技能系统解决方案

------“技能实现基于数据改变”版本

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 作者 | 内容 | 备注 |
|  |  | 初始文档 | 技能系统的整体设计 |
|  |  |  |  |

# 概述

技能系统作为服务器端基础功能，将与属性、战斗、装备四大系统为角色战斗提供全部逻辑功能。

整个战斗逻辑中，技能系统只为角色提供技能效果的逻辑，即，改变角色属性，添加Effect和Buff/Debuff，具体的数值计算将由独立的战斗系统提供。所以，本文档只分析技能系统的逻辑处理，不涉及任何伤害计算。

“基于参数配置”，概括而言，就是将技能配置（条件、效果等）以参数的形式统一提供，同时，技能的执行结果也是大量属性的变化，而不是直接在技能执行时直接执行具体逻辑。使用此方式的目的是，便于与其它系统（属性、战斗、装备）的对接。同时，也是数据驱动的一种典型应用。

# 基础框架

## 系统结构

系统结构，指技能系统将包含的独立模块，包括：

* 配置数据（技能表格、效果表格、特效表格）
* 技能执行逻辑（技能管理、技能执行接口）
* 技能效果处理逻辑（效果执行单元）

 简而言之，技能本质上是，**提取**“技能配置数据”，**执行**“技能执行逻辑”，**得到**“技能效果”，循环“技能效果”，**修改**目标角色属性或调用相应API。

特别注意，**技能效果须独立于技能执行逻辑**，甚至可以在服务器时将其设计成独立模块。

## 技能结构

技能结构，指具体技能构成，包括：

技能数据（技能实体数据、技能单元数据）

执行逻辑

技能效果

# 配置数据

配置数据，策划使用excel编辑，转换成xml形式，供程序使用。

## 技能实体数据

技能外围配置数据，如等级、条件、描述等。

## 技能单元数据

技能的内部数据，如目标类型、直接效果、动画相关等。

## 技能效果数据

技能的Effect效果，如数值变化、技能动作等。

### 技能效果数据-静态效果

修改目标静态属性

### 技能效果数据-动态效果

修改目标动态属性

### 技能效果数据-触发效果

定义触发效果

# 技能执行逻辑

技能逻辑只定义技能执行时的一些逻辑接口和标准实现。如：

发动技能API

开始蓄力

完成蓄力

激活效果

执行效果

完成效果

结束技能

取消技能

中断技能

特别注意，**标准上，所有技能只需一份技能执行逻辑，绝对不允许为每个技能独立编写执行逻辑。但也允许通过覆盖某些接口的方式，为某些极特殊的技能定制部分额外逻辑。这也是该系统区别于“基于API的技能系统”的最大区别。**

# 效果处理逻辑

效果处理逻辑，即执行技能绑定效果，。

建议，**如果是即时效果，将在技能处理逻辑中直接执行效果处理（如直接伤害）；如果是持续效果，将在多次执行效果处理逻辑（如Buff/Debuff）。**

效果处理逻辑，依据之前定义的效果分类，提供对应接口：

开始处理效果

循环执行效果

结束处理效果

修改基础属性

添加额外属性

执行对应API

# 参数配置

以上设计可以看出，技能系统不对针对任何具体技能逻辑，而是只提供一个标准的逻辑结构（技能执行逻辑、效果处理逻辑）。**技能的独特性由技能配置数据的参数变化来提供。**类似一个方程式：

# 疑难解答

## 为什么不为每个技能单独编写处理逻辑？

技能、战斗、装备、属性的解耦需要，如图

 属性系统是游戏的数据核心，用于连接其它相关系统。

技能和装备的最终目的的改变属性，为战斗计算提供数据来源。

如果使用API形式，除难以解耦之外，更致命的是使得API频繁改动，且参数难以准确设定。

## 如何处理特殊技能？

如果预定的参数难以实现，则可通过继承技能逻辑，覆盖部分接口的方式扩展逻辑。但整体而言，技能的接口是不会有变化的。

## 是否存在开发进度的依赖性

使用本系统，开发人员可以在完整的技能设计案出炉之前，实现绝大多数技能逻辑。因为系统并未涉及任何具体技能的设定。而是将深入技能的本质，即数据的改变。

# 备注

# 参考