# Introduction of my Final Assignment

# *Endless Journey*

# 《无尽征途》

## 1.游戏介绍：

开局，创建属于自己的角色

角色属性包含攻击、血量（上限）、防御（对方的攻击会经过其按比例削减）、金钱、可装载的特殊技能和可更改的道具

主角通过在主线剧情（*暂时只写了新手教程*）中打怪、去训练场、无尽模式等提升自己的攻击、防御、血量上限，获取金币和道具（*训练场和无尽暂时只写了获得金币*）

也可以去商店购买道具，但有些道具只能打剧情、无尽模式获得（*无尽模式的道具获取我还没写*）

还可以去技能创建台自己设计道具（*当然开发的时候也是在技能创建台写的道具，占据技能列表1000-1999格。玩家则可以在0-999自由发挥*），这很重要，否则过不了新手村。

### 1.1主线剧情：

我真的来不及写了，只做了5关

主线剧情中，玩家每一关，或输或赢，都会得到属性升级（攻击力、防御力、血量上限、金币）。赢了还能得到额外道具并解锁下一关

### 1.2 支线剧情：

剧情文件是从.dat文件读取的，因此只要替换这个文件就可以玩别的剧情，当然我还没来得及写

玩家存档中内置了一个数组，存储玩家在各个剧情中的坐标，最多支持100个。

### 1.3 战斗

战斗功能被剧情模式、训练场模式、无尽模式所使用，玩法是：

1. 回合制，玩家先进攻
2. 每个回合前，可以通过回车选择并释放技能
3. 战斗中，会出现一个闪动的蓄力条，在达到最大值（左边一点点，因为有延迟）时按下空格，可以最大化自己的攻击
4. 把对方的血量打到小于等于0，游戏结束

### 1.4 技能：

技能和道具是游戏最核心的玩法之一，通过在对局中临时改变玩家或者敌人的属性，实现在下一轮（或者整个战斗对局中）的增益效果

### 1.5 商店

有一些开发者创建的默认道具可供玩家购买，不过更建议自己创建

### 1.6 训练场

有五个难度，随即在一定范围内生成对手，赢了可以获得金币

### 1.7 无尽模式

类似于训练场，但是是无限个回合的

打的回合越多，奖励指数增长

每个回合前不会把血量补满，也不允许中途购买道具

每个回合结束后，可以选择退出，得到奖励；也可以继续下一轮，但如果输了，只能拿到少量奖励

## 2.功能介绍与其实现方式：

### 2.1控制功能介绍

### 2.2角色存储实现

攻击、血量（上限）、防御、金钱、特殊技能、可更改的道具，一起组成一个结构体

struct role\_stats  
{  
 char Name[101];  
 int ATK;  
 int HP;  
 float DEF;  
 int Money;  
 int SKILL[2];  
 int PROPS[100][2];  
};

struct 角色属性  
{  
 名字;  
 攻击;  
 血量;  
 防御;//比如=0.9，即对方的攻击会变成原来的90%  
 金钱;  
 特殊技能;//特殊技能是一个技能码，对应到txt  
 道具;  
}

其中特殊技能用一维数组，道具用二维数组存储，包含道具代码、道具剩余数量，比如两个个加金币的道具和一个加攻击的道具长这样：

特殊技能 = {9，1}//表述技能码9，状态可用。每次打架可以用一次，用完改成0.打完更新  
  
道具[100][2];//最多100个道具  
道具 =   
{  
{1,2},//1是金币道具代码，2是个数  
{2,1}//2是加攻击道具代码  
}

#### 2.2.1特殊技能与道具

##### 2.2.1.1特殊技能与道具的定义

特殊技能与道具也是结构体，因为他们在战斗中的实现方式，是对属性执行额外的修改（见下面的介绍）

因此特殊技能与道具结构体是对攻击、血量（上限）、防御、金钱四个属性的映射，长这样：

struct 特殊技能与道具  
{  
 技能发动时的提示词;//比如”已使用技能XXX，对方血量减少20“  
 可使用的场景;//指示在战斗中（2）、非战斗（1）、全部禁用（0）这三种状态，就比如你不能在非战斗情况下使用  
 对自己或是对对方;//1对自己，2对对手  
 对攻击的操作;//是百分比，1.2代表增加到120%  
 对血量的操作;//是整数，增减血量上限或者增加当前血量，做加法  
 对防御的操作;//是百分比，0.8代表防御掉20%.注意在削减对方的时候，要大于一  
 对金钱的操作;//是整数  
}

0道具不可使用，一般用来作为收藏，也可以到某个时机解锁

比如增加一百金钱的结构体，应该这么定义：

struct 特殊技能与道具 金钱礼包  
{  
 "技能xxx已使用，金钱+100!\n";  
 1;//非战斗情况下  
 1;//对自己  
 1;  
 1;  
 1;  
 100；  
}

有一个细节，当状态为非战斗时，肯定是对自己发出的

同时血量这一个参数，非战斗时候加上限，战斗时候加当前值

具体定义代码

struct SKILL\_PROPS//技能结构体  
{  
 char prompt[101];  
 int state;  
 int object;  
 float Operate\_ATK;  
 int Operate\_HP;  
 float Operate\_DEF;  
 int Operate\_Money;  
 int Fight\_Times;//指示战斗中是几个回合有效还是对局有效。1是一回合，2是两个回合，0是整个对局  
 int Price;  
 char Introduction[201];  
 //技能发动时的提示词;//比如”已使用技能XXX，对方血量减少20“  
 //可使用的场景;//指示在战斗中、非战斗在、全部禁用这三种状态，就比如你不能在非战斗情况下使用  
 //对自己或是对对方;  
 //对攻击的操作;//都是百分比，1.2代表增加到120%  
 //对血量的操作;//增减血量上限  
 //对防御的操作;  
 //对金钱的操作;  
};  
typedef struct SKILL\_PROPS SKILL\_PROPS;

所有道具都存储在一个结构体组里面，最多2000个

struct SKILL\_PROPS;  
#define MAX\_SKILLS 2000  
SKILL\_PROPS SKILL\_PROPS\_LIST[MAX\_SKILLS];

##### 2.2.1.2特殊技能与道具在对战或者非对战时对角色属性执行修改的方式

而为了链接这个结构体和属性结构体，定义一个函数，用于根据技能道具，修改属性数值

函数接收特殊技能或道具的代号（2.2.1.3有详细介绍），根据当前场景，是对战或者是非对战时，调用时填充参数。这两个参数

然后先查询角色属性（特殊技能和道具两个数组）中这个代号存不存在（可不可用），若可用，执行对全局变量”角色属性“的修改

已经被写入到常量变量中的

技能是自己主动释放的，而对于怪，也可以设定技能，随机释放或者每次都释放（把怪物的属性里面某个技能的数量改成999）

##### 2.2.1.3特殊技能与道具的书写、读取和写入

~~在一个txt里面写了所有的道具特殊技能，程序启动时打开并筛选为特殊技能、写入内存~~

~~txt的格式长这样：~~

#START  
技能代号（这是唯一的，读取后写入内存，然后比如要展示我现在的技能以及其介绍，我就通过存储在存档文件里面的我的技能代号，先打开这个txt筛选出的”技能“数据，再在里面根据技能代号找技能介绍。  
再比如我要在战斗中使用道具禁用对方的防御，这个道具的代码是11，那么我就输入11。此时通过2.2.1.2提到的函数，）  
技能名称  
技能使用场景//为了把包括获取的礼包/技能、角色特殊技能区分开来而不写在两个文件里，这里指示它。  
技能详细介绍文本（用于商店）  
技能价格（用于商店）  
然后是技能结构体  
 技能发动时的提示词;  
 可使用的场景;  
 对自己或是对对方;  
 对攻击的操作;  
 对血量的操作;  
 对防御的操作;  
 对金钱的操作;  
#END

鉴于中文不好存储，解决方案改为：

1. 写一个技能生成器，开发者用它来写技能
2. 保存整个技能列表结构体为二进制文件
3. 玩家可以通过消耗金币来自己创造技能

*好吧其实中文也没啥问题，用ANSI编码即可。但是需要专门写一个函数用来解析*

### 2.3战斗功能及其实现

每轮战斗，本质上是对角色的属性进行修改，包括血量增减、攻击力增减（越打越强嘛）

每轮产生的攻击计算公式：

$$ 发起攻击的最终数值 = 自己的攻击属性值 战斗时机值 自己技能中对攻击的操作

$$

每轮受到的伤害计算公式：

每轮战斗后计算双方剩余的生命值，<=0就结束，返回结果

战斗开始时，会创建一个新的结构体，结构体存储怪的属性，属性的书写和读取方式和玩家的是一样的

把结构体拆开？主要是血量上限，要把它变成在对局中可以被修改的血量

战斗的模式参考Inusuku的游戏Living With Sister: Monochrome Fantasy，回合制打怪，攻击的具体大小和发动的时机有有关

普攻就是简单的削减怪的血量，被攻击的时候会被削减自己的血量

特殊技能的实现方式，是对属性执行额外的修改，比如：

额外减少对方的血量20%

禁用对方的防御

增加自己的防御指数

增加100金钱（这个可以当作一个宝箱道具）

而在战斗结束后，会对角色的属性执行操作，，包括打赢了/打输了，金币、血量上限、攻击

### 2.4商店功能及其实现

商店是利用金币换取道具的场所。在非战斗场景下，可以进入

商店读取可售卖的道具（道具列表1500-1999），按照其价格售卖给玩家

### 2.5角色数据（即存档）写入和读取

一个txt存储如下几项数据：

1. 角色属性结构体
2. 角色相对于剧情所处的位置

游戏先读取属性结构体，结构体里面的特殊技能、~~道具则需要先读取特殊技能与道具文件，把他们写入到两个变量”特殊技能 列表“和”道具列表“~~用一个列表存储

这样当要调用特殊技能与道具文件信息时，就通过结构体里面两个数组存储的索引查询（最直接的，查询当前可用技能函数）

而怪的数据创建也是一样

### 2.6剧情功能的实现

剧情应当包括： 每一关的编码（坐标）

每一关的蓄力条难度指数，调节蓄力条速度的 每一关的介绍词 每一个怪的属性（结构体stats） 输赢的后果（攻击力、生命值等） 输赢获得的道具（一些特殊技能道具），以技能码形式定义，存储在SKILL\_PROP\_LIST和技能道具文件中， 开始剧情后应该要可以选择，某些剧情可以重复玩，有些只能玩一次，设置一个码来识别（通关一整个个剧情之后应该刷新一遍）

typedef struct  
{  
 char Name[101];  
 char Introduction[401];  
 role\_stats Opponent\_Stats;  
 SKILL\_PROPS Win\_Consequence;  
 SKILL\_PROPS Lose\_Consequence;//这个直接定义在存档文件里，打完之后借用USP函数直接对玩家操作，比如增加100金币  
 int Win\_Award\_Code;//这个是写在SKILL\_PROPS\_LIST里面的，只能通过战斗获得的技能/道具  
 int Repeatable;//指示这个关卡能不能重复玩，比如某些特殊道具只能获得一次，就设置成只能玩一次。0指示可以重复，1指示只能玩一次  
 //这个的判断方式是，检查玩家存档里关卡当前位置码，如果大于这个结构体内定义的Serial\_Num，就不允许。  
 //在整个通关后，修改存档中的关卡位置码为0，即可都玩。  
  
}Story\_Line;

以上，一关存储为一个结构体，整个作为结构体组

玩家的存档包含其在剧情中的坐标，剧情模式的总控函数会解析，然后呈现给玩家可玩的关卡

### 2.7训练场功能的实现

这是主控，monster是随机生成的一个怪物的属性

OPPONENT = monster;  
 if (fight())  
 {  
 PLAYER.Money += OPPONENT.Money;  
 printf("你胜利了！你获得了%d个金币！", OPPONENT.Money);  
 printf("按任意键继续。。。\n");  
 while (!\_kbhit()) {}  
 }  
 else  
 {  
 printf("没实力就别来\n");  
 printf("按任意键继续。。。\n");  
 while (!\_kbhit()) {}  
 }

玩家指定难度后，程序会生成一个怪物，其属性值按照一定比例映射难度值

打赢之后，综合难度值和怪物属性，玩家会获得奖励

int calculate\_reward\_gold(struct role\_stats monster, int difficulty)  
{  
 difficulty \*= 5;  
 // 基础金币奖励与难度成正比  
 int base\_gold = difficulty \* 100;  
  
 // 根据攻击力增加金币奖励  
 int atk\_bonus\_gold = monster.ATK \* 2;  
  
 // 根据生命值增加金币奖励  
 int hp\_bonus\_gold = monster.HP / 2;  
  
 // 防御力越低（防御能力越强），金币奖励越多  
 int def\_bonus\_gold = (int)(100/ monster.DEF);  
  
 // 总金币奖励为各部分之和  
 int total\_gold = base\_gold + atk\_bonus\_gold + hp\_bonus\_gold + def\_bonus\_gold;  
  
 return total\_gold;  
}

### 2.8menu函数功能及其实现

我一开始写的是命令式界面，方便。后来收到罗睿哲启发写了选择式菜单

菜单接收：

1. 按钮数量
2. 按钮名称组
3. 按钮详情组
4. 按钮映射功能码数组
5. 显示模式代码

通过上下左右键或者wasd移动，通过回车或者空格选择，函数会返回按下确认的位置对应数组中的代码

这个函数代替了玩家的命令输入和程序的命令解析，直接返回命令码

原先：

1. 解析命令并返回命令码

int AnalyseCC(const char \*f\_message)  
{  
 if (  
 strstr(f\_message, "退出") ||  
 strstr(f\_message, "Exit") ||  
 strstr(f\_message, "exit") ||  
 strstr(f\_message, "tuichu") ||  
 strstr(f\_message, "TC") ||  
 strstr(f\_message, "tc") ||  
 strstr(f\_message, "0")  
 )  
 {  
 logger(INFO, ConsoleCommand, "[AnalyseCC]-接收到退出游戏关键字，启动退出游戏进程");  
 return 0;  
 }  
  
// 省略。。。。。。  
 else if (  
 strstr(f\_message, "商店") ||  
 strstr(f\_message, "sd") ||  
 strstr(f\_message, "SD") ||  
 strstr(f\_message, "SHOP") ||  
 strstr(f\_message, "shop") ||  
 strstr(f\_message, "Shop") ||  
 strstr(f\_message, "5")  
 )  
 {  
 logger(INFO, ConsoleCommand, "[AnalyseCC]-接收到商店关键字，启动进程");  
 return 5;  
 }  
  
 else if (  
 strstr(f\_message, "9") ||  
 strstr(f\_message, "退出开发者模式")  
 )  
 {  
 logger(INFO, ConsoleCommand, "[AnalyseCC]-接收到退出开发者模式关键字，启动进程");  
 return 9;  
 }  
  
 else  
 {  
 return -1;  
 }  
}

1. 根据命令码打开对应功能

switch (f\_code)  
 {  
 case 0:  
 case -10999://0是退出命令  
 if (last\_f == 0)  
 {  
 PLAYER\_SAVE.Player\_Stats = PLAYER;  
 SaveSaveFile(&PLAYER\_SAVE, &save\_count);  
 logger(INFO, Saver, "游戏存档已自动保存");//保存存档函数  
 exit(0);//退出  
 return 0;  
 }  
 else  
 {  
 printf("确认退出？重复刚才的退出字词即确认\n");  
 Sleep(1000);  
 last\_f = 0;//设置全局变量last\_f的值为当前的f\_code  
  
 return 0;//第一次退出命令时，需要再一次确认  
 }  
 break;  
  
 case 1:  
 last\_f = 1;  
 StoryLineModeControl();  
 return 1;  
 break;  
  
// 省略。。。。。。  
  
 case 8:  
 last\_f = 8;  
 Choose\_My\_Skill();  
 return 1;  
 break;  
 case 9:  
 develop\_mode = 0;//退出开发者模式  
 return 1;  
 break;  
 default://未知命令，返回控制台提示（其实就是Analyse CC返回的-1）  
 logger(INFO, ConsoleCommand, "未知的控制台指令，返回指令列表");  
 printf("————————未知的控制台指令，返回指令列表，注意所有的指令都要加上#cc————————\n");  
 printf("0：退出/tc/TC/tuichu/Exit/exit\n");  
 printf("1：战斗/Fight/fight/zd/ZD/zhandou\n");  
 printf("2：技能创建/道具创建/jncj/djcj/SPC/spc\n");  
 printf("3：剧情查询/剧情创建/剧情生成/jqcx/jqsc/SLC/slc\n");  
 printf("4: 属性查询/属性/人物状态/我/Stats/stats/STATS\n");  
 printf("5: 商店/Shop/shop/SHOP\n");  
 printf("9: 退出开发者模式\n");  
 printf("===============================================\n");  
 break;  
 }  
  
 return 0;

现在：

1. 解析

int cc\_menu\_arr[MAX\_BUTTONS] = { 0,1,2,3,4,5,6,7,8 };  
 const char\* cc\_menu\_buttons[MAX\_BUTTONS] =  
 {  
 "退出",  
 "剧情模式",  
 "技能查询与创建",  
 "剧情查看与生成",  
 "我的",  
 "商店",  
 "训练场",  
 "无尽模式",  
 "装载技能"  
 };  
 const char\* introduction[MAX\_BUTTONS] = {  
 NULL,  
 NULL,  
 "强烈建议使用，尤其剧情模式，先进去看看怪物的特性，然后ESC出来配置技能，对症下药"  
 };  
 if (PLAYER\_SAVE.Story\_Line\_Coordinates[STORY\_LINE\_CODE] < 2)  
 {  
 cc\_menu\_arr[2] = -2;  
 introduction[2] = "技能创建功能在完成剧情第二关之后开放，主要是因为第三关不用自定义技能打不过去，不信你试试";  
 }  
  
 if (PLAYER\_SAVE.Story\_Line\_Coordinates[STORY\_LINE\_CODE] < 5)  
 {  
 cc\_menu\_arr[5] = -5;  
 cc\_menu\_arr[6] = -6;  
 cc\_menu\_arr[7] = -7;  
 introduction[5] = "商店功能在完成剧情第五关之后开放";  
 introduction[6] = "训练场功能在完成剧情第五关之后开放";  
 introduction[7] = "无尽模式在完成剧情第五关之后开放";  
  
 }

1. 映射功能码，根据命令码打开对应功能

（略）

## 3.代码结构介绍：

### 3.1functions.h

头文件

### 3.2GameMain.cpp

包含了文件的加载、启动台

### 3.3Logger.cpp

日志器，这东西我是从Python那边学来的，但我感觉者更适合有错误处理的类似“Try:”语句的，C语言我一出错就停止运行

不过在游戏开发前期，还是很可以指示错误位置的

### 3.4Saver.cpp

这个文件里定义了载入和保存函数

### 3.5ConsoleCommand.cpp

定义了主菜单，负责解析主函数进程中输入的命令（主要是一级菜单发送的）并连接到具体功能函数

还写了一个menu函数，详情见2.8

### 3.6Fight.cpp

被剧情模式、无尽模式、训练场调用

### 3.7Skill\_Props\_Creator.cpp

负责技能创建及其展示

### 3.8Story\_Line.cpp

负责解析剧情结构体和呈现剧情

也负责创建、修改剧情（开发者模式）

### 3.9Train.cpp

包含训练场和随机生成对手函数，后者在3.10也被使用

### 3.10Endless\_Mode.cpp

无尽模式的主要部分

### 3.11Skill\_Props.cpp

负责处理战斗中传来的技能操作，临时修改玩家或者敌人属性

### 3.12Mine.cpp

展示玩家当前的所有属性

### 3.13商店

略

## 4.问题与改进

代码的文件结构还是很杂乱，手生

用了很多全局变量，因为一开始没有完全设计好

占用内存太大，之后改进的话，会更多使用malloc来分配

有很多接口给了但是没有用到，等待更新

另外剧情太短了，不过更新很方便