# 技能系统

因为这里mark了完整版，所以即便是之前已经有的内容这里可能也要重复，而且会更详细，会把一些之前不确定的内容尽可能变为可执行的确定内容（但这并不意味着以后完全不修改了= =），但有些内容在该阶段确实不能确定，那么这部分内容会在适当的时候补全。

## 主动技能

所谓主动技能，指直接释放就可以造成具体的效果。对应召唤技能与被动技能。

请对应最新配置表阅读以下内容。

### 编号

* 如果是使用我之前编写的配置表工具与配套代码的话，编号（id）是一定会有的，即便在配置表里没有明确声明该字段。

如果配置表没有明确声明id字段，在代码里会自动生成ulong类型的id，并且从1开始自增。

### 名称

* 技能的名称。

事实上我是考虑过多语言的，但即便如此，在最初的配置表里也应该使用中文来作为标记，否则非常反人类……

为此，我专门准备了一个EasyConfig（简称EC）的类来解决这个问题，EC可以将数据（string）以数组形式存放，也可以按照KEY-VALUE键值对的方式来存放（还可以支持一个key对应多个value）到TXT，这里直接将配置的中文名认为是KEY就行，如果在EC里找到该KEY，那么就取对应的VALUE，否则就是用KEY为显示文本。

### 描述

* 描述技能特性与关键数值。
* 在技能描述里，会用到部分转义字符，也需要对富文本进行支持（其实只会用到颜色控制）。
* 转义字符：使用“[key]”的形式，其中key会根据需求增加，通常是技能宿主的属性字段，也可以是该技能中的某个字段值。

一个通常的用法是：对目标造成[skillDamge]倍相当于你当前生命值（[nhp]）的物理伤害。

其中[skillDamge]是技能中的一个字段（假设存在），[nhp]是宿主当前生命。

召唤物是可以成为宿主的。

### 目标类型

* targetType，TargetType类型的参数。
* 它的设计初衷，是决定技能的释放方式，在游戏中可能会有以下几种技能释放方式：
* 需要玩家指定1个可见怪物；
* 指定1个格子（无视格子是否翻开，无视格子上是否有物体，不包含障碍）；
* 指定1个空格子（没有物体，其实这样它就必须是翻开的状态，不包含障碍）；
* 指定1个未翻开的格子（Near状态，即邻接玩家的格子同样算未翻开，不包含障碍）；
* 指定1个已经翻开的格子（不包含障碍）；
* 指定1个障碍（不可通过的格子）；

FindTarget类是用来专门解决目标寻找问题的， FindTarget将自己作为消息参数发送出去（原谅我已经不知道该怎么给消息起名了，话说我觉得中文还是蛮好用的…），接收该消息的Block会自行判断自己是否符合筛选条件，并把自己加入FindTarget的目标列表。

* 以及，不需要玩家指定目标（比如全体攻击技能，或者给自己释放BUFF，或者随机目标）。

实际情况中，还需要考虑到技能技能的射程。



* 综上，我们游戏中的技能不会出现让玩家选择1个以上的目标（通常这种会变成随机目标），要嘛选1个目标，要嘛1个都不用选。

### 目标限制

注意，配置表有修改

* targetLimit，TargetLimit类型的1维数组（枚举信息在<技能.xlasm>[枚举参数]中）。
* 它其实是一个过滤器，比如某些技能不能对某类技能释放，某些技能不能对某个强度的怪物释放。
* 之后根据使用场景进行增删。



如果这里为空，其实程序里读取到的是None

### 多目标

* multipleType，枚举类型的参数。
* 现在支持内容如下（某些内容需要配合targetArg）



### 目标参数

* targetArg，二维float数组。
* 第一维（第一次分割）表示对N个目标产生攻击，第二维表示分段伤害。
* 第一维目前只会在multipleType为Random时起效，multipleType为其他值时，只会取第一个元素。
* 第二维表示这次攻击伤害被分成了几段。

我并不希望一个很华丽演出的效果打了几分钟，结果过程中目标根本不掉血，只有最后一下暴了一个数值…或者每次攻击都只能发射一个炮弹？所以这个参数的作用就是把一个本来伤害100的攻击，分成了多次进行。

如果飞行特效是非激光类，那么有多少次伤害，就会实际产生多少次飞行道具（炮弹？），如果是激光类，表示本次激光造成N次伤害。但是他们不论有多少次伤害，总量还是100，所以这个参数叫伤害权重，如果是1;3这种形式，表示会产生2次伤害，第一次造成25，第二次造成75.

关于SuperArray：

该字段，读取到的是SuperArray<float>这种类型的对象。以配置内容“1;2;3|4;5”为例。你可以使用superArray.Count()获取第一次分割的长度，结果是2（superArray.Count(0)得到3，它表示第一次分割第0号元素的长度）。要获取某个元素可以使用superArray[1,1]，结果是5。注意，你不能使用superArray[0]直接获取值（除非它是个一维的参数），但是可以用superArray.ToArray(0)或者superArray.ToList(0)来将第一维的0号元素转换成数组。如果直接对superArray进行foreach遍历，它会依次返回1,2,3,4,5.

### 飞行特效

* effect\_fly，一维string数组。
* 字符串就是prefab的名字，原则上会把飞行特效做成一个个prefab，在prefab上绑定脚本以实现飞行特效自己的特性（比如有些飞直线，有得飞曲线），是飞行道具还是激光。
* 如果元素大于1个，每次就随机挑一个。（激光就只取第一个）

### 命中特效

* effect\_hit，一维string数组。
* 和飞行特效一样，它也是一个prefab的名字，它会在对目标造成伤害时，在目标身上播放。
* 同样，如果有多个元素，就每次随机挑一个。

### 射程

* carry，一维int数组。
* 它有且仅有2个元素，第一个元素表示最短距离，第二个表示最大距离，目标必须在这个范围内，射程才能判断成功。

### 给自己附加状态

待定

### 给敌人附加状态

待定

### 直接伤害

* damage，sting类型的参数。
* 这里放的是一组四则混合运算（支持加减乘除，括号）的公式，其计算结果就是最终的直接伤害，所以我会将攻防计算全放在这里，在功能层面只需要提供攻防双方的属性，无需关心计算过程。

举例（这是我上一个DIY项目使用的方案）：



其中，冒号前表示这次伤害的类型（M为magic），冒号之后是四则混合运算，其中的各种参数是事前约定好的属性表示方式（包含技能释放者与技能命中者）。

解析四则混合运算最麻烦的地方是需要先将公式转换成逆波兰式，好在这部分功能我已经写好了，使用SuperTool.ToRpn函数，可以将一个合法的四则混合运算字符串转换成逆波兰式，比如上面那个转换后为，之后就好处理了，稍微google一下算法，后面的很容易（每个游戏的属性方案不一样，所以第二步计算不能照搬之前项目的解析方案）。

* 还是要等属性确定才能做这部分。

### 属性改变

* pro\_change，一维string数组。
* 最通常是用来恢复生命值，但也有一些很特殊的用法，比如以下技能



其中的T是临时存放属性。

* 同样要等属性确定才能做这部分。

### 消耗时间

* costTime，float类型的参数。
* 使用一个技能需要消耗的时间。

### 能耗

* usePower，float类型的参数。
* 使用一个技能需要消耗的能量，注意这只在该技能属于召唤物，而且召唤的行为模式是复制攻击时才会用到。