实验报告：类与对象

樊子谦

软件学院 软件工程 2024级1班

1. 引子

本实验工程量较大，总计接近500行，其中除了实验要求之外，做了各种对用户输入数据的正确性判定、清理屏幕保证美观等额外工作，并引入了mt19937、匿名函数等高版本C++的内容，致力为大一下学期的实训做铺垫。

1. 实验过程

**阅读实验要求**

选做任务和必做任务之间是存在一定矛盾的，因为必做任务指定在main函数里面创建精灵等，而选做任务中又要进行一些类似“游戏”的操作，这要求对各个对象的管理需要比较严格。但是处于同一作用域（main函数）的情况下，必做任务占用的内存不会自动销毁，对象一直存在，而delete只能删除动态定义的内存，管理起来很麻烦。于是将这两个任务分开来做。后面发现main函数里写太多东西太难看了，一了百了写成了函数，毕竟在main函数里面执行函数也算是在main函数里面做到了。（逃）

**前期工作**

**随机数实现**

这个实验任务必定是要用到随机数的。我们利用mt19937生成随机数。mt19937具有比传统rand()更加优秀的性能，生成的随机数序列质量也更高。为了方便，我们直接把获取一个范围内的随机自然数写成以下函数（其中lower和upper分别为随机数上下限）：

int get\_rand\_num(int lower, int upper) {

mt19937 rd(random\_device{}());

uniform\_int\_distribution<int> dist(lower, upper);

return dist(rd);

}

**范围判定**

看了一眼题面，存在很多判定范围的需求，我们单独写一个函数。第一个参数保证了可以同时判断很多元素是否在范围内（传入类型为vector），第二个参数就是范围的上下界。

bool in\_range(const auto &r, const pair<int, int> &range) {

for (auto i : r)

if (!(i >= range.first && i <= range.second)) return false;

return true;

}

**主函数与输入异常处理**

上面提到了要把必做任务和选做任务分开实现，那么必定是在main函数通过用户输入分开操作的。但作为程序员，需要知道有输入的地方就意味着必定会有“顾客来麦当劳点了一份隆江猪脚饭”的事情发生，所以我们必须得特判。

用test\_mode代指必做任务（测试模式），extra\_mode代表选做任务，然后就有如下处理，这里在main函数尾部清屏并且反复调用自己达成一个可以反复执行任务直到用户主动退出的效果：

int main() {

cout << "Hello! Please choose the mode:\n";

cout << "1. Test Mode\n";

cout << "2. Extra Mode\n";

cout << "3. Exit\n";

int mode = INT\_MIN;

while (1) {

cin >> mode;

if (mode == 1)

test\_mode();

else if (mode == 2)

extra\_mode();

else if (mode == 3)

return 0;

else {

// cout << mode;

// output: 0

cout << "Invalid input.\n";

continue;

}

break;

}

system("pause");

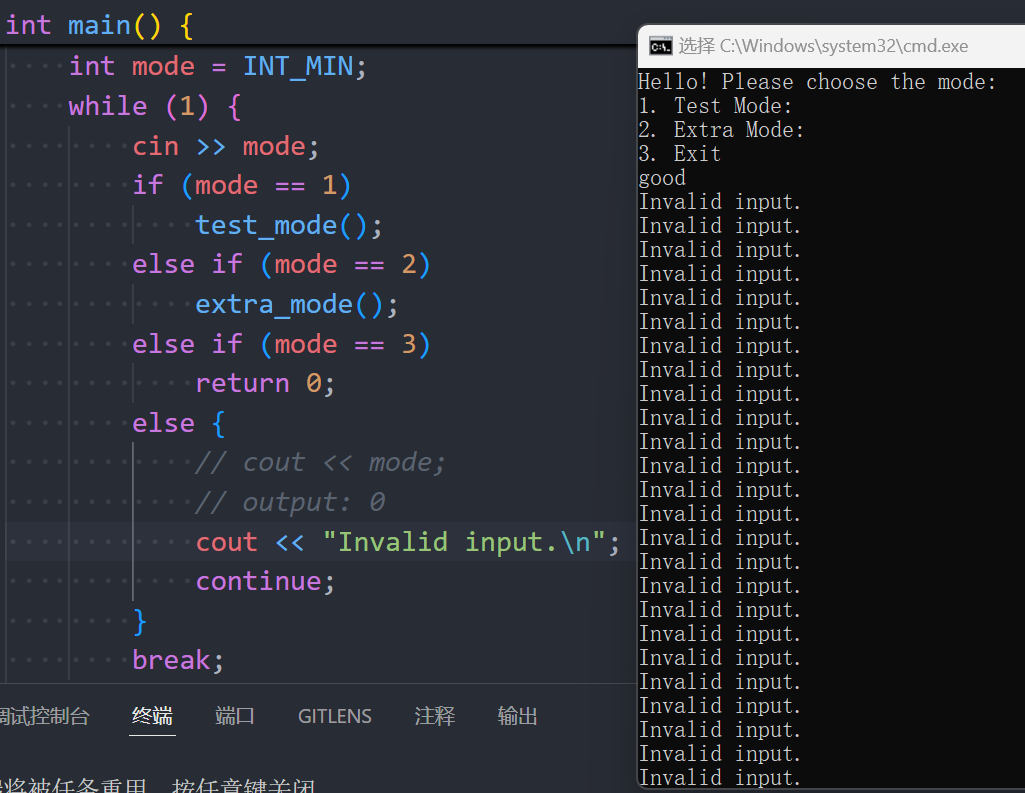
system("cls");

main();

return 0;

}

理想很丰满，现实很骨感。本来以为有个else干预就可以解决用户胡乱输入的问题，但是发现输入类型与int不符的时候，会莫名其妙陷入死循环。



查阅资料发现，cin是输入流，读取数据；>>会从输入流中提取值并储存到后面的变量。如果变量类型不匹配，cin会进入fail state不再尝试读取任何输入，并且把这个错误的东西留在输入缓存区里面，导致以后每一次循环都反复上述的行为。为了解决这个问题，要使用cin.clear()清除错误标志，用cin.ignore()丢弃缓存区中的字符（直到遇见指定字符，这里指定为换行符）。

直接添加如下代码。考虑到超高的复用率，将所有的input操作实现为函数：

void deal\_with\_bad\_input() {

cin.clear();

cin.ignore(INT\_MAX, '\n');

cout << "Invalid input.\n";

}

 void input(auto &x) {

cin >> x;

while (cin.fail()) {

deal\_with\_bad\_input();

cin >> x;

}

}

以上准备工作就结束了，下面逐一阅读任务。

**实验要求与任务实现**

请设计一个精灵类Spirit，其成员要求如下：

1.具有private的data members

（1）name，精灵的名字，键盘输入

（2）x，精灵的位置坐标X轴的值，键盘输入

（3）y，精灵的位置坐标Y轴的值，键盘输入

（4）health，精灵的health生命力初值设为1000

（5）aliveState，精灵的生存状态，初值设为true

2.具有静态数据成员

（1）number，记录已经创建的精灵数量，初值为0

（2）alivenumber，记录生存状态为true的精灵数量，初值为0

非常简单的要求，把中文翻译成代码即可。因为number和alivenumber都是静态的，所以在类外初始化。

class Spirit {

private:

string name;

int x;

int y;

int health = 1000;

bool aliveState = true;

public:

static int number;

static int alivenumber;

};

int Spirit::number = 0;

int Spirit::alivenumber = 0;

3.具有的public的member function

（1）构造函数1，无参数的构造函数，创建精灵对象并初始化，number+1，alivenumber+1

（2）构造函数2，有参数的构造函数（精灵的名字和位置坐标），创建对象并初始化，number+1，alivenumber+1

（3）takeDamage函数：实现对精灵的伤害，首先需要判断aliveState的值，如果是true，将health - damage，若伤害后的精灵health<=0，则health设为0，将aliveState设为false，alivenumber-1

（4）setPosition函数：需要判断新位置是否在0-200之间，如果是的话，就设置精灵新位置，否则报错，仍保持原位置，并输出错误提示信息。

（5）getPositionX函数：返回精灵的位置坐标X

（6）getPositionY函数：返回精灵的位置坐标Y

（7）getName函数：返回精灵的名字

（8）getHealth函数：返回精灵的生命值

（9）getNumber函数：返回精灵世界一共有多少精灵

（10）getAliveNumber函数：返回精灵世界一共有多少alive的精灵

（11）getInfo函数：输出精灵的各种信息，包括姓名、生命值、生存状态、坐标等

要求实现的内容特别多，不过也是按部就班实现。

特地提一下一些特别的地方：

初始化中，并没有加上对异常位置的判定。这点在下面的一个任务里面需要特别注意。

takeDamage函数中，函数类型为int类型而非void类型，返回值代指精灵在受伤前是否死亡以及受伤前未死亡的前提下受伤后是否死亡，用于后续额外任务，之后会作说明。

setPosition函数的要求特别不负责任：出现异常了就输出invalid position之后不做任何操作并保持原位置，那在额外任务中，玩家移动到这个非法位置的话怎么办？直接让玩家跳过这个回合？这显然不合理。所以在调用这个函数的时候要注意这些处理。

代码如下：

int takeDamage(int damage) {

if (aliveState) {

cout << "Spirit " << name << " has been attacked and lost " << damage << " health." << endl;

health -= damage;

if (health <= 0) {

health = 0;

aliveState = false;

alivenumber--;

cout << "Spirit is dead." << endl;

return 0; // Dead

}

return 1;

} else

return -1; // Error

}

void setPosition(int x, int y, pair<int, int> range = {0, 200}) {

if (in\_range(vector<int>{x, y}, range)) {

this->x = x;

this->y = y;

} else {

cout << "Invalid position." << endl;

}

}

void changeName(string name) {

this->name = name;

}

int getPositionX() {

return x;

}

int getPositionY() {

return y;

}

string getName() {

return name;

}

int getHealth() {

return health;

}

int getAliveNumber() {

return alivenumber;

}

void getInfo() {

cout << "Name: " << name << endl;

cout << "Health: " << health << endl;

cout << "Position: (" << x << ", " << y << ")" << endl;

cout << "Alive: " << (aliveState ? "Yes" : "No") << endl;

cout << endl;

}

4.main函数功能

（1）在main函数中通过不同的构造函数创建不同精灵。

（2）调用takeDamage函数，测试精灵的生存状态。

（3）测试类中定义的所有函数。比如设置新位置，输出精灵的各种信息等。

（4）设置对象数组管理10个精灵并初始化

（5）用随机数模拟10次随机事件，即对精灵i的k点伤害，输出相关信息，随机事件结束后，输出所有对象的基本信息

直接把一系列操作写到函数里面，具体操作在前面已经提及。

void test\_mode() {

// task 1

string name1;

cout << "Please enter the name of the first spirit: \n";

input(name1);

cout << "Please enter the position of the first spirit, separated by a space: \n";

int x1, y1;

input(x1), input(y1);

while (!in\_range(vector<int>{x1, y1}, {0, 200})) {

cout << "Invalid position.\n";

input(x1), input(y1);

}

Spirit s1(name1, x1, y1);

s1.getInfo();

Spirit s2 = Spirit();

string name2;

cout << "Please enter the name of the second spirit: \n";

input(name2);

cout << "Please enter the position of the second spirit, separated by a space: \n";

int x2, y2;

input(x2), input(y2);

while (!in\_range(vector<int>{x2, y2}, {0, 200})) {

cout << "Invalid position.\n";

input(x2), input(y2);

}

s2.changeName(name2);

s2.setPosition(x2, y2);

s2.getInfo();

system("pause");

system("cls");

// task 2

cout << "Assume that s1 is attacked by 999 damage and 1 damage." << endl;

s1.takeDamage(999);

s1.getInfo();

s1.takeDamage(1);

s1.getInfo();

system("pause");

system("cls");

// task 3

cout << "Test all the functions." << endl;

cout << "Position of s1: (" << s1.getPositionX() << ", " << s1.getPositionY() << ")" << endl;

cout << "Name of s1: " << s1.getName() << endl;

cout << "Health of s1: " << s1.getHealth() << endl;

cout << "Alive number: " << s1.getAliveNumber() << endl;

cout << endl;

system("pause");

system("cls");

// task 4

cout << "Randomly generate 10 spirits." << endl;

Spirit spirits[10];

for (int i = 0; i < 10; i++) {

int x = get\_rand\_num(0, 200);

int y = get\_rand\_num(0, 200);

string name = "Spirit" + to\_string(i);

spirits[i] = Spirit(name, x, y);

spirits[i].getInfo();

}

system("pause");

system("cls");

// task 5

cout << "Randomly attack 10 spirits." << endl;

for (int i = 0; i < 10; i++) {

int object = get\_rand\_num(0, 9);

int damage = get\_rand\_num(0, 1000);

spirits[object].takeDamage(damage);

}

s1.getInfo();

s2.getInfo();

for (int i = 0; i < 10; i++)

spirits[i].getInfo();

cout << "Task finished." << endl;

system("pause");

system("cls");

cout << Spirit::number << ' ' << Spirit::alivenumber << endl; // 22 20

Spirit::number = 0;

Spirit::alivenumber = 0;

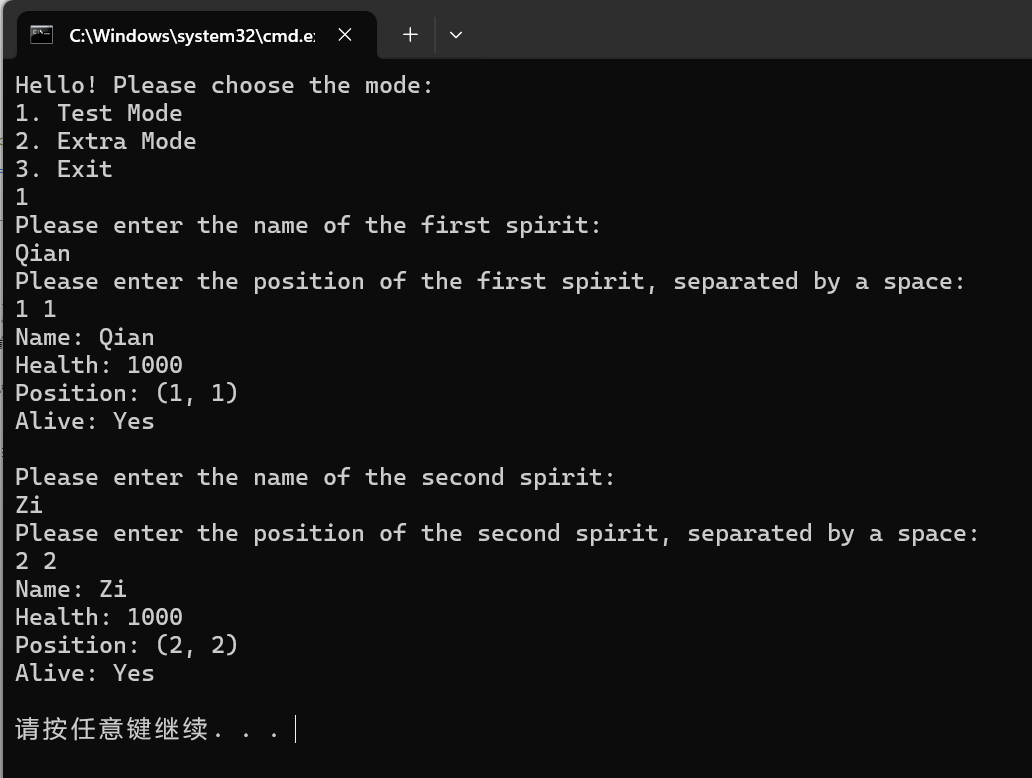
cout << Spirit::number << ' ' << Spirit::alivenumber << endl; // 0 0

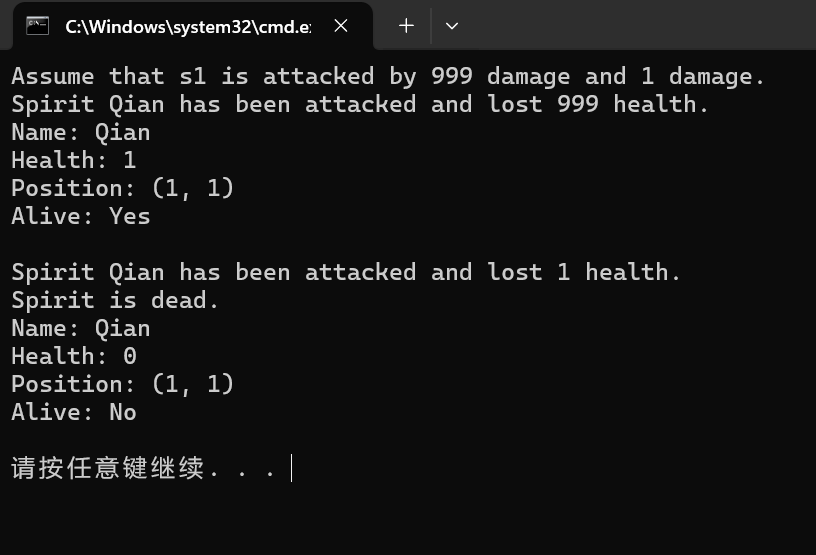
system("pause");

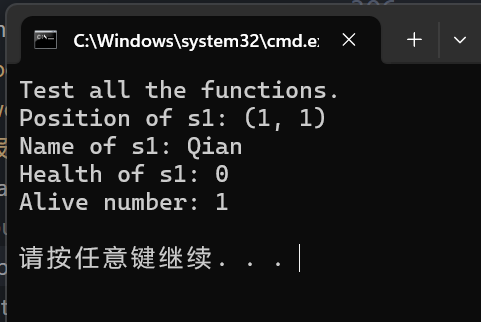
system("cls");

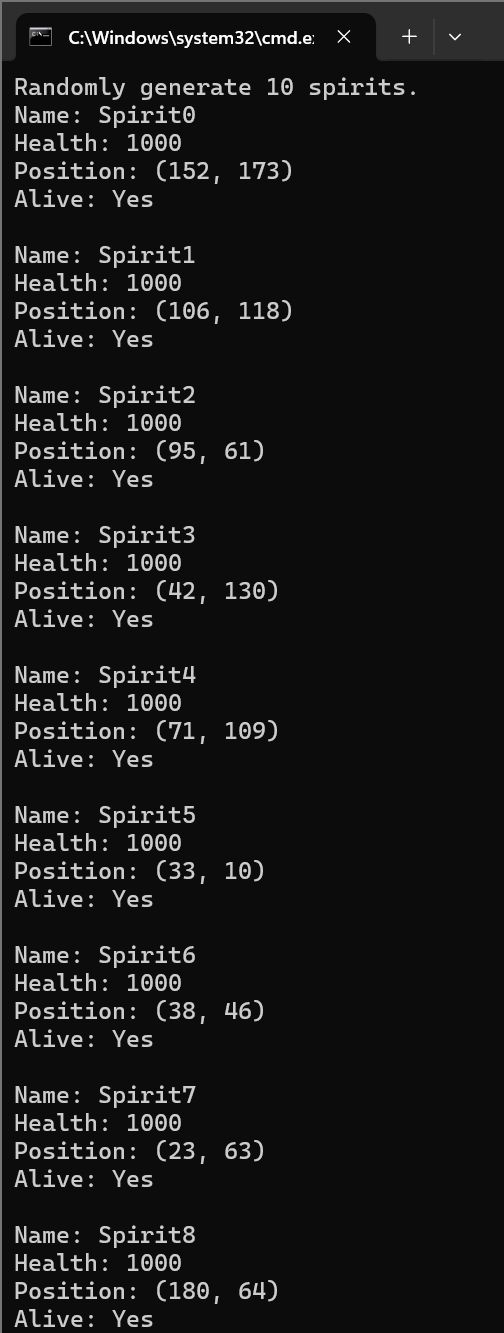
}

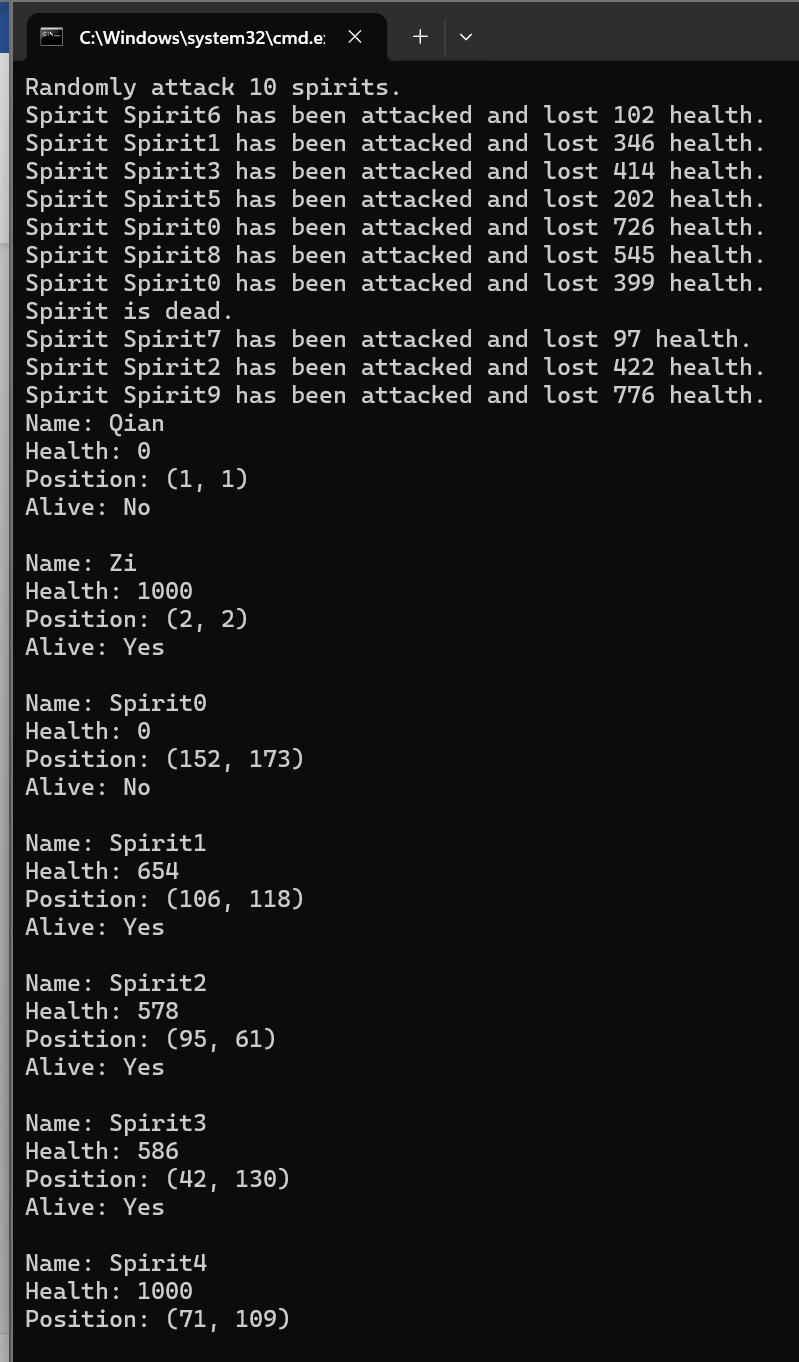
测试结果如下面一系列图片所示，经检验无问题。











接下来就是额外任务了。我们设想做一个PVE的游戏。

为了方便管理，把玩家单独当做一个特殊的对象处理，灵魂一起管理，存在一个vector。

题目给的要求是输出地图，地图的大小是200个，不用实践都知道这200\*200的地图输在控制台上会有多丑。于是我们规定玩家输入地图大小，而且地图大小只能是5-20.

然后生成地图。这里我们做一个巧妙的处理：规定玩家与Spirits都不能站在同一个格子，用地图来表示它们所在的位置，同时索引它们自己！例如，我将玩家的id当成-1，玩家站在(1,1)处，那么mp[1][1]=-1.这样我们实现技能的时候就更加容易了!另外精灵如果死了，要记得把它的位置用0覆盖。

然后是设计技能。技能描述如下：（技能刚开始是我用英文写的，这里的中文用GPT翻译了一下，这里的“灵魂”应该是“精灵”的意思）

**游戏规则：**

1. 每回合玩家可以移动灵魂或使用技能。
2. 如果玩家使用技能杀死了一个灵魂，那么在敌人灵魂的回合它将不会行动。
3. 地图上的灵魂会随机使用它们的技能。
4. 灵魂将按照它们的ID顺序使用技能。如果前一个灵魂杀死了后一个灵魂，后一个灵魂仍然会使用它的技能，然后在这回合结束前死亡。
5. 当玩家或地图上所有其他灵魂死亡时，游戏结束。
6. 如果玩家死亡，游戏失败。

**灵魂的技能：**

1. 横扫（15%）：灵魂会攻击所有其他灵魂，造成0到200之间的随机伤害。
2. 普通攻击（30%）：灵魂会攻击玩家，造成0到10之间的随机范围和0到100之间的随机伤害。
3. 新生（5%）：灵魂会在地图上创建一个新的灵魂。
4. 移动（30%）：灵魂会移动到一个随机位置。
5. 治疗（5%）：灵魂会治疗自己，恢复0到1000之间的随机生命值。
6. 十字斩（15%）：灵魂会攻击同一行和同一列的所有其他灵魂，造成0到1000之间的随机伤害。

**玩家的技能：**

1. 横扫：玩家会攻击所有其他灵魂，造成0到200之间的随机伤害。
2. 死亡来临：随机杀死地图上一个活着的灵魂。
3. 治疗：玩家会治疗自己，恢复0到1000之间的随机生命值。
4. 十字斩：玩家会攻击同一行和同一列的所有其他灵魂，造成500点伤害。
5. 移动：玩家会移动到选定的位置。

这里的概率我们用get\_skill函数实现。为了清晰表示这个函数是用来干什么的，我将它写成匿名函数。代码如下：

void extra\_mode() {

int map\_max\_index;

cout << "Please enter the max index of the map (5-20), the index starts from 0: ";

while (1) {

input(map\_max\_index);

if (map\_max\_index < 5 || map\_max\_index > 20)

cout << "Invalid input.\n";

else

break;

}

vector<vector<int>> mp(map\_max\_index + 1, vector<int>(map\_max\_index + 1, 0));

auto print\_map = [&]() {

for (int i = 0; i <= map\_max\_index; i++) {

for (int j = 0; j <= map\_max\_index; j++) {

if (mp[i][j] == 0) cout << '#';

if (mp[i][j] > 0) cout << '\*';

if (mp[i][j] == -1) cout << 'P';

}

cout << endl;

}

};

cout << "Please enter the name of your spirit: \n";

string name;

input(name);

cout << "Please enter the number of your opponents (1-10): \n";

int n = 0;

while (1) {

input(n);

if (n < 1 || n > 10) {

cout << "Invalid input.\n";

continue;

}

break;

}

system("cls");

cout << "Randomly generate " << n << " spirits." << endl;

vector<Spirit> spirits;

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

int x = get\_rand\_num(0, map\_max\_index);

int y = get\_rand\_num(0, map\_max\_index);

if (mp[x][y] > 0) {

i--;

continue;

}

mp[x][y] = i + 1;

string name = "Spirit" + to\_string(i + 1);

spirits.push\_back(Spirit(name, x, y));

spirits[i].getInfo();

}

system("pause");

system("cls");

cout << "The map is as follows:" << endl;

print\_map();

cout << endl;

cout << "Please decide the position of your spirit, separated by a space: \n";

cout << "Remember, you can't place your spirit on the same position as the other spirits.\n";

int x, y;

while (1) {

input(x), input(y);

if (!in\_range(vector<int>{x, y}, {0, map\_max\_index}) || mp[x][y]) {

cout << "Invalid position.\n";

continue;

}

break;

}

mp[x][y] = -1;

Spirit player(name, x, y);

system("cls");

cout << "The map is as follows:" << endl;

print\_map();

cout << "Your spirit has been placed on the map." << endl;

system("pause");

system("cls");

Print\_Rules();

cout << "The game is about to start." << endl;

for (int round = 0; Spirit::alivenumber > 1; round++) {

cout << "Round " << round << endl;

cout << "Your spirit: " << endl;

player.getInfo();

cout << "The spirits: " << endl;

for (int i = 0; i < n; i++)

spirits[i].getInfo();

cout << "The map:" << endl;

print\_map();

system("pause");

cout << "Please choose your action: \n";

cout << "Skill 1: Sweep Away\n";

cout << "Skill 2: The Death is Coming\n";

cout << "Skill 3: Heal\n";

cout << "Skill 4: Cross Slash\n";

cout << "Skill 5: Move\n";

cout << "If you forget the rules, please enter 6.\n";

int skill;

input(skill);

if (skill < 1 || skill > 6) {

cout << "Invalid input.\n";

round--;

continue;

}

if (skill == 6) {

Print\_Rules(1);

round--;

continue;

}

if (skill == 1) {

cout << "You use Sweep Away.\n";

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

if (mp[spirits[i].getPositionX()][spirits[i].getPositionY()] == i) {

int damage = get\_rand\_num(0, map\_max\_index);

int alive = spirits[i].takeDamage(damage);

if (alive == 0)

mp[spirits[i].getPositionX()][spirits[i].getPositionY()] = 0;

}

}

}

if (skill == 2) {

cout << "You use The Death is Coming." << endl;

int object = get\_rand\_num(0, n - 1);

while (mp[spirits[object].getPositionX()][spirits[object].getPositionY()] != object + 1) {

object = get\_rand\_num(0, n - 1);

}

int alive = spirits[object].takeDamage(INT\_MAX);

if (alive == 0)

mp[spirits[object].getPositionX()][spirits[object].getPositionY()] = 0;

}

if (skill == 3) {

cout << "You use Heal." << endl;

int heal = get\_rand\_num(0, 1000);

player.takeDamage(-heal);

}

if (skill == 4) {

cout << "You use Cross Slash." << endl;

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

if (mp[spirits[i].getPositionX()][spirits[i].getPositionY()] == i + 1) {

int alive = spirits[i].takeDamage(500);

if (alive == 0)

mp[spirits[i].getPositionX()][spirits[i].getPositionY()] = 0;

}

}

}

if (skill == 5) {

cout << "You use Move." << endl;

cout << "Please input 2 numbers, seperated by a space:";

int x, y;

while (true) {

input(x), input(y);

if (in\_range(vector<int>{x, y}, {0, map\_max\_index}) && mp[x][y] == 0) {

mp[player.getPositionX()][player.getPositionY()] = 0;

player.setPosition(x, y);

mp[x][y] = -1;

break;

} else {

cout << "Invalid position." << endl;

}

}

}

cout << "Now the map is as follows:" << endl;

print\_map();

system("pause");

system("cls");

cout << "The spirits use their skill." << endl;

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

if (mp[spirits[i].getPositionX()][spirits[i].getPositionY()] != i + 1) continue;

auto get\_skill = [&]() {

int rand\_num = get\_rand\_num(1, 100);

if (rand\_num <= 15) return 1; // Sweep Away

if (rand\_num <= 45) return 2; // Odinary Attack

if (rand\_num <= 50) return 3; // New Born

if (rand\_num <= 80) return 4; // Move

if (rand\_num <= 85) return 5; // Heal

return 6; // Cross Slash

};

int skill = get\_skill();

if (skill == 1) {

cout << "Spirit " << spirits[i].getName() << " uses Sweep Away." << endl;

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

if (j != i && mp[spirits[j].getPositionX()][spirits[j].getPositionY()] == j + 1) {

int damage = get\_rand\_num(0, map\_max\_index);

int alive = spirits[j].takeDamage(damage);

if (alive == 0)

mp[spirits[j].getPositionX()][spirits[j].getPositionY()] = 0;

}

}

if (mp[player.getPositionX()][player.getPositionY()] == -1) {

int damage = get\_rand\_num(0, map\_max\_index);

player.takeDamage(damage);

}

}

if (skill == 2) {

cout << "Spirit " << spirits[i].getName() << " uses Odinary Attack." << endl;

if (mp[player.getPositionX()][player.getPositionY()] == -1) {

int range = get\_rand\_num(0, 10);

if (abs(player.getPositionX() - spirits[i].getPositionX()) <= range && abs(player.getPositionY() - spirits[i].getPositionY()) <= range) {

int damage = get\_rand\_num(0, 100);

player.takeDamage(damage);

}

}

}

if (skill == 3) {

cout << "Spirit " << spirits[i].getName() << " uses New Born." << endl;

int x = get\_rand\_num(0, map\_max\_index);

int y = get\_rand\_num(0, map\_max\_index);

if (mp[x][y] == 0) {

mp[x][y] = n + 1;

string name = "Spirit" + to\_string(n + 1);

spirits.push\_back(Spirit(name, x, y));

spirits[n].getInfo();

n++;

}

}

if (skill == 4) {

cout << "Spirit " << spirits[i].getName() << " uses Move." << endl;

mp[spirits[i].getPositionX()][spirits[i].getPositionY()] = 0;

int x = get\_rand\_num(0, map\_max\_index);

int y = get\_rand\_num(0, map\_max\_index);

if (mp[x][y] == 0) {

mp[x][y] = i + 1;

spirits[i].setPosition(x, y);

} else {

mp[spirits[i].getPositionX()][spirits[i].getPositionY()] = i + 1;

}

}

if (skill == 5) {

cout << "Spirit " << spirits[i].getName() << " uses Heal." << endl;

int heal = get\_rand\_num(0, 1000);

spirits[i].takeDamage(-heal);

}

if (skill == 6) {

cout << "Spirit " << spirits[i].getName() << " uses Cross Slash." << endl;

int damage = get\_rand\_num(0, 1000);

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

if (j != i && mp[spirits[j].getPositionX()][spirits[j].getPositionY()] == j) {

spirits[j].takeDamage(damage);

}

}

if (mp[player.getPositionX()][player.getPositionY()] == -1) {

player.takeDamage(damage);

}

}

}

cout << "The map is as follows:" << endl;

print\_map();

}

if (player.getHealth() <= 0)

cout << "You lose\n";

else

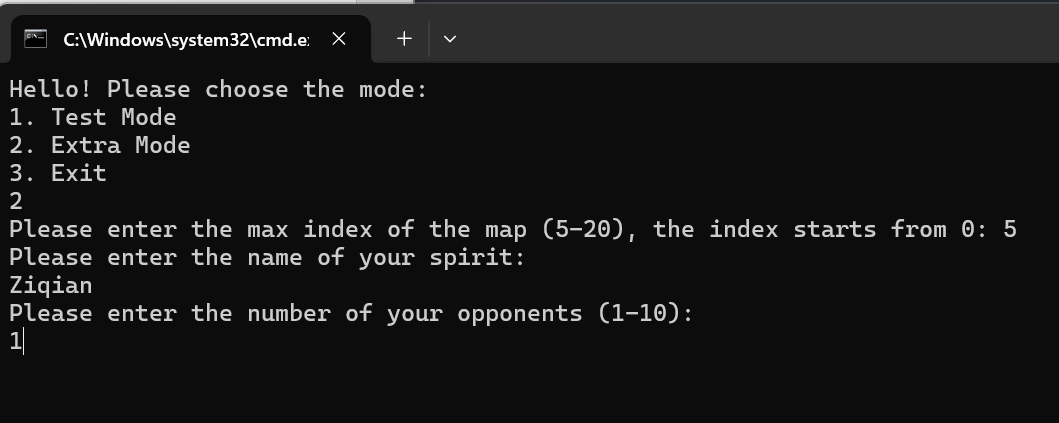
cout << "You win.\n";

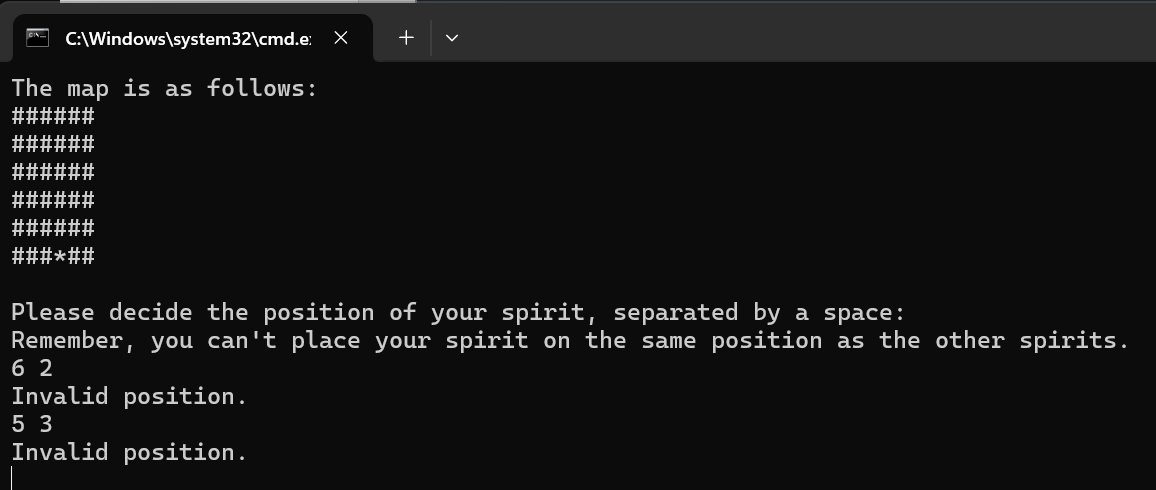
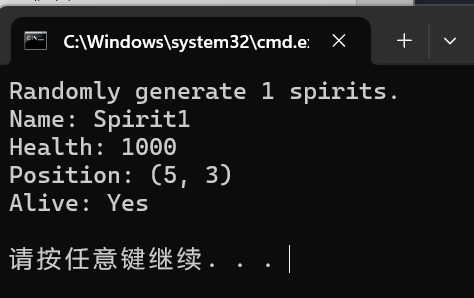
system("pause");

system("cls");

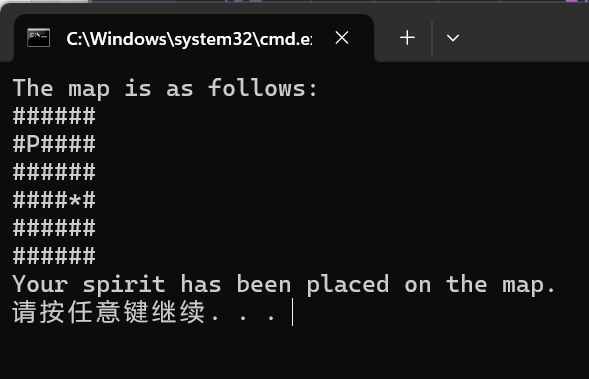
}

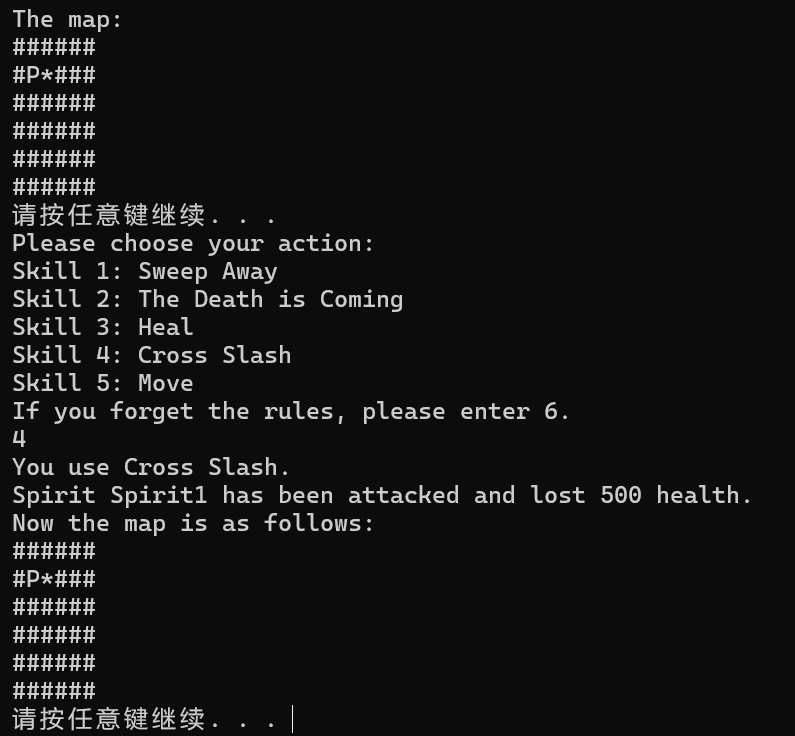
效果如下图所示，因为截图的时候发现一些bug，改了之后重新运行了，部分显示会不一样。

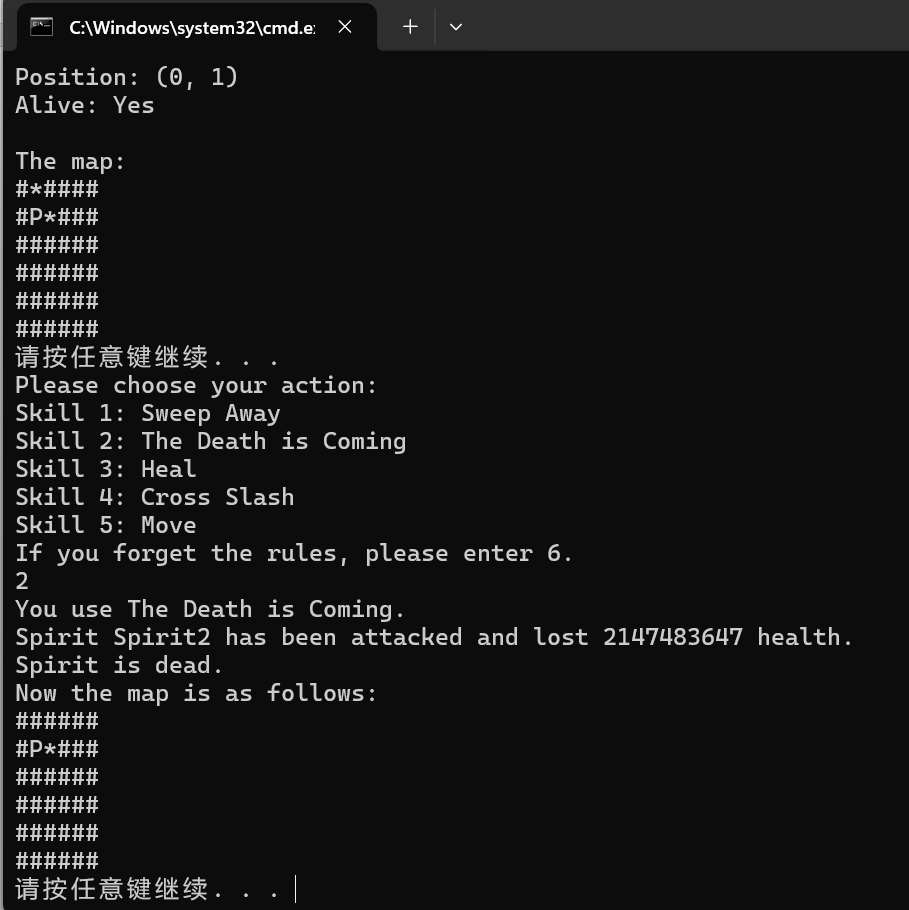




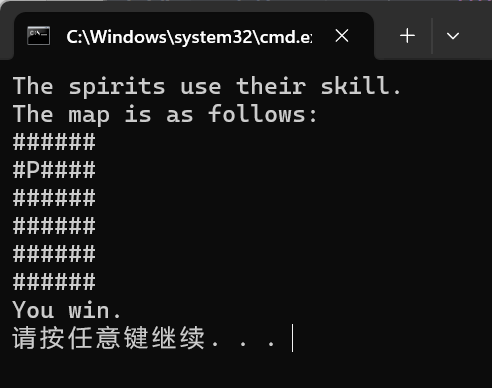
（第一个是越界了，第二个是重合了，这些都会很好地判定）











完整进行了一局游戏并且没有bug，可喜可贺。

完整代码如下：

#include <bits/stdc++.h>

using namespace std;

// use mt19937 to generate random number

int get\_rand\_num(int lower, int upper) {

mt19937 rd(random\_device{}());

uniform\_int\_distribution<int> dist(lower, upper);

return dist(rd);

}

bool in\_range(const auto &r, const pair<int, int> &range) {

for (const auto &i : r)

if (!(i >= range.first && i <= range.second)) return false;

return true;

}

void deal\_with\_bad\_input() {

cin.clear();

cin.ignore(INT\_MAX, '\n');

cout << "Invalid input.\n";

}

void input(auto &x) {

cin >> x;

while (cin.fail()) {

deal\_with\_bad\_input();

cin >> x;

}

}

class Spirit {

private:

string name;

int x;

int y;

int health = 1000;

bool aliveState = true;

public:

static int number;

static int alivenumber;

Spirit() {

number++;

alivenumber++;

}

Spirit(string name, int x, int y) {

this->name = name;

this->x = x;

this->y = y;

number++;

alivenumber++;

}

int takeDamage(int damage) {

if (aliveState) {

cout << "Spirit " << name << " has been attacked and lost " << damage << " health." << endl;

health -= damage;

if (health <= 0) {

health = 0;

aliveState = false;

alivenumber--;

cout << "Spirit is dead." << endl;

return 0; // Dead

}

return 1;

} else

return -1; // Error

}

void setPosition(int x, int y, pair<int, int> range = {0, 200}) {

if (in\_range(vector<int>{x, y}, range)) {

this->x = x;

this->y = y;

} else {

cout << "Invalid position." << endl;

}

}

void changeName(string name) {

this->name = name;

}

int getPositionX() {

return x;

}

int getPositionY() {

return y;

}

string getName() {

return name;

}

int getHealth() {

return health;

}

int getAliveNumber() {

return alivenumber;

}

void getInfo() {

cout << "Name: " << name << endl;

cout << "Health: " << health << endl;

cout << "Position: (" << x << ", " << y << ")" << endl;

cout << "Alive: " << (aliveState ? "Yes" : "No") << endl;

cout << endl;

}

};

int Spirit::number = 0;

int Spirit::alivenumber = 0;

void Print\_Rules(bool only\_skill = false) {

if (!only\_skill) {

cout << "The rule is as follows:\n";

cout << "1. You can move your spirit or use a skill in a round.\n";

cout << "2. If you use a skill and kill a spirit, it won't do anything when it's the spirits' move.\n";

cout << "3. The spirits on the map will use their skill randomly.\n";

cout << "4. The spirits will use their skill in the order of their id. If the previous spirit kills the after spirit, the after spirit will still use its skill and die before the next round.\n";

cout << "5. The game will end when you or all the other spirits on the map are dead.\n";

cout << "6. If you are dead, you lose the game.\n";

cout << endl;

}

cout << "The skill of the spirits is as follows:\n";

cout << "1. Sweep Away(15%): The spirit will attack all the OTHER spirits with a random damage between 0 and 200.\n";

cout << "2. Odinary Attack(30%): The spirit will attack the player with a random range between 0 and 10 and a random damage between 0 and 100.\n";

cout << "3. New Born(5%): The spirit will create a new spirit on the map.\n";

cout << "4. Move(30%): The spirit will move to a random place.\n";

cout << "5. Heal(5%): The spirit will heal itself with a random health between 0 and 1000.\n";

cout << "6. Cross Slash(15%): The spirit will all the OTHER spirits in the same line and column with a random damage between 0 and 1000.\n";

cout << "\n";

cout << "The skill of the player is as follows:\n";

cout << "1. Sweep Away: The player will attack all the OTHER spirits with a random damage between 0 and 200.\n";

cout << "2. The Death is Coming: Kill a random alive spirit in the map.\n";

cout << "3. Heal: The player will heal itself with a random health between 0 and 1000.\n";

cout << "4. Cross Slash: The player will attack all the OTHER spirits in the same line and column with 500 damage.\n";

cout << "5. Move: The player will move to a selected place.\n";

system("pause");

system("cls");

}

void test\_mode() {

// task 1

string name1;

cout << "Please enter the name of the first spirit: \n";

input(name1);

cout << "Please enter the position of the first spirit, separated by a space: \n";

int x1, y1;

input(x1), input(y1);

while (!in\_range(vector<int>{x1, y1}, {0, 200})) {

cout << "Invalid position.\n";

input(x1), input(y1);

}

Spirit s1(name1, x1, y1);

s1.getInfo();

Spirit s2 = Spirit();

string name2;

cout << "Please enter the name of the second spirit: \n";

input(name2);

cout << "Please enter the position of the second spirit, separated by a space: \n";

int x2, y2;

input(x2), input(y2);

while (!in\_range(vector<int>{x2, y2}, {0, 200})) {

cout << "Invalid position.\n";

input(x2), input(y2);

}

s2.changeName(name2);

s2.setPosition(x2, y2);

s2.getInfo();

system("pause");

system("cls");

// task 2

cout << "Assume that s1 is attacked by 999 damage and 1 damage." << endl;

s1.takeDamage(999);

s1.getInfo();

s1.takeDamage(1);

s1.getInfo();

system("pause");

system("cls");

// task 3

cout << "Test all the functions." << endl;

cout << "Position of s1: (" << s1.getPositionX() << ", " << s1.getPositionY() << ")" << endl;

cout << "Name of s1: " << s1.getName() << endl;

cout << "Health of s1: " << s1.getHealth() << endl;

cout << "Alive number: " << s1.getAliveNumber() << endl;

cout << endl;

system("pause");

system("cls");

// task 4

cout << "Randomly generate 10 spirits." << endl;

Spirit spirits[10];

for (int i = 0; i < 10; i++) {

int x = get\_rand\_num(0, 200);

int y = get\_rand\_num(0, 200);

string name = "Spirit" + to\_string(i);

spirits[i] = Spirit(name, x, y);

spirits[i].getInfo();

}

system("pause");

system("cls");

// task 5

cout << "Randomly attack 10 spirits." << endl;

for (int i = 0; i < 10; i++) {

int object = get\_rand\_num(0, 9);

int damage = get\_rand\_num(0, 1000);

spirits[object].takeDamage(damage);

}

s1.getInfo();

s2.getInfo();

for (int i = 0; i < 10; i++)

spirits[i].getInfo();

cout << "Task finished." << endl;

system("pause");

system("cls");

cout << Spirit::number << ' ' << Spirit::alivenumber << endl; // 22 20

Spirit::number = 0;

Spirit::alivenumber = 0;

cout << Spirit::number << ' ' << Spirit::alivenumber << endl; // 0 0

system("pause");

system("cls");

}

void extra\_mode() {

int map\_max\_index;

cout << "Please enter the max index of the map (5-20), the index starts from 0: ";

while (1) {

input(map\_max\_index);

if (map\_max\_index < 5 || map\_max\_index > 20)

cout << "Invalid input.\n";

else

break;

}

vector<vector<int>> mp(map\_max\_index + 1, vector<int>(map\_max\_index + 1, 0));

auto print\_map = [&]() {

for (int i = 0; i <= map\_max\_index; i++) {

for (int j = 0; j <= map\_max\_index; j++) {

if (mp[i][j] == 0) cout << '#';

if (mp[i][j] > 0) cout << '\*';

if (mp[i][j] == -1) cout << 'P';

}

cout << endl;

}

};

cout << "Please enter the name of your spirit: \n";

string name;

input(name);

cout << "Please enter the number of your opponents (1-10): \n";

int n = 0;

while (1) {

input(n);

if (n < 1 || n > 10) {

cout << "Invalid input.\n";

continue;

}

break;

}

system("cls");

cout << "Randomly generate " << n << " spirits." << endl;

vector<Spirit> spirits;

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

int x = get\_rand\_num(0, map\_max\_index);

int y = get\_rand\_num(0, map\_max\_index);

if (mp[x][y] > 0) {

i--;

continue;

}

mp[x][y] = i + 1;

string name = "Spirit" + to\_string(i + 1);

spirits.push\_back(Spirit(name, x, y));

spirits[i].getInfo();

}

system("pause");

system("cls");

cout << "The map is as follows:" << endl;

print\_map();

cout << endl;

cout << "Please decide the position of your spirit, separated by a space: \n";

cout << "Remember, you can't place your spirit on the same position as the other spirits.\n";

int x, y;

while (1) {

input(x), input(y);

if (!in\_range(vector<int>{x, y}, {0, map\_max\_index}) || mp[x][y]) {

cout << "Invalid position.\n";

continue;

}

break;

}

mp[x][y] = -1;

Spirit player(name, x, y);

system("cls");

cout << "The map is as follows:" << endl;

print\_map();

cout << "Your spirit has been placed on the map." << endl;

system("pause");

system("cls");

Print\_Rules();

cout << "The game is about to start." << endl;

for (int round = 0; Spirit::alivenumber > 1; round++) {

cout << "Round " << round << endl;

cout << "Your spirit: " << endl;

player.getInfo();

cout << "The spirits: " << endl;

for (int i = 0; i < n; i++)

spirits[i].getInfo();

cout << "The map:" << endl;

print\_map();

system("pause");

cout << "Please choose your action: \n";

cout << "Skill 1: Sweep Away\n";

cout << "Skill 2: The Death is Coming\n";

cout << "Skill 3: Heal\n";

cout << "Skill 4: Cross Slash\n";

cout << "Skill 5: Move\n";

cout << "If you forget the rules, please enter 6.\n";

int skill;

input(skill);

if (skill < 1 || skill > 6) {

cout << "Invalid input.\n";

round--;

continue;

}

if (skill == 6) {

Print\_Rules(1);

round--;

continue;

}

if (skill == 1) {

cout << "You use Sweep Away.\n";

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

if (mp[spirits[i].getPositionX()][spirits[i].getPositionY()] == i) {

int damage = get\_rand\_num(0, map\_max\_index);

int alive = spirits[i].takeDamage(damage);

if (alive == 0)

mp[spirits[i].getPositionX()][spirits[i].getPositionY()] = 0;

}

}

}

if (skill == 2) {

cout << "You use The Death is Coming." << endl;

int object = get\_rand\_num(0, n - 1);

while (mp[spirits[object].getPositionX()][spirits[object].getPositionY()] != object + 1) {

object = get\_rand\_num(0, n - 1);

}

int alive = spirits[object].takeDamage(INT\_MAX);

if (alive == 0)

mp[spirits[object].getPositionX()][spirits[object].getPositionY()] = 0;

}

if (skill == 3) {

cout << "You use Heal." << endl;

int heal = get\_rand\_num(0, 1000);

player.takeDamage(-heal);

}

if (skill == 4) {

cout << "You use Cross Slash." << endl;

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

if (mp[spirits[i].getPositionX()][spirits[i].getPositionY()] == i + 1) {

int alive = spirits[i].takeDamage(500);

if (alive == 0)

mp[spirits[i].getPositionX()][spirits[i].getPositionY()] = 0;

}

}

}

if (skill == 5) {

cout << "You use Move." << endl;

cout << "Please input 2 numbers, seperated by a space:";

int x, y;

while (true) {

input(x), input(y);

if (in\_range(vector<int>{x, y}, {0, map\_max\_index}) && mp[x][y] == 0) {

mp[player.getPositionX()][player.getPositionY()] = 0;

player.setPosition(x, y);

mp[x][y] = -1;

break;

} else {

cout << "Invalid position." << endl;

}

}

}

cout << "Now the map is as follows:" << endl;

print\_map();

system("pause");

system("cls");

cout << "The spirits use their skill." << endl;

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

if (mp[spirits[i].getPositionX()][spirits[i].getPositionY()] != i + 1) continue;

auto get\_skill = [&]() {

int rand\_num = get\_rand\_num(1, 100);

if (rand\_num <= 15) return 1; // Sweep Away

if (rand\_num <= 45) return 2; // Odinary Attack

if (rand\_num <= 50) return 3; // New Born

if (rand\_num <= 80) return 4; // Move

if (rand\_num <= 85) return 5; // Heal

return 6; // Cross Slash

};

int skill = get\_skill();

if (skill == 1) {

cout << "Spirit " << spirits[i].getName() << " uses Sweep Away." << endl;

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

if (j != i && mp[spirits[j].getPositionX()][spirits[j].getPositionY()] == j + 1) {

int damage = get\_rand\_num(0, map\_max\_index);

int alive = spirits[j].takeDamage(damage);

if (alive == 0)

mp[spirits[j].getPositionX()][spirits[j].getPositionY()] = 0;

}

}

if (mp[player.getPositionX()][player.getPositionY()] == -1) {

int damage = get\_rand\_num(0, map\_max\_index);

player.takeDamage(damage);

}

}

if (skill == 2) {

cout << "Spirit " << spirits[i].getName() << " uses Odinary Attack." << endl;

if (mp[player.getPositionX()][player.getPositionY()] == -1) {

int range = get\_rand\_num(0, 10);

if (abs(player.getPositionX() - spirits[i].getPositionX()) <= range && abs(player.getPositionY() - spirits[i].getPositionY()) <= range) {

int damage = get\_rand\_num(0, 100);

player.takeDamage(damage);

}

}

}

if (skill == 3) {

cout << "Spirit " << spirits[i].getName() << " uses New Born." << endl;

int x = get\_rand\_num(0, map\_max\_index);

int y = get\_rand\_num(0, map\_max\_index);

if (mp[x][y] == 0) {

mp[x][y] = n + 1;

string name = "Spirit" + to\_string(n + 1);

spirits.push\_back(Spirit(name, x, y));

spirits[n].getInfo();

n++;

}

}

if (skill == 4) {

cout << "Spirit " << spirits[i].getName() << " uses Move." << endl;

mp[spirits[i].getPositionX()][spirits[i].getPositionY()] = 0;

int x = get\_rand\_num(0, map\_max\_index);

int y = get\_rand\_num(0, map\_max\_index);

if (mp[x][y] == 0) {

mp[x][y] = i + 1;

spirits[i].setPosition(x, y);

} else {

mp[spirits[i].getPositionX()][spirits[i].getPositionY()] = i + 1;

}

}

if (skill == 5) {

cout << "Spirit " << spirits[i].getName() << " uses Heal." << endl;

int heal = get\_rand\_num(0, 1000);

spirits[i].takeDamage(-heal);

}

if (skill == 6) {

cout << "Spirit " << spirits[i].getName() << " uses Cross Slash." << endl;

int damage = get\_rand\_num(0, 1000);

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

if (j != i && mp[spirits[j].getPositionX()][spirits[j].getPositionY()] == j) {

spirits[j].takeDamage(damage);

}

}

if (mp[player.getPositionX()][player.getPositionY()] == -1) {

player.takeDamage(damage);

}

}

}

cout << "The map is as follows:" << endl;

print\_map();

}

if (player.getHealth() <= 0)

cout << "You lose\n";

else

cout << "You win.\n";

system("pause");

system("cls");

}

int main() {

cout << "Hello! Please choose the mode:\n";

cout << "1. Test Mode\n";

cout << "2. Extra Mode\n";

cout << "3. Exit\n";

int mode = INT\_MIN;

while (1) {

input(mode);

if (mode == 1)

test\_mode();

else if (mode == 2)

extra\_mode();

else if (mode == 3)

return 0;

else {

// cout << mode;

// output: 0

cout << "Invalid input.\n";

continue;

}

break;

}

system("pause");

system("cls");

main();

return 0;

}

心得：  
通过这次类与对象的实验，我深刻体会到了面向对象编程的强大之处。在实现精灵类`Spirit`和相关功能时，我不仅加深了对C++语法的理解，还学会了如何有效地管理内存和处理用户输入。此外，通过设计额外的游戏模式，我锻炼了逻辑思维和问题解决能力。这次实验不仅提升了我的编程技能，也增强了我对软件开发的兴趣。