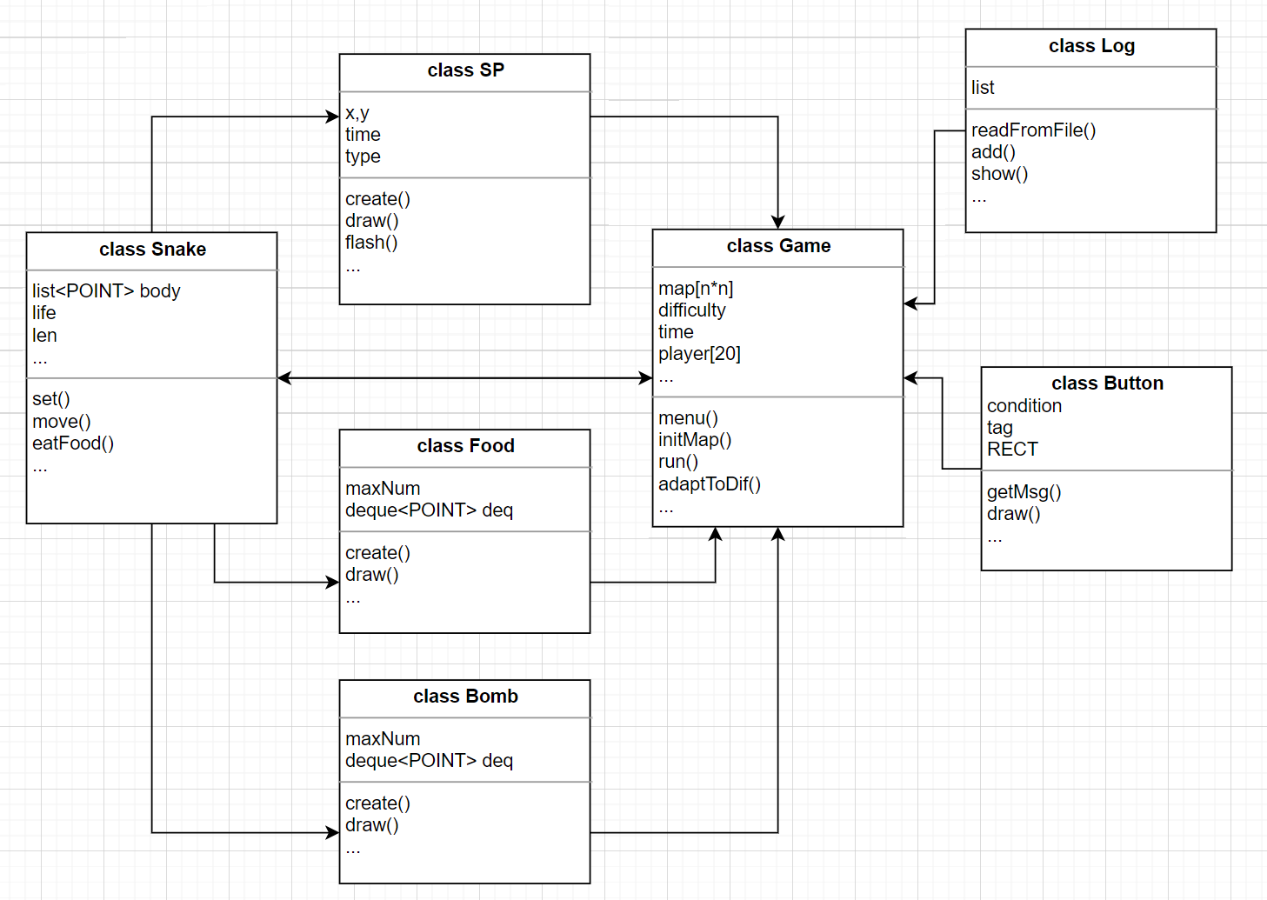
1-9.本地记录界面。可以更改指定用户名或删除指定记录，或按用户名查找记录。





1. 设计思路

2-1.使用面向对象思想，对于蛇(Snake)、食物(Food)、特殊食物(SP)、有害道具(Bomb)、按钮(Button)、本地记录(Log)各建立一个class，再建立一个class Game，存放游戏地图、游戏设置、游戏流程等信息。



2-2.对于class Snake，由于只需要重点考虑蛇头与蛇尾，所以用一个list来存放位置信息，front()即为蛇头，back()即为蛇尾，另外还有蛇的前进方向、长度、血量、生命值、蛇头和蛇身的图片（部分情况可能有变化）。蛇的速度由于以Sleep()的方式体现，且受到难度设置影响，所以放在class Game中。蛇的前进过程move()即为通过蛇头位置与方向判断下一个点->判断下一个点是否撞墙/撞自己->取出末尾项，将下一个点放在链表首位，对按键的响应(包括暂停)放在函数changeDir()中，该函数为一个条件为\_kbhit()的循环。

2-3.class Game中包含了游戏开始界面、菜单和子菜单、游戏流程等多个函数，如标准模式函数int StandardGame(int mode)。在这一过程中，先为Snake、Food、Log等申请动态内存，之后显示UI，设置计时起点，然后进入循环。循环以蛇死亡，即生命数归零为结束条件，每次循环中蛇进行一次移动，并进行一次按键的响应，同时进行是否掉血/是否死亡/是否吃到道具的判断（由于食物和掉血道具的位置信息还储存在各自的链表中，所以可以先移动再进行判断），并更新游戏时间，用Sleep()实现动画效果和特殊道具的闪烁显示效果。循环结束时将本次游戏的数据存入Log中

2-4.class SP表示限时的特殊道具，其中包含一个time\_t数据，通过游戏进行时与计时功能相作用实现限时效果。class Bomb和Food各自以一个链表结构来存储每一个道具的位置信息，并在生成与消耗时与Game中的信息进行同步

2-5.class Log用链表结构读取文件1.log中记录的历史信息，也可读取2.log中的最高分信息，在一次标准游戏结束后同时更新两者。

2-6.class Button主要用于实现菜单和设置里的按钮功能，能根据鼠标位置和是否左键点击作不同显示。

1. 问题及解决方案

3-1.当蛇转弯时恰好在墙壁（或食物）与食物中间，吃下前方的食物后身体变长会占到身后的墙壁/食物一格。对于这种情况，可以在蛇变长的函数中对身后格子进行判断，若为空则正常变长，若不为空则向两侧寻找位置。

3-2.当蛇恰好呈一个环形，即头前一格为尾巴时，按一般写法（即判断前方一格是否为蛇身体），但其实不应算作撞到自己，在判断是否自撞的函数中获取蛇尾坐标进行判断即可。

3-3.RPG模式第五关中boss释放攻击的招式偷懒直接用Snake类实现，导致了一些问题，如：攻击路线上有道具时会把该位置的道具在Game中的map上消掉，但其自身类中仍有位置信息，导致不再刷新新的道具，直到蛇误打误撞碰到道具类中存储的位置。解决方案：1、攻击结束后将类中信息同步至Game中并显示；2、攻击与道具相遇时套用蛇吃食物等的函数，重新刷新道具位置。

1. 心得体会

4-1.先建立类图，大致分清各个类的功能与联系，避免被类之间的关系绕晕，减少后续修改工作量。

4-2.画UI、按钮等小功能实现起来比想象中要麻烦的多。

4-3.利用类的继承可以更好地减少代码复用，如Bomb和Food、SP，Snake和RPG模式中的敌方攻击等，但开始写的时候还没学，后面也懒得改了……

1. 源代码

class Game

{

private:

int\* map;

int dif;

int score;

double score\_power; //分数倍率

int lose\_hp;

int pause;

time\_t start\_time;

time\_t last\_time;

char player[20];

public:

Game(int);

~Game();

void clear();

void initMap(int(\*nmap)[3] = NULL);

void clearMap();

void adaptToDif(Snake& snake);

void preTips();

int start();

int menu();

int \_menu();

void menu\_show\_detail(LPCTSTR str, RECT\* r);

void settings();

void setting\_Dif();

void setting\_BGM();

void setting\_Name();

void localLog();

int run();

int standardGame(int mode);

int gameOver(bool is\_RPG = false);

int doublePlayer();

bool changeDir\_double(Snake& snake1, Snake& snake2);

int gameOver\_double(int winnner);

void showScore\_double(int score, int player);

void showGameUI\_double();

void showItemRules\_double();

bool is\_headToHead(Snake& snake1, Snake& snake2);

int \_menu\_RPG();

int RPG(int level);

int RPG\_G1();

int RPG\_G2();

int RPG\_G3();

int RPG\_G4();

int RPG\_G5();

void showMsg(LPCTSTR str);

void showDia(int speaker, LPCTSTR content, int yDia);

void showGameUI\_RPG(int level);

void setGuard(int x, int y, int dir);

void drawHP\_RPG(double hp);

void updateHP\_RPG(int before, int after);

void alarm(Snake& snake, Food& food, SP& sp, Bomb& bomb);

bool is\_mapFull();

void updateScore(int type, bool clear = false);

void showScore(int add);

void showGameUI(int mode);

void showTime();

void showItemRules();

void showModeTitle(LPCTSTR title);

void inputBox(RECT& r, char\* str, int maxlen);

void showPointWall(int x, int y);

void showPointVoid(int x, int y);

void showPointWeak(int x, int y);

void showPointLight(int x, int y);

bool is\_pointEmpty(int x, int y) const;

void setPoint(int x, int y, int type);

void clearPoint(int x, int y);

friend class Snake;

friend class SP;

};

int Game::standardGame(int mode)

{

if (BGM\_ON) {

mciSendString(\_T("open bin/music/bk\_nor.mp3 alias BGM"), NULL, 0, NULL);

mciSendString(\_T("play BGM repeat"), NULL, 0, NULL);

}

Snake\* snake = new(std::nothrow) Snake();

if (snake == NULL)

return -1;

Food\* food = new(std::nothrow) Food();

if (food == NULL)

return -1;

Bomb\* bomb = new(std::nothrow) Bomb();

if (bomb == NULL)

return -1;

SP\* sp = new(std::nothrow) SP();

if (sp == NULL)

return -1;

Log\* log = new(std::nothrow) Log("bin/log/1.log", "bin/log/2.log");

if (log == NULL)

return -1;

Item m\_i(mode, this->dif, this->player);

initMap();

adaptToDif(\*snake);

showGameUI(mode);

snake->setInMap(\*this);

snake->drawHP();

snake->drawLife();

int loop\_times = 0;

getchar();

last\_time = 0;

time(&start\_time);

m\_i.begin\_time = this->start\_time; //记录起始时间至文件

log->showMaxScore(mode); //显示最高分

//先生成一次

if (this->dif != DIF\_IDIOT)

bomb->create(\*this);

food->create(\*this);

sp->create(\*this, 15);

while (!snake->is\_gameOver() && (!is\_mapFull())) {

if (!snake->changeDir()) {

//ESC暂停时

showTime(); //刷新计时

getchar();

time(&start\_time); //重置计时起点

}

if (!snake->move(\*this)) {

snake->loseLife();

updateScore(0);

if (snake->is\_gameOver())

break;

if (mode == 1) {

snake->turnToVoid(\*this);

snake->reborn(\*this);

}

else if(mode==2)

snake->turnToWall(\*this);

else if(mode==3)

snake->turnToFood(\*this, \*food);

showTime(); //刷新计时

getchar();

snake->updateHP(0, 16);

time(&start\_time); //重置计时起点

}

if (snake->eatFood(\*food)) {

snake->longer(\*this);

updateScore(1);

}

if (snake->eatSP(\*sp)) {

snake->longer(\*this);

if (dif < DIF\_HELL) {

if (snake->life < 5)

snake->updateLife(1);

}

updateScore(2);

}

if (snake->hitBomb(\*bomb)) {

snake->loseHP(lose\_hp);

updateScore(-1);

if (snake->is\_dead()) {

snake->loseLife();

if (snake->is\_gameOver())

break;

if (mode == 1)

snake->turnToVoid(\*this);

else if (mode == 2)

snake->turnToWall(\*this);

else if (mode == 3)

snake->turnToFood(\*this, \*food);

showTime(); //刷新计时

getchar();

snake->updateHP(0, 16);

time(&start\_time); //重置计时起点

}

}

//生成食物

if (loop\_times == 20) {

food->create(\*this);

if (this->dif != DIF\_IDIOT)

bomb->create(\*this);

loop\_times = 0;

}

showTime();

if (loop\_times % 5) //闪烁效果实现

sp->updateTime(\*this, 30);

Sleep(pause);

loop\_times++;

}

m\_i.score = this->score;

time(&start\_time);

m\_i.end\_time = this->start\_time;

m\_i.last\_time = this->last\_time;

log->add("bin/log/1.log", m\_i);

log->writeMaxScore("bin/log/2.log");

delete snake;

delete food;

delete bomb;

delete sp;

delete log;

if (BGM\_ON) {

mciSendString(\_T("stop BGM"), NULL, 0, NULL);

mciSendString(\_T("close BGM"), NULL, 0, NULL);

}

int temp = this->gameOver();

clearMap();

return temp;

}

class Snake

{

private:

int len;

int HP;

int dir;

int life;

std::list<POINT>body;

IMAGE IM\_HEAD;

IMAGE IM\_SNAKE;

public:

Snake(POINT p, int dire);

Snake(int l = 3, int hp = 16, int lf = 3, int x = 17, int y = 17, int color = 0);

~Snake();

void clear();

void set(int l = 3, int hp = 16, int lf = 3, int x = 17, int y = 17);

void setInMap(Game& game);

int move(Game& game);

void longer(Game& game);

bool eatFood(Food& food);

bool eatSP(SP& sp);

bool hitBomb(Bomb& bomb);

bool hitWall(Game& game, int x, int y);

bool hitSelf(Game& game, int x, int y);

void revDir();

int changeDir();

bool is\_dead();

bool is\_gameOver();

void halfCut(Game& game);

void loseLife();

void loseHP(int delta);

void turnToWall(Game& game);

void turnToFood(Game& game, Food& food);

void turnToVoid(Game& game);

void reborn(Game& game);

bool is\_pointEmpty(Game& game, int x, int y);

void clearPoint(Game& game, int x, int y);

void setPoint(Game& game, int x, int y);

void changeHead(int type);

int getHeadX() const;

int getHeadY() const;

int getTailX() const;

int getTailY() const;

void draw(int x, int y, bool is\_head = 0, bool clear = 0);

void drawSnake(bool clear = 0);

void drawHP(double hp);

void drawHP();

void updateHP(int before, int after);

void drawLife();

void updateLife(int add);

friend class Game;

};

int Snake::move(Game& game)

{

if (body.empty())

return 0;

POINT p = body.front(), p0 = body.front();

switch (dir) {

case DIR\_UP:

p.y--;

break;

case DIR\_DOWN:

p.y++;

break;

case DIR\_LEFT:

p.x--;

break;

case DIR\_RIGHT:

p.x++;

break;

}

if (hitWall(game, p.x, p.y)||hitSelf(game, p.x, p.y)) {

return 0;

}

//画蛇头

body.push\_front(p);

setPoint(game, p.x, p.y);

draw(p.x, p.y, 1);

draw(p0.x, p0.y); //原蛇头变为蛇身

//清除蛇尾

p = body.back();

body.pop\_back();

clearPoint(game, p.x, p.y);

draw(p.x, p.y, 1, 1);

return 1;

}

class Food

{

private:

std::deque<POINT> deq;

int maxNum;

IMAGE IM\_FOOD;

public:

Food(int max = 5);

~Food();

void clear();

void create(Game& game);

void draw(int x, int y, bool clear = false);

void draw();

int getNum() const;

friend class Snake;

};

class Log

{

private:

list<Item> item;

int max[3];

public:

Log(const char\* file, const char\* file\_max);

~Log();

int clearAll(const char\* file, const char\* file\_max);

int readFromFile(const char\* file);

int writeToFile(const char\* file);

int addToFile(const char\* file, Item& \_item);

int add(const char\* file, Item& \_item);

void sortByScore(bool is\_up = false);

void sortByTime(bool is\_up = false);

void sortByMode(bool is\_up = true);

int readMaxScore(const char\* file);

int writeMaxScore(const char\* file);

void showMaxScore(int mode);

void show();

void showHeadTips(const int x[], const int y0 = 50, const int ye = 550);

void showFocus(int focus, bool clear = false);

void showData(int page, const int each\_page = 20, char\* str = NULL);

void inputBox(RECT& r, char\* str, int maxlen);

int getPageLine(int page);

int getMaxScore(int mode);

int sum();

friend class Game;

};