

## part 1 规范性修改

### 修改1

#### 修改前

```
// 检查有没有被龙卷风伤到的单位
bool checkattack(Field& field, Unit* u, int rch, int cch) {
    if (!(field.inBounds(u->getRow()+rch, u->getCol()+cch)) || (field.getTerrain(u->getRow()+rch, u->getCol()+cch) != PLAIN) || !field.getUnit(u->getRow()+rch, u->getCol()+cch)) {
        return false;
    }
    return true;
}
```

#### 修改后

```
// 检查有没有被龙卷风伤到的单位
bool checkattack(Field& field, Unit* u, int rch, int cch) {
    if (!(field.inBounds(u->getRow()+rch, u->getCol()+cch))
        || (field.getTerrain(u->getRow()+rch, u->getCol()+cch) != PLAIN)
        || !field.getUnit(u->getRow()+rch, u->getCol()+cch))
    {
        return false;
    }
    return true;
}
```

#### 说明：

修改前if条件的代码太长，长行拆分不太合理，故进行修改，使一行表达一个条件。

### 修改2

#### 修改前

```
void Unit::oppo() {
    ud = -ud;
    lr = -lr;
}
```



#### 修改后

```
void Unit::turnBack() {
    upOrDown = -upOrDown;
    leftOrRight = -leftOrRight;
}
```

#### 说明：

修改前函数名称不能很好反映函数的功能，以及变量命名不够直观，不可拼读，故进行修改，提高其可读性。

# 修改3

修改前	
	<pre>Field* f = loadMap(cin, units);  if (f == NULL) {     cout &lt;&lt; "Failed to load map!" &lt;&lt; endl;     return 0; }</pre>
修改后	
	<pre>vector&lt;Unit&gt; units; Field* f = loadMap(cin, units);  if (f == nullptr) {     cout &lt;&lt; "Failed to load map!" &lt;&lt; endl;     return 0; }</pre>

说明：  
修改前误把指针变量用“==”与 NULL比较，故进行修改，将NULL改为nullptr。

## part 2 对我最有帮助的规范性建议

### 建议1

重要	是否用隐含错误或冗余的方式写if语句？例如
	(1) 将布尔变量直接与 TRUE、FALSE 或者 1、0 进行比较。
	(2) 将浮点变量用 “==” 或 “!=” 与任何数字比较。
	(3) 将指针变量用 “==” 或 “!=” 与 NULL比较。
	(4) 把 “==” 误写成 “=”。

经常把==误写成=，然后出bug了，自己单看还看不出来。

### 建议2

	注释是否清晰并且必要？
重要	注释是否有错误或者可能导致误解？

有时候刚开始写的时候懒得写注释，写到后面回来看，已经忘记这个函数的作用和实现方法了。

### 建议3

重要	程序中是否出现相同的局部变量和全部变量？
----	----------------------

习惯性在for循环里使用变量int i，但如果for循环里还有一个for循环，有时候就会不小心用同一个变量名表示，导致自己的混乱

## 疑问

代码自查表里说：

是否用隐含错误或冗余的方式写if语句？例如  
(2) 将浮点变量用"=="或"!="与任何数字比较。

那么我该如何判断两个浮点数是否相等呢？