Hero Era Battleground

By Jed
www.jeddd.com

请先阅读 readme.md。

〇、 术语说明

为避免歧义，特此说明本文档中部分容易混淆的术语。

1. **法术**：此词语与示例文档中的“魔法“一词完全等价，本文档中前者完全替换后者。
2. **基础攻击（防御）力**：是英雄自身的属性，默认为 10。当英雄创建后，该值任何情况下都不会改变。也就是说英雄的基础攻击（防御）力与装备无关。
3. **物攻**：物理攻击，对玩家显示为 Attack Damage，源码中相应函数或变量名为 `phyAtk`。
4. **法攻**：法术攻击，与魔法攻击一词完全等价。对玩家显示为 Ability Power，源码中响应函数或变量名为 `magAtk`。
5. **物抗/法抗**：对玩家分别显示为 Armor 和 Magic Defence，源码中响应函数或变量名为 `phyDef` 和 `magDef`。
6. **合成装备**：作为名词，即为攻击型装备与防御型装备的综合体，故又名“**综合型装备**“；作为动词，即为花费额外金钱将两种装备合二为一的动作。详细定义见下文。
7. **购买装备**：购买装备即购买并穿戴，效果是花钱提升自身属性。不存在买了但是不穿戴的情况。
8. **卖出装备**：卖出装备即脱下装备并卖了换钱。不存在脱下但是不卖掉的情况。
9. **装备库存**：不同于英雄的装备栏。装备库存指的是预先定义好的若干装备，英雄购买装备时，实际上是从装备库存中购买的。

一、 需求分析

这里介绍本次项目的需求，同时包含本次大项目的功能概述。

- **英雄属性和行为**

  基本**数值类属性**：生命值、基础攻击（物攻和法攻）、基础防御（物抗和法抗）、金钱数量、装备数量、暴击率；
  附加**非数值属性**：装备栏。

  查询：场外行为。以表格方式打印出英雄的属性、装备等。
  攻击：主动行为。可以对其它英雄对象造成攻击，目的是减少对方的生命值。
  购买装备：主动行为。减少金钱数量，给装备栏增加内容，目的是提升自己的攻击或防御力。
  合成装备：主动行为。花额外的钱将两个装备合二为一，目的是减少装备总数（因为一个英雄最多同时穿戴三个装备）。
  卖出装备：主动行为。将装备脱下并卖掉换钱，得到装备原价的 60%。
  回血：主动行为。花费金钱，增加自己的生命值。
  防御：被动行为。当自己受到其它英雄攻击时触发，目的是减弱对方给自己数值属性造成的削减，下文有详细的计算方法。
  状态检查：被动行为。每回合任何一方操作过后都对对方的存活/死亡状态进行检查。

- **装备属性**

  价格：购买装备所需要的金钱数。
  加成：一个具体的装备应该能够提升英雄的攻击或防御力。当然，这里说的“提升”并不是直接增加英雄的基础攻击或防御力那个变量，而是在真正发生攻击行为时，使用“基础攻击或防御

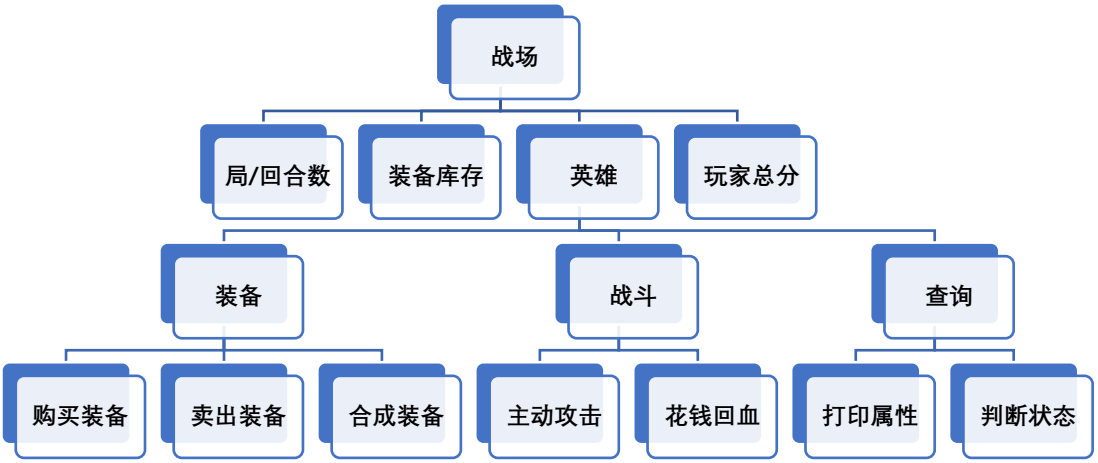
力+装备加成”这样的方式来进行计算。对应函数为 get_real_phyAtk 等四个。计算方法见“数据设计”板块。

- **战场属性**

回合数：玩家英雄和敌人英雄个进行一次操作视为一回合，每个完整的回合内都要进行两次英雄状态检查。
局数：一局可能包含多个回合。一局游戏中，如果有一方英雄死亡，那么该局结束。由玩家选择是否再来一局。
装备库存：内有预先定义好的装备。详细内容见本文档末尾附录。

二、 实现思路

- **功能结构**

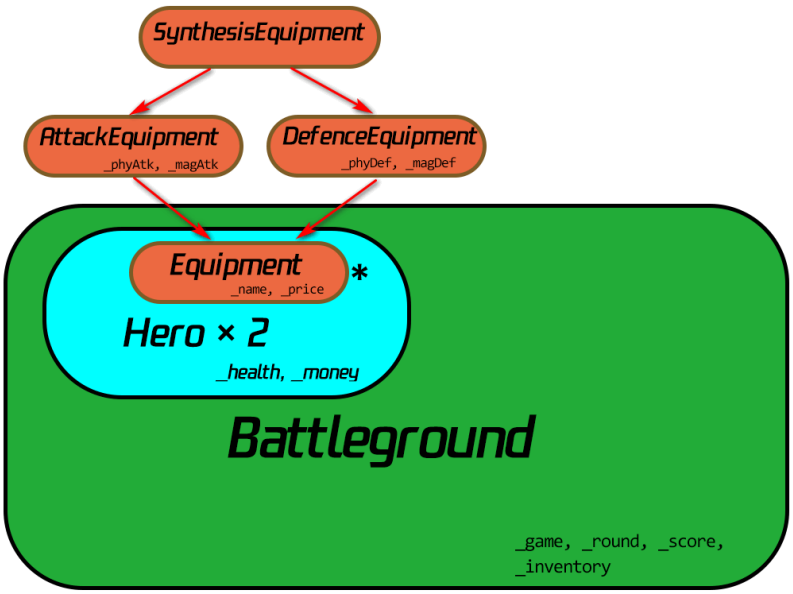


- **包含和继承关系**

战场类（Battleground）中包含两个英雄，分别是由用户控制的**玩家英雄**和由电脑控制的**敌人英雄**，此外还有表示**装备库存**、**当前局数**、**当前回合数**、**玩家总得分**等成员变量。

一个英雄类（Hero）含有诸如生命值、各种攻击防御力的数据成员，同时还有含有指向装备的指针的 vector 用于储存该英雄已购买的装备。

装备类（Equipment）派生出攻击型装备类（AttackEquipment）和防御型装备类（DefenceEquipment），后二者又共同派生出综合型装备类（SynthesisEquipment）。英雄类中的 vector 储存的是 Equipment*，即指向基类的指针，由它可以调用任何派生装备（向



上造型），在使用装备时，再用 dynamic_cast 向下造型即可知道该装备是什么类型的装备。

图注说明：大框在小框外围表示**包含关系**，星号表示**以指针的方式包含**；红色箭头表示**继承关系**，由派生类指向基类。每个类右下角以下划线开头的单词是该类中较为重要的**数据成员**（并非所有成员都出现在了上图中）。

三、 数据设计

- **装备系列类**

装备系列类包含 8 个文件：Equipment.h(.cpp)、AttackEquipment.h(.cpp)、DefenceEquipment.h(.cpp)、SynthesisEquipment.h(.cpp)

间接派生类	直接派生类	基类	成员	数据类型	备注
合成装备	攻击装备	装备	_name，装备名称	string	装备的名称
			_price，装备价格	price	购买装备需要花费的金钱数
			_phyAtk，物攻加成	int	物理攻击数值增量
			_magAtk，法攻加成	int	法术攻击数值增量
	防御装备	装备[2]	_name，装备名称	string	装备的名称
			_price，装备价格	price	购买装备需要花费的金钱数
			_phyDef，物抗加成	int	物理防御数值增量
			_magDef，法抗加成	int	法术防御数值增量

备注：

1. 合成装备的命名方法：一个合成装备（SynthesisEquipment）由一个攻击型装备（AttackEquipment）和一个防御型装备（DefenceEquipment）组合而成，其名称（_name）格式如下："**攻击型装备名-plus-防御型装备名**"，如：knife 与 shield 合成为 knife-plus-shield。此功能在构造函数中实现。
2. AttackEquipment 类和 DefenceEquipment 继承 Equipment 的方式为虚继承（virtual public）。

- **英雄（Hero）类**

包含文件：Hero.h(.cpp)
此类中成员较多，因此使用表格列出所有成员函数。

成员函数	接收参数	返回值类型及含义	备注
构造函数			
Hero	共 7 个[2]	-	
基本操作			
get_name	void	string，英雄名称	
get_health	void	int，英雄生命值	显示为"HP"
get_phyAtk	void	int，英雄基础物理攻击力	基础能力随英雄而定，与装备完全无关
get_magAtk	void	int，英雄基础法术攻击力	
get_phyDef	void	int，英雄基础物理防御力	

get_magDef	void	int，英雄基础法术防御力	
get_equipcnt	void	int，英雄已购买的装备个数	显示为“Equipment”
get_money	void	int，英雄拥有的金钱数	显示为“MONEY”
get_equipment	int num	Equipment*，第 num 个装备的指针	0≤num≤2
isalive	void	bool，判断英雄是否活着	true 当且仅当 HP > 0
reset	共 4 个	void	重置英雄状态
装备操作			
buy_equipment	Equipment*	bool，金钱钱要足够够且现有装备数量小于 3 才返回 true	无论购买成功或失败，都打印反馈信息
remove_equipment[1]	int num	bool，num 有效则返回 true	同上，打印反馈信息
remove_equipment[3]	Equipment*	bool，通过指针删除装备	同上
sell_equipment	int num	bool，num 有效则返回 true	得到装备价格的 60%
syn_equipment[5]	int x, int y	bool，合成成功则返回 true	合成攻击型装备和防御装备，额外花费 15%金钱
战斗操作			
get_real_phyAtk	void	int，总物理攻击力	get_real_* 系列成员函数获取英雄的总能力，包括基础能力和装备加成
get_real_magAtk	void	int，总法术攻击力	
get_real_phyDef	void	int，总物理防御力	
get_real_magDef	void	int，总法术防御力	
get_crit_chance	void	int，暴击几率	
attack	Hero&	int，实际给对方造成的伤害	
crit[8]	Hero&	int，实际给对方造成的伤害	attack 时概率触发
normal_attack[7]	Hero&	int，实际给对方造成的伤害	
recover[9]	void	bool，回血成功则返回 true	见下方备注
damage_taken	int n	int，实际减少的生命值	

备注:

1. 表中灰色的函数为 private 的，只允许被类中其他函数调用。
2. 出于数据合理性考虑，英雄初始生命值设置为 100，物理、法术的攻击和防御力均为 10，初始金钱数为 800。
3. remove_equipment 与 sell_equipment 不同。前者为 private 的，仅供其它成员函数调用；后者是 public 的，可以由玩家主动调用。remove_equipment 在程序中的作用是供 sell_equipment 和 syn_equipment 调用。
4. Hero 类中**重载+=运算符**，myHero += knife 效果等同于 myhero.buy_equipment(knife)；同理，**重载-=运算符**，myHero -= 0 效果等同于 myhero.sell_equipment(0)。
5. syn_equipment 操作为一个英雄主动地合成已经在自己装备栏中的 2 个装备，目的是减少装备栏中的装备数量以供购买新装备。这 2 个装备必须是一个攻击型装备（AttackEquipment）和一个防御型装备（DefenceEquipment），**不能合成两个同类型的装备。合成装备需要额外花费两个子装备价格之和的 15%**。
   syn_equipment 函数是英雄的主动操作，实际运行时可由玩家决定。该成员函数的作用与 main.cpp 中不同，后者是直接构建一个既有攻击特性又有防御特性的“综合“型装备。
   合成装备不允许再进行合成。
6. 英雄使用 vector 容器储存装备(Equipment*)，因此不需要“装备数量“这一成员变量，因为 vector 的增加、删除操作会自动改变其 size()的大小。英雄类中仍保留 get_equipcnt 函数方便使用，此函数直接返回 vector 的 size()。
7. **伤害计算方法**:

$$\text{免伤比} = \left(\frac{\textit{敌人防御力}}{\textit{敌人防御力} + C_{IMMUNE}}\right) \times 100\%$$

$$\text{实际伤害} = (1 - \text{敌人免伤比}) \times \text{英雄攻击力}$$

C_IMMUNE 为常数，默认为 24.0。

8. 每次 attack 对方的时候，有一定几率触发暴击，暴击可以造成 1.5 倍伤害。暴击几率的计算公式如下：暴击几率 = 剩余金钱数 / 2000，最多不超过 100%。
9. 回血。英雄可以主动选择回血而不攻击，回血花费 200 金钱，恢复 15 生命值，生命值最多不超过 100。如果生命值已经为 100，那么除了花钱并浪费一回合，无任何效果。

- **战场（Battleground）类**

包含文件：Battleground.h(.cpp)。

数据成员	描述	备注
Hero _player	玩家英雄	由玩家输入确定
Hero _enemy	敌人英雄	系统随机指定
int _game	游戏局数	
int _round	该局中的回合数	
int _score	玩家总得分	
vector<Equipment*> _inventory	装备库存（所有可购买的）	

成员函数	参数	返回值类型及含义	备注
game_loop	void	void，游戏开始后的多局循环	每局结束后询问是否再开一盘
round_loop	void	bool，true 代表敌人死亡，false 代表玩家死亡	一方英雄死亡，即一局结束
get_game	void	int，获取局数	
get_round	void	int，获取局内的当前回合数	
get_score	void	int，获取总得分	
get_inventory	int num	Equipment*，获取第 num 个装备	
print_inventory	void	void，显示装备库存	

- **简易 UI（非类）**

包含文件：ui.h(.cpp)。

函数	描述	备注
split_line	分割线	100 个等号
print_introduction	刚进入游戏的介绍画面	字面意思，见注释
print_help	游戏说明	
print_bye	结束画面	

- **常量（非类）**

文件：macro.h。

定义了各种常量，宏名均为 C_开头。详见栏目（四）。除此之外，还有若干以 SET_开头的宏函

数，如 SET_CYAN、SET_WHITE 等，是用来控制台显示颜色的。

```
/* Those preprocessor directives use Windwos API to print colorful characters */
#define SET_CYAN SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),FOREGROUND_INTENSITY|FOREGROUND_GREEN|FOREGROUND_BLUE)
#define SET_WHITE SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),FOREGROUND_INTENSITY|FOREGROUND_RED|FOREGROUND_GREEN | FOREGROUND_BLUE)
#define SET_RED SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),FOREGROUND_INTENSITY|FOREGROUND_RED)
#define SET_GREEN SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),FOREGROUND_INTENSITY|FOREGROUND_GREEN)
#define SET_YELLOW SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),FOREGROUND_INTENSITY|FOREGROUND_RED|FOREGROUND_GREEN)
```

四、 本文档涉及到的常量及数据计算方法

本板块的内容大都在上文都出现过，此处作为汇总方便查阅。下文中 C 开头的变量为可变常量，均以宏的形式定义在 macro.h 中。

- **伤害计算**

$$\text{实际伤害} = \left(1 - \frac{\textit{敌人防御力}}{\textit{敌人防御力} + C_{IMMUNE}}\right) \times \textit{攻击力}$$

C_IMMUNE 为免伤比常数，为 **24.0**.。

- **暴击率计算**

$$\text{暴击率} = \frac{\textit{金钱数}}{C_{CRIT\_MONEY}} \times 100\%$$

C_CRIT_MONEY 为暴击率上限金钱数，为 **2000**。如果英雄金钱数≥2000，那么暴击率为 100%。

- **暴击加成计算**

$$\text{暴击伤害} = \text{普通攻击伤害} \times \mathrm{C_{CRIT}}$$

C_CRIT 为暴击加成倍率，为 **1.5**。

- **攻击敌人获得金钱计算**

$$\text{收入} = \text{造成的伤害} \times \mathrm{C_{RECEIVE}}$$

C_RECEIVE 为收入系数，为 **5**。如造成对方 10 伤害，则自己收入 50 金钱。

- **合成装备**

$$\text{合成额外花费} = (\text{攻击型装备价格} + \text{防御型装备价格}) \times \mathbf{0.15}$$

- **卖出装备收入计算**

$$\text{收入} = \text{装备原价格} \times \mathrm{C_{SELL}}$$

C_SELL 为卖出装备价格系数，为 **0.6**。如果装备是合成装备，则有：

$$\text{合成装备原价格} = \text{攻击型装备价格} + \text{防御型装备价格}$$

不包含合成装备额外花费的那 15%的金钱。

- **得分计算**

1. 每次造成敌人伤害，**造成了多少伤害就加多少分**；
2. 每回合得分**加 100**；
3. 游戏结束时，得分加 **200×局数**。

五、 附录

- **测试代码**

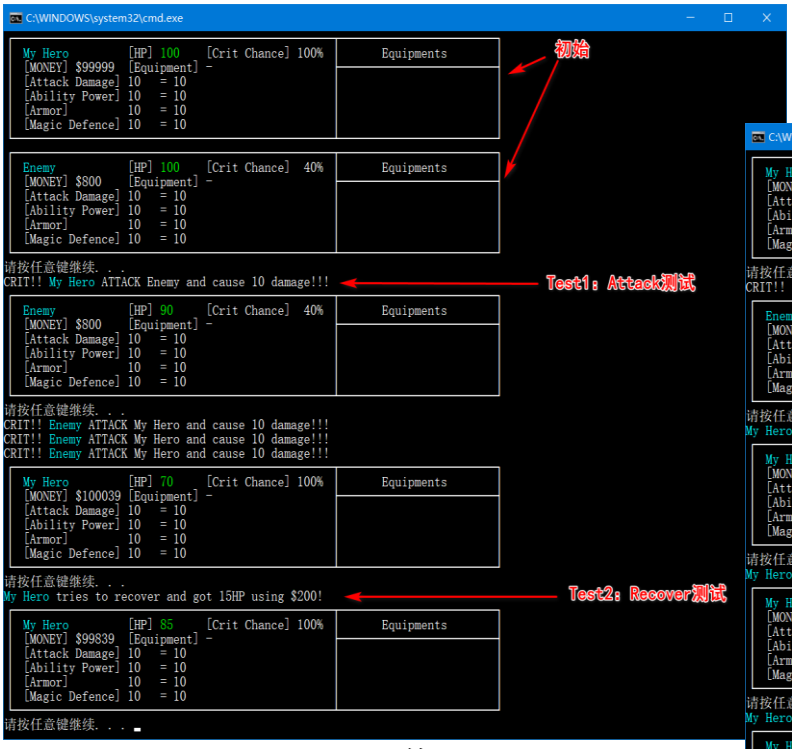
为了方便调试，专门设计测试模式。

`game_test` 函数用于手动测试代码，该函数位于 `main.cpp` 中。当通过`-t` 参数运行程序时将进入测试模式，且不会显示任何 UI 界面。注意：`game_test` 与 `game_main` 函数完全并列，在 `main` 函数中位于同一个 `if-else` 中。

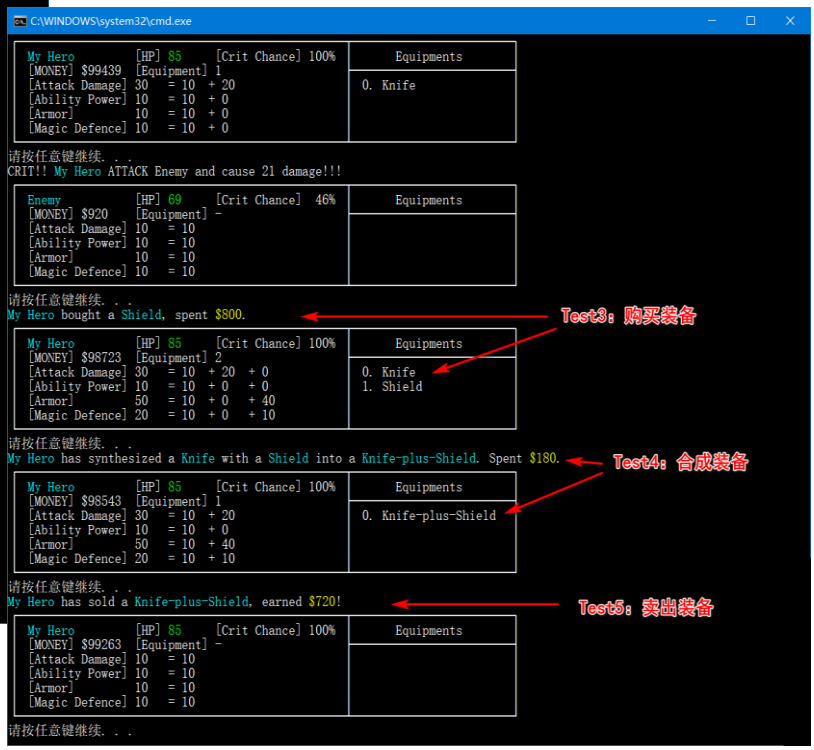
在项目根目录下有 `testmode.bat`，可以快速进入测试模式。

```
void game_test() {
    // Hero myhero(name, money, health, phyAtk, magAtk, phyDef, magDef)
    Hero myhero("My Hero", 99999, 100, 10, 10, 10, 10);
    Hero enemy("Enemy");
    cout << myhero << endl << enemy << endl;
    system("pause");

    // Test 1: Attack test.
    myhero.attack(enemy);
    cout << enemy << endl;
    system("pause");

    // Test 2: Recover test.
    enemy.attack(myhero);
    enemy.attack(myhero);
    enemy.attack(myhero);
    cout << myhero << endl;
    system("pause");

    myhero.recover();
    cout << myhero << endl;
    system("pause");

    // This are two demo equipment.
    AttackEquipment* knife = new AttackEquipment("Knife", 400, 20, 0);
    DefenceEquipment* shield = new DefenceEquipment("Shield", 800, 40, 10);
    // Test 3: Buy a equipment in two ways (function and operator overloading).
    myhero.buy_equipment(knife);
    cout << myhero << endl;
    system("pause");

    myhero.attack(enemy);
    cout << enemy << endl;
    system("pause");

    myhero += shield;
    cout << myhero << endl;
    system("pause");

    // Test 4: Synthesize two equipments.
    myhero.syn_equipment(0, 1);
    cout << myhero << endl;
    system("pause");

    // Test 5: Sell equipment.
    myhero.sell_equipment(0);
    cout << myhero << endl;
    system("pause");

    cout << "Test is over." << endl;
    system("pause");
}
```

测试模式运行结果：



第一屏



第二屏

- **预置的装备库存**

类别	装备名称	价格	Attack Damage	Ability Power	Armor	Magic Defence
Attack Equipment 攻击型装备	Knife	400	20	0	-	-
	Sword	650	45	25	-	-
	Poison	1000	30	50	-	-
	AK47	1400	90	10		
Defence Equipment 防御型装备	Vest	500	-	-	30	2
	Shield	800	-	-	40	10
	Immunity	1200	-	-	50	50
	Magnetic_field	1500	-	-	80	20
Synthesis Equipment 综合型装备	尚未设计					

后期版本可设计更多装备。

- **游戏帮助（print_help）**

```
In this game you will play a hero. You will participate in multiple games if you
can keep alive. In each game, you will encounter an enemy hero, and you will conduct
multiple rounds of fighting. At the beginning, both heroes have 100 HP. In each
round, you and the enemy can each perform one operation. These operations include
"attack", "recover", "buy equipment," "sell equipment," etc.
In this battle, if you defeat the enemy, then the game ends. You can choose to
start a new game again and accumulate points. Of course you can also exit the game.
However, if your hero dies, the game ends immediately and you will see your final
score.
Enjoy it!
```

- **更新日志**

v1.0.3 (2018.06.09)
修复了 Hero 和 Battleground 中析构函数导致内存错误的问题。

v1.0.2 (2018.06.07)
重新封装了 SynthesisEquipment 类，改用 set 方法。

v1.0.1 (2018.06.03)
修复了购买装备时不合法输入导致程序崩溃的问题。

已知 Bug：Sleep 函数和_getch 函数导致的流缓冲问题。
描述：程序在 Sleep 的时候，按键操作仍被记录，程序恢复执行时这些输入直接被_getch()读取，导致多于操作。例如，每次轮到玩家操作时，如果快速连续按很多次 A，那么后续几个回合将自动进行。