魔兽世界3文档

xxxxxxxxxx xxx

# 说明

在对应的文件夹中有类图classdiagram.svg，使用浏览器打开即可查看。

# 需求变化分析

## 需求变化总结

一共有N + 2个城市（司令部也算城市），发挥作用了，武士可以走到城市发生战斗，城市分奇偶决定进攻方。

武士的生命值减少到0，算战死，需要从场上消失。

武器有攻击力了，有的武器有特殊的效果，有的武器有耐久度。

战斗前对武器进行排序，双方战斗中循环轮流使用武器，直到武器无变化或一方战死。

战斗后胜利方会缴获武器，有上限。

有些武士也会有特殊的行为，dragon会欢呼，ninja不会被bomb炸到，iceman前进扣血，lion会逃跑。wolf开战前会抢武器。

武士报告武器需要报告所有的武器。

## 理清一部分不需要考虑的细节

司令部虽然也被当做城市，但是由于武士生产发生在武士前进之前，武士前进发生在武士战斗之前，所以武士永远不会在司令部内发生战斗，不用额外考虑。

lion逃跑发生在前进之前，所以不需要考虑lion在敌方司令部逃跑的特殊情况，因为lion一旦进入敌方司令部，游戏就结束了，不会再进入到一个小时之后的下一次逃跑环节，所以这个需求暂时不用考虑。

一个城市只能容纳红方蓝方士兵各一个的需求也不需要考虑，因为司令部也算城市，武士生产出来就放在司令部中，到时间就往前前进，碰面就发生战斗，无论战斗结果如何，武士都会继续往前走，综合来看，每一个时刻每个城市内最多只有红方蓝方各一个武士，司令部在每个时刻最多有一个武士，也就是生产武士的那一刻，或者敌方武士进入的那一刻，所以不用多余的考虑。

关于武士生命值变为负数时当作变为0处理，其实不用处理，因为武士死亡后不会再打印任何关于死亡武士血量的信息。

# 需求实现

由于需求是逐步变更的，这里主要说明针对变化的部分的做法，关于魔兽世界2的部分不再赘述。

## 城市、司令部

首先城市需要容纳武士，司令部也可以放置武士，所以首先让司令部继承城市，由分析可知，司令部的武士列表不需要了，直接移除。然后在城市中添加一些成员属性，包括城市的编号，指向前一个城市的引用，指向后一个城市的引用，指向在这个城市的红方武士引用redWarrior，指向在这个城市的蓝方武士的引用blueWarrior。由于需要得知城市的编号信息，相比于之前的魔兽世界，在GameMain类中增加了给城市编号赋值的操作，同时在构造司令部和城市时增加了城市与城市之间连接构造双向链表的操作。

司令部建造武士时，根据自己的红蓝选边，决定是调用this.setRedWarrior还是this.setBlueWarrior。

## 武士死亡

武士需要死亡，所以城市类新增了一个detach方法，在这个这个方法中根据调用者的选边(getSide获取)，决定把redWarrior设置为null还是把blueWarrior设置为null。

然后在武士类增加一个成员变量city，指向当前自己所在的城市。接着增加一个die方法，在这个方法中，调用city的detach方法，将自己从城市中解绑脱离，然后调用EventSystem的unregister方法反注册自己，完成善后工作。

## 武士武器报告

在Logger类中使用一个Map统计所有的某个武士所有武器的数量，然后依次输出即可。

## 武器

### 武器使用

接着说一下武器的问题，由于增加了攻击力，在此先选择将攻击力硬编码在代码中（考虑后面的需求变化，这里选择硬编码。这点将在魔兽世界5改进，这里先选择最简单的实现，怎么简单怎么来）。

武器需要被使用，所以武器基类需要写一个抽象的接口use()，use中传递两个参数，分别是攻击者和被攻击者，达到武器和使用者的解耦。这个接口返回这把武器在本次使用完成后能否被再次使用。

因为战斗需要战斗到武器状态无变化，所以武器需要提供一个抽象接口canDamage判断这把武器还能不能造成伤害。

### 武器排序

由于要对武器进行排序，而排序有两种排法，一种是战斗前的排序，也就是按照武器编号先排，然后按照耐久度排序。一种是抢武器时的排序，按照武器编号先排，然后按照耐久度后排。因为不是所有的武器都有耐久度这一个属性，所以在武器基类中写两个比较方法compareTo（@Override）和robCompare，这两个方法在基类中的实现都一样，只单纯的比较两种武器的编号，把个性化的行为延迟到子类中实现。

### 武器抢夺

武士基类中有一个robWeapon方法，可以返回一个列表，里面装着抢到的武器。这个方法首先获得被抢夺方的武器列表，如果发现没有武器就返回空列表，否则就使用Collections.sort(Weapon::robCompare)来对武器进行排序，之后根据自己拥有的武器情况计算最多能抢夺几把武器，然后抢夺尽可能多的武器（最多10个），并返回这些抢夺获得的武器，使用stream.limit方法控制数量。

武士基类中还有一个robWarrior方法，传入被抢夺的对象。这个方法首先调用robWeapon方法获得抢到的武器，然后将其添加到自己的武器列表中，并从被抢夺的武士的武器列表中移除这些武器。

### 关于Arrow

弓箭用两次就没了，所以arrow有一个durable属性，表示这个弓箭的剩下使用次数。

因为即使弓箭无法造成伤害也要使用，所以canDamage永远返回真。

很自然，use方法就是使被攻击方的生命值减少攻击方的攻击力的30%，然后durable属性减少1，返回durable大于0即可。

然后根据题目要求，需要重写compareTo和robCompare方法，在compareTo中第一步首先调用基类的compareTo方法以满足里氏替换原则，接着在基类的比较返回值为0的情况下，返回两把弓箭的耐久度的比较，用过的排在前面。至于robCompare，同理，刚开始要调用基类的robCompare方法，在基类比较方法返回值为0时返回两把弓箭耐久度的比较，用过的排在后面。

### 关于Bomb

Bomb是一次性消耗物品，毫无疑问canDamage返回真。

由于Bomb没有自己的特殊的属性，所以不用重写compareTo和robCompare方法。

在use方法中，首先让被攻击者的生命减去40%攻击方的攻击力。接着还需要特判一下攻击者是不是Ninja，如果不是就让攻击者的生命减去刚刚造成的伤害的一半的伤害。由于用一次就没了，所以use方法永远返回假。

### 关于Sword

Sword是永久物品，所以canDamage返回在当前攻击者的攻击力下，Sword能造成的伤害是否大于0。

由于Sword没有自己的特殊的属性，所以不用重写compareTo和robCompare方法。

在use方法中，让被攻击者的生命减去20%攻击方的攻击力。Sword耐久度是永久，所以use方法永远返回真。

## 战斗

### 战斗结束判断

战斗系统需要判断武士能否继续战斗下去，所以在武士基类中提供一个方法canDamage，首先判断这个武士是否已经死亡，如果已经死亡就直接返回假，武士没有武器也直接返回假，否则就遍历这个武士拥有的所有武器，依次调用武器的canDamage方法，如果有任意一把武器返回真，就代表这个武士的状态还没有稳定，战斗还不能结束，返回真。如果所有武器的canDamage方法都返回假，就返回假。

### 战斗逻辑

接着说一下城市类的战斗逻辑实现，首先某个城市在战斗事件发生时，需要同时有蓝方和红方的武士。

战斗开始前，需要调用双方武士的的sortWeapon方法（该方法使用Collections.sort来对武士的武器列表进行排序）对各自的武器排好序。

紧接着根据城市的编号奇偶决定哪方先出手，把这个信息记录在一个叫isRedTurn的变量中。

然后按顺序双方循环依次出手。具体的做法如下：

循环的结束条件是双方都还活着（isAlive）并且任意一方的canDamage方法返回真（换句话说，任意一方没有稳定），循环继续，轮到某一方时只要它还能发起攻击（也就是canDamage方法返回真）就调用这个武士的attack方法，并传入受害者，如果不能发起攻击（也就是canDamage返回假）就直接将isRedTurn取反，进入下一轮循环。

武士的attack方法具体来说就是将从武器列表中拿出第一把武器，然后调用这把武器的use方法，如果返回为真，这将这把武器重新添加到武器列表的末尾。

在attack方法执行完成后，将isRedTurn取反，进入下一轮循环。

当循环的条件不满足，退出循环后，战斗结束，进入战斗结算环节。

### 战斗结算

如果双方均存活，调用Logger输出双方存活的信息。

如果双方均死亡，调用Logger输出双方死亡的信息，然后调用两个武士的die方法，做好武士的善后工作。

如果一方胜利，则调用胜利方的robWarrior方法抢夺失败方的武器，然后调用失败方的die方法做好善后的工作。

## 武士的特殊行为

### 龙欢呼

在基类中写一个getWinMessage方法，返回一个空实现。Dragon类重写这个方法，返回龙欢呼的信息，在Logger类打印胜利消息时，将会调用武士类的getWinMessage方法，其他武士返回的是空，而龙返回的是龙的欢呼信息。

### 忍者不会被炸弹炸到

这一点在上面介绍Bomb这个武器的部分已经讲过了，这里不再赘述。

### 冰人前进扣血

冰人在接收到前进的事件后，血量扣除10%，这个逻辑在Iceman类的acceptEvent中实现。

### 狮子逃跑

狮子在接收到前进的事件后忠诚度减少一定的值，当接受到狮子逃跑事件后判断当前的忠诚度是否已经小于等于0，如果是的话就调用Logger输出逃跑的信息，接着调用自己的die方法善后处理。

### 狼抢武器

当狼接受到抢劫武器事件后，首先判断所在的城市内有没有对方的武士，接着判断对方是不是狼。如果是狼的话就不抢，否则就调用robWeapon方法按照战胜抢武器的标准抢劫到一堆武器，然后从中挑出编号最小的，比如00112222，就从中挑出00这两把武器，因为列表是排好序的，所以只需要stream.filter方法筛出编号与第一把武器相同的武器的即可。接着从敌方的武器列表中移除处理完成的武器，并将这些新获得的武器加入到自己的武器列表中，最后调用Logger类报告抢劫的情况，抢劫的过程至此完成了。

## 事件系统

这个版本的事件系统做了一些更新，移除了原来的根据字符串找对象通知事件的notify方法，取而代之的是使用Class对象通知的notify方法，也就是根据类来通知具体的对象，比如我想要通知所有的城市某个事件就写notify(City.class, event)即可。这个方法内部使用循环的方法按照注册到消息系统的顺序依次检查并通知，所谓的检查也就是看xxx.class == yyy.class，确保是需要通知的对象，按顺序循环通知是为了保证输出的顺序和题目的要求一致。

同时还提供一个全局通知的notify方法，也就是通知所有的对象某个事件。同样也是使用循环的方法遍历所有对象挨个通知。

## 时钟系统

因为增加了很多事件，所以需要修改时钟系统，将原来的Clock类代码删除，重新设计。

在其中增加一个成员变量elapse代表当前走过的时间，然后再增加一个Queue<Event>用于按顺序存储系统有可能发生的所有事件。

修改原来的start方法，首先一开始就将自己注册到事件系统中（这么做的原因后面介绍），然后将所有可能发生的事件new出来，按顺序入队列，最后进行一个循环，循环终止条件是elapse已经超过模拟时间。循环体中每次从队列头获得一个事件的引用，调用handle方法后检查这个事件是否已经被处理，如果返回为假就代表还没到这个事件发生的时间，直接将当前的elapse += 5然后继续下一轮循环，如果返回为真代表这个事件已经被处理了，就将这个事件从队头取出，重新添加到队尾，elapse += 5，开启下一轮循环。

在时钟系统中还需要重写acceptEvent方法，如果事件系统给它传递一个EndEvent，代表游戏已经结束，直接将elapse设置为10000，这样在下一轮循环中循环条件就会不满足，从而结束循环了。

## 事件

所有的事件都继承自一个基类Event，基类中记录着这个事件发生的时和分，hour、minute。相对于魔兽世界2，需要修改Event类的toString方法，使其增加输出分的部分。

Event中还有一个handle方法，传入一个时间，返回这个事件有没有被处理。所有事件的handle方法都会判断现在这个时刻有没有发生这个事件。

由于多了很多需求，所以新增了很多事件，这里一一列举。

### 武士降生事件

很简单，这个事件就是魔兽世界1、2中的TimeEvent，只不过是修改了名字而已。

这个事件的handle方法会根据传入的时间判断是否到时间发生这个事件了，如果没到就放回假，如果到了就根据传入的时间设置hour和minute（后面的事件刚开始都要做这个处理，所以后面不再赘述），并使用时间系统来通知所有的司令部生产武士，把自己作为通知的事件（EventSystem.notify(Headquarter.class, this)）。

司令部接收事件生产武士的过程上面已经说过，这里不再赘述。

### 狮子逃跑事件

处理过程同上一个事件，但是这里通知的是狮子（Lion.class）。

狮子逃跑的处理上面说过，不再赘述。

### 武士前进事件

武士前进需要交换相邻的城市的武士，红武士往东走，蓝武士往西走，所以需要在事件体中设置一个redWaitWarrior代表来自前一个城市的红武士，由于顺序的自西向东遍历城市，所以不用存储蓝武士了。

除此以外还需要有一个成员变量，gameEnd，代表着游戏是否已经结束了。

在handle方法内首先清空将redWairWarrior变量设置为null，然后使用事件系统通知所有的武士，让他们了把自己的自定义行为做完，比如Lion减少忠诚度，Iceman减少生命值。

接着再通知所有的City（司令部也是城市），在城市的acceptEvent方法中，首先将自己的redWarrior与事件中的redWaitWarrior交换，并修改交换后的redWarrior的city指向自己，再打印武士行进的信息。然后将next城市中的蓝武士接过来，覆盖本城市的blueWarrior，修改它的city变量指向自己，打印武士行进的信息。

司令部的处理流程如下：

如果是红方的司令部，也是将自己的redWarrior与事件中的redWaitWarrior交换（交换后redWarrior就是null了），然后把与之连接的城市的蓝武士接过来，如果发现blueWarrior不为null，代表该司令部已被占领，直接打印司令部被占领的信息，同时设置事件中的gameEnd为真。

如果是蓝方的司令部，把事件中的redWaitWarrior直接设置到自己的redWarrior中，然后把自己的blueWarrior设置为null，因为已经送出去了，接着如果发现redWarrior不为null，代表该司令部已被占领，直接打印司令部被占领的信息，同时设置事件中的gameEnd为真。

在事件的handle方法最后，如果发现gameEnd被设置成真，就使用事件系统给Clock发送一个EndEvent，通知时钟系统结束时钟循环（对上面为什么Clock类也是GameObject的子类，也是观察者的解释）。

至此，武士行进事件处理完毕。

### 狼抢夺武器

直接使用事件系统通知所有的狼抢武器即可，抢夺武器的过程上面说过，不再赘述。

这里要注意的是，因为需要从左到右汇报抢夺的顺序，时间系统中只有城市的通知顺序是有序的，所以是先通知城市，城市接收到这个事件后再将其委托给处于该城市中的狼来处理（调用狼的acceptEvent方法）。

### 战斗事件

直接使用事件系统通知所有的城市即可。战斗的过程上面说过，不再赘述。

### 司令部汇报生命元事件

直接使用事件系统通知所有的司令部即可，司令部收到事件后使用Logger打印对应的信息。

### 武士报告武器事件

直接使用事件系统通知所有的城市即可，城市接收到事件后先使用Logger打印该城市的红方武士的武器情况，再使用Logger打印该城市蓝方武士的武器情况。（为了满足题目的打印要求）

# 内存管理问题

这个代码中有部分内容无法被垃圾回收器回收，比如城市之间的循环引用，城市无法收回，模拟时间到达之后由于城市无法被回收，所以导致城市之内的指向武士的引用没有释放，导致部分武士也无法被回收，但是由于量比较少，不会占用过多内存，所以在魔兽世界3中也就不再做过多的后处理，这部分不能被回收的对象的问题将在魔兽世界5中得到完美的解决。