(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 105677334 A
(43)申请公布日 2016.06.15

(21)申请号 201511026459.X

(22)申请日 2015.12.29

(71)申请人 珠海金山网络游戏科技有限公司
地址 519000 广东省珠海市吉大景山路莲山巷8号金山软件大厦
申请人 成都西山居互动娱乐科技有限公司

(72)发明人 吴海权 陈汉辉 李茂 陈镇秋 易勇军

(74)专利代理机构 广州嘉权专利商标事务所有限公司 44205
代理人 俞梁清

(51)Int.Cl.
G06F 9/44(2006.01)
A63F 13/60(2014.01)

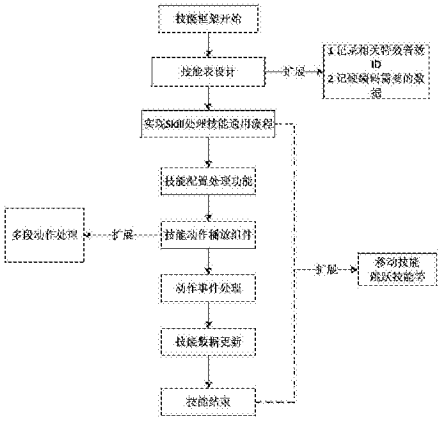
权利要求书2页 说明书9页 附图3页

(54)发明名称

一种通用的游戏状态控制系统和方法

(57)摘要

本发明的技术方案包括一种通用的游戏状态控制系统和方法，其中系统包括：技能准备模块、技能差分模块、技能释放模块和技能结果模块。其中方法包括：对游戏角色释放的技能判定条件使用判定数据表进行存储，然后将数据表封装成应用程序接口；对不同的技能进行检测；在游戏角色释放技能时，通过对差分数据表进行索引获取通用技能特征进行直接引用执行；对技能完成状况进行判断。本发明简化降低系统的复杂度，提高框架的灵活性以及可维护性。



CN 105677334 A

1.一种通用动作游戏技能设计的系统，其特征在于，该系统包括：

技能准备模块，对游戏角色释放的技能判定条件使用判定数据表进行存储，然后将数据表封装成应用程序接口，用于游戏启动时调用应用程序接口对游戏人物技能释放条件进行判断；

技能差分模块，对不同的技能进行检测，对具有相同结构特征的技能进行差分处理，对通用结果部分特征技能进行封装处理，并使用差分表数据表存储；

技能释放模块，对游戏角色释放技能时，通过对差分数据表进行索引获取通用技能特征进行直接引用执行，进而单独处理非通用技能特征部分；

技能结果模块，用于对技能完成状况进行判断，进而执行对释放技能对游戏角色状态造成的影响进行恢复处理。

2.根据权利要求1所述的通用动作游戏技能设计的系统，其特征在于，所述的技能准备模块包括：

判定获取模块，用于对游戏角色的技能信息进行获取，获取技能释放条件并进行调取，对技能及对应的判定条件使用判定数据表进行存储；

判定接口化模块，用于对所述判定获取模块存储的判定数据表进行进一步接口化处理，使判定数据表能够被游戏程序进行调用判定；

判定检测模块，当游戏程序启动时，对判定检测进行初始化，进而对游戏角色技能进行先行判定。

3.根据权利要求1所述的通用动作游戏技能设计的系统，其特征在于，所述的技能差分模块包括：

抽取模块，用于多游戏角色技能具有的共同特征进行抽取，并进行重新封装为基础技能接口，能够接收调用；

差异模块，用于对不具有相同特征的游戏角色技能特征使用对应的函数接口进行处理；

索引模块，对所述抽取模块和差异模块的技能特征均使用差分数据表存储，然后再封装为接口，还用于接收所述技能释放模块的索引。

4.根据权利要求1所述的通用动作游戏技能设计的系统，其特征在于，所述的技能释放模块包括：

对需要使用的技能进行分析，进一步，在释放技能时，通过基础技能接口获取基础技能数据，对差异技能特征使用对应的函数进行处理，完成技能释放还包括对技能释放时的音效、特效、开始与结束时间、帧数据、参数配置进行对应触发和执行，所执行的音效、特效、开始使用数据表进行存储，并允许自定义修改。

5.根据权利要求1所述的通用动作游戏技能设计的系统，其特征在于，所述的技能结果模块包括：

触发模块，使用触发器对技能释放过程进行判定，如果技能释放完成，则触发器相应并执行下一个模块，如果技能触发特殊情况，则调取对应的函数接口处理技能触发的特征情况；

后续处理模块，对完成技能释放的游戏角色状态进行还原。

6.一种通用动作游戏技能设计的方法，其特征在于，该方法包括：

对游戏角色释放的技能判定条件使用判定数据表进行存储，然后将数据表封装成应用程序接口，用于游戏启动时调用应用程序接口对游戏人物技能释放条件进行判断；

对不同的技能进行检测，对具有相同结构特征的技能进行差分处理，对通用结果部分特征技能进行封装处理，并使用差分表数据表存储；

对游戏角色释放技能时，通过对差分数据表进行索引获取通用技能特征进行直接引用执行，进而单独处理非通用技能特征部分；

对技能完成状况进行判断，进而执行对释放技能对游戏角色状态造成的影响进行恢复处理。

7.根据权利要求6所述的通用动作游戏技能设计的方法，该方法还包括：

对游戏角色的技能信息进行获取，获取技能释放条件并进行调取，对技能及对应的判定条件使用判定数据表进行存储；

对需要使用的技能进行分析，进一步，在释放技能时，通过基础技能接口获取基础技能数据，对差异技能特征使用对应的函数进行处理，完成技能释放还包括对技能释放时的音效、特效、开始与结束时间、帧数据、参数配置进行对应触发和执行，所执行的音效、特效、开始使用数据表进行存储，并允许自定义修改；

对所述判定获取模块存储的判定数据表进行进一步接口化处理，使判定数据表能够被游戏程序进行调用判定；

程序启动时，对判定检测进行初始化，进而对游戏角色技能进行先行判定。

8.根据权利要求6所述的通用动作游戏技能设计的方法，该方法还包括：抽取模块，用于多游戏角色技能具有的共同特征进行抽取，并进行重新封装为基础技能接口，能够接收调用；

对不具有相同特征的游戏角色技能特征使用对应的函数接口进行处理；

对所述抽取模块和差异模块的技能特征均使用差分数据表存储，然后再封装为接口，还用于接收所述技能释放模块的索引。

9.根据权利要求6所述的通用动作游戏技能设计的方法，该方法还包括：

对需要使用的技能进行分析，进一步，在释放技能时，通过基础技能接口获取基础技能数据，对差异技能特征使用对应的函数进行处理，完成技能释放还包括对技能释放时的音效、特效、开始与结束时间、帧数据、参数配置进行对应触发和执行，所执行的音效、特效、开始使用数据表进行存储，并允许自定义修改。

10.根据权利要求6所述的通用动作游戏技能设计的方法，该方法还包括：

使用触发器对技能释放过程进行判定，如果技能释放完成，则触发器相应并执行下一个模块，如果技能触发特殊情况，则调取对应的函数接口处理技能触发的特征情况；

对完成技能释放的游戏角色状态进行还原。

一种通用的游戏状态控制系统和方法

技术领域

[0001] 本发明涉及一种通用的游戏界面行为控制系统和方法，属于游戏开发技术领域。

背景技术

[0002] 战斗在游戏中，有着举足轻重的作用。战斗经常也是游戏论坛里讨论最热门的话题之一。战斗好玩、真实，可以增加玩家的游戏体验，感觉更加虚拟的世界里更加地真实生动有趣。很多时候，玩家会因为战斗胜利，而洋洋得意，充满自豪和成就感。也有很多时候，因为战斗失利，从而，转向去升级技能、装备等，想尽办法，甚至不惜重金砸进去提升自己的能力，以期望打败对手或者通关。而技能是战斗重要的组成部分，不过，由于技能系统涉及的内容比较多，包括角色的动作、特效、音效、位移、战斗结果、冷却时间(CD)更新、属性变化等，而每一小项里面，又需要再次，如动作可能为:循环的动作、没有动作放技能、二个动作组合起来放在一个技能里、三段动作组合起来在一个技能里、三段动作里某一段动作是循环的、动作播放到n时间位置时，需要定帧(停卡一段时间)再播放动等。如特效又分为:起手、蓄手、出手、打击、收招等特效都不同。由于技能系统涉及方方面的因素，比较多，因此，技能系统也往往是动作手游较难设计好的一个系统。很多公司在技能系统的实现的时候，往往都是往用“硬编码”(hardcode)的方式。当策划提出设计一个技能的时候，更多的是去接是硬编码。但是这种设计方式有一个弊端，随时技能数量的增加，会发现后面越来越难以增加技能，同时，由于设计与别的模块:特效、角色属性、动作模块、状态模块等耦合度太大，导致修改、增加技能难度非常大，几乎是“牵一发而动全身”，各种bug也莫名其妙地出现，防不胜防。

[0003] 针对上述问题，本文综合分析各个模块的特点和需求，将容易变化的数据使用配表去实现，以增加技能灵活性以及扩展性。同时，将特殊的效果，那些效果需要hardcode去实现的，转移去单独的模块去实现，这样既可以保证技能系统的功能和表现，同时，设计者还可以通过表去调整技能表现的效果。

发明内容

[0004] 针对现有技术问题，本发明提供一种通用的游戏状态控制系统和方法。

[0005] 本发明的技术方案一方面包括一种通用动作游戏技能设计的系统，包括:技能准备模块，对游戏角色释放的技能判定条件使用判定数据表进行存储，然后将数据表封装成应用程序接口，用于游戏启动时调用应用程序接口对游戏人物技能释放条件进行判断;技能差分模块，对不同的技能进行检测，对具有相同结构特征的技能进行差分处理，对通用结果部分特征技能进行封装处理，并使用差分表数据表存储;技能释放模块，对游戏角色释放技能时，通过 对差分数据表进行索引获取通用技能特征进行直接引用执行，进而单独处理非通用技能特征部分;技能结果模块，用于对技能完成状况进行判断，进而执行对释放技能对游戏角色状态造成的影响进行恢复处理。

[0006] 进一步，所述的技能准备模块包括:判定获取模块，用于对游戏角色的技能信息进

行获取，获取技能释放条件并进行调取，对技能及对应的判定条件使用判定数据表进行存储；判定接口化模块，用于对所述判定获取模块存储的判定数据表进行进一步接口化处理，使判定数据表能够被游戏程序进行调用判定；判定检测模块，当游戏程序启动时，对判定检测进行初始化，进而对游戏角色技能进行先行判定。

[0007] 进一步，所述的技能差分模块包括：抽取模块，用于多游戏角色技能具有的共同特征进行抽取，并进行重新封装为基础技能接口，能够接收调用；差异模块，用于对不具有相同特征的游戏角色技能特征使用对应的函数接口进行处理；索引模块，对所述抽取模块和差异模块的技能特征均使用差分数据表存储，然后再封装为接口，还用于接收所述技能释放模块的索引。

[0008] 进一步，所述的技能释放模块包括：对需要使用的技能进行分析，进一步，在释放技能时，通过基础技能接口获取基础技能数据，对差异技能特征使用对应的函数进行处理，完成技能释放还包括对技能释放时的音效、特效、开始与结束时间、帧数据、参数配置进行对应触发和执行，所执行的音效、特效、开始使用数据表进行存储，并允许自定义修改。

[0009] 进一步，所述的技能结果模块包括：触发模块，使用触发器对技能释放过程进行判定，如果技能释放完成，则触发器相应并执行下一个模块，如果技能触发特殊情况，则调取对应的函数接口处理技能触发的特征情况；后续处理模块，对完成技能释放的游戏角色状态进行还原。

[0010] 本发明另一方面提供一种通用动作游戏技能设计的方法，包括：对游戏角色释放的技能判定条件使用判定数据表进行存储，然后将数据表封装成应用程序接口，用于游戏启动时调用应用程序接口对游戏人物技能释放条件进行判断；对不同的技能进行检测，对具有相同结构特征的技能进行差分处理，对通用结果部分特征技能进行封装处理，并使用差分表数据表存储；对游戏角色释放技能时，通过对差分数据表进行索引获取通用技能特征进行直接引用执行，进而单独处理非通用技能特征部分；对技能完成状况进行判断，进而执行对释放技能对游戏角色状态造成的影响进行恢复处理。

[0011] 进一步，所述方法还包括：对游戏角色的技能信息进行获取，获取技能释放条件并进行调取，对技能及对应的判定条件使用判定数据表进行存储；用于对需要使用的技能进行分析，进一步，在释放技能时，通过基础技能接口获取基础技能数据，对差异技能特征使用对应的函数进行处理，完成技能释放还包括对技能释放时的音效、特效、开始与结束时间、帧数据、参数配置进行对应触发和执行，所执行的音效、特效、开始使用数据表进行存储，并允许自定义修改。

[0012] 对所述判定获取模块存储的判定数据表进行进一步接口化处理，使判定数据表能够被游戏程序进行调用判定；

[0013] 程序启动时，对判定检测进行初始化，进而对游戏角色技能进行先行判定。

[0014] 进一步，所述方法还包括：抽取模块，用于多游戏角色技能具有的共同特征进行抽取，并进行重新封装为基础技能接口，能够接收调用；对不具有相同特征的游戏角色技能特征使用对应的函数接口进行处理；对所述抽取模块和差异模块的技能特征均使用差分数据表存储，然后再封装为接口，还用于接收所述技能释放模块的索引。

[0015] 进一步，所述方法还包括：对需要使用的技能进行分析，进一步，在释放技能时，通过基础技能接口获取基础技能数据，对差异技能特征使用对应的函数进行处理，完成技能

释放还包括对技能释放时的音效、特效、开始与结束时间、帧数据、参数配置进行对应触发和执行，所执行的音效、特效、开始使用数据表进行存储，并允许自定义修改。

[0016] 进一步，所述方法还包括：使用触发器对技能释放过程进行判定，如果技能释放完成，则触发器相应并执行下一个模块，如果技能触发特殊情况，则调取对应的函数接口处理技能触发的特征情况；对完成技能释放的游戏角色状态进行还原。

[0017] 本发明的有益效果在于：

[0018] 1、对技能模块进行了模分，将复杂的技能系统，划分成小模块，简化了系统复杂度；

[0019] 2、技能表里，可以填写判断条件API，方便以后技能判断条件的调整；

[0020] 3、技能表里可以填技能的处理模块，这种既可以公性的技能的数据和流程，又增加程序的灵活性以及扩展性；

[0021] 4、事件触发机制，可以灵活地处理技能里触发特效音效的问题，并且统一集中管理，方便修改，扩展也很容易；

[0022] 5、提供了丰富的动作播放机制；

[0023] 6、提供了灵活的抛物线扩展机制；

[0024] 7、HardCode实现技能特效效果时，不影响框架；

[0025] 8、大量使用组件，如位移组件、动作播放组件等，这些组件有机结合在一起，组成了复杂的技能系统，当系统出现bug，主要去相应的组件，可以快速定位以及查找到bug，大大量降低系统的复杂度，提高框架的灵活性，以及可维护性。

附图说明

[0026] 图1所示为根据本发明实施方式的总体流程图；

[0027] 图2所示为根据本发明实施方式的动作游戏系统组成图；

[0028] 图3所示为根据本发明实施方式的主体流程图。

[0029]

具体实施方式

[0030] 为了使本发明的目的、技术方案和优点更加清楚，下面结合附图和具体实施例对本发明进行详细描述。

[0031] 本专利提案针以现有的技术的缺陷，首先对整个系统进行结构的分析，然后，使用表去描述相关的静态与动态的数据结构，最后，通过动态加载组件的方式去实现系统比较困难、并且特点鲜明的模块。

[0032] 图1所示为根据本发明实施方式的总体流程图。具体实施步骤如下所示：

[0033] 1、设计技能表、动作表、特效、音效表等技能系统需要用到的表。并且在根据附带上程序所需要的参数；

[0034] 2、根据需求，编写相技能主要处理流程模块Skill，接着，根据需着，写编写差异性的技能模块，继承自Skill模块，如移动技能等；

[0035] 3、编写1段技能动作处理和播放Attack(攻击)模块，保证一段动作的技能系统可以正常使用；

[0036] 4、编写n段技能动作处理和播放模块；

[0037] 5、编写均速位移模块。并根据此，编写各种各样的位移组件，以供使用；

[0038] 6、编写事件触发模块，并索引特效音效进行播放；

[0039] 7、整合进战斗系统：从输入到释放处理。

[0040] 图2所示为根据本发明实施方式的动作游戏系统组成图。技能系统涉及的模块包括输入、动作、特效、音效、位移、技能放完后的处理、技能中断、技能伤害等方方面面。当然，由于每个游戏的设计和需求都不太相当，上述的需求，也只是技能系统的一部分。有部分需求或者还没有包括在上面。由此，技能系统是一个相对复杂的系统。

[0041] 图3所示为根据本发明实施方式的主体流程图。在本实施例中，所述系统的主体流程为：从流程来看，我们可以将技能业分为：技能释放前、技能释放过程中、技能结果反馈、技能对角色后面的行为影响，四大部分。其中整个技能设计系统最为复杂在释放过程中。这个过程需求变化也最为频繁。根据该流程及模块的特点，可能将技能系统划分成以下4大重要组成部分，很多业务及细节，都是包括中这些系统里面。如：打断处理、技能释放条件判断，归属技能处理模块。处理好这四部分的流程以及设计，整个技能架构将可以建立起来。

[0042] 本发明的技术方案还包括技能释放前部分的设计，包括输入、技能判断条件。输入这个所有的游戏设计都一样，按到了技能图标，发消息给技能处理模块。如按键按键1，则触发消息：InputMsg(密钥)。这一步需求相对固定，技能释放判断条件分成两部分：分别是通用判断条件，以及特殊判断条件。

[0043] 本发明针对技能判断还提供了一种配表处理方式，具体如下(T1)～(T8)所示：

[0044] (T1).代码处理通用判断条件时，写在一个模块函数里，逐条进行检查即可。对对特效判断条件，由需求可能各种变更，则可以通过配表的方式来处理，本专利表使用Excel，导出转成lua脚本作为配置，如下表1所示：

[0045]

int	String	String@default	String
技能ID	技能名	图标	技能释放条件
1001	战士普通物理攻击	1001	Status：CheckBuff(1003)
1002			
1003			
…	…	…	…

[0046] 表1

[0047] 通过这种方法，当技能判断条件，发生变化时改表，程序增加一个处理API即可，对原来的设计和代码没有任何影响。

[0048] (T2)技能释放中的设计，技能通过前置条件判断，进入技能释放中的时候，这个过程有着共性和特殊部分，共性部分包括：

[0049] 1音效、特效触发；

[0050] 2技能都有开始和结束状态；

[0051] 3技能结束后，都需要调用对后面结果处理模块；

[0052] 4技能每帧需要刷新；

[0053] 5技能获取配置参数相关接口。

[0054] 特殊部分包括：

[0055] 1技能结构的整体结构不同；

[0056] 2技能的动作播放有多种可能。包括一段动作，二段动作，三段动作...七段动作，并且，动作播放的可能性也不同；

[0057] 3技能的位移：有可能没有位移，可能有位移。可能位移是抛物线，有可能位移是匀速运动，甚至要求是固定点位移：第一次移到往前2米，第二次往右3米等；

[0058] 4其它附带的效果及表现；

[0059] (T3)设计共性部分，表设计参考如图1所示，其中特效音效表是一个可以扩展的表，如表2所示：

[0060]

int@ukey	string@default	table_string@default	table_string@default	table_string@default
type_id	Info	Start	Pre	Cast

[0061]

1001	战士普通物理攻击			skill/physics/kiattack/cast,0
...				
...	...	...	...	...

[0062] 其列对应技能触发的事件，如start技能一开始播的特效音效。pre:起手之前播的特效、音效。可以通过扩展列，而方便增加事件对应的特效和音效处理。

[0063] 针对表，部分技能基类的伪lua代码如下：

```
function Skill:Ctor(owner, typeId)
    Super.Ctor(self)
    --读取技能配置文件
    self._IsSuccess = false; --技能初始化相关变量
    self._actor = self:CreateAnimTracker()
end

function Skill:Update()    --技能更新函数
end

function Skill:GetBaseCfg(key, defaultValue) --技能获得配置参数接口
end
```

[0064]

```
function Skill:SkillSuccess() --技能释放成功标志
    self._IsSuccess = true
end

function Skill:InitResultCheck() --初始化技能的检测方式
end

function Skill:PreStart() --技能准备开始
end

function Skill:Use(arg) --正式释放技能
end
```

[0065] 由此，我们提取了技能的通用部分的函数。简单公用部分的技能，如瞬发类的普通物理攻击技能，实现函数相关逻辑，即可以进行技能释放。接着，我们来处理特殊的技能情况。

[0066] (T4)技能结构的整体结构不同的情况，针对特殊的技能，其结构都有着普通技能的共性，因此继承自基类:Skill，那样，通用的一些接口，都不需要再进行封装，我们只需要处理差异部分。例如，冲锋，这个技能，在放技能里，需要控制移动。那么相于，在原来的结构上增加位移处理。因此，冲锋技能只需要在原来的通用技能类晨，增加向前位移功能即可以。参考伪代码，增加了SKillMove类继承自Skill，代码参考如下：

[0067]

```
SkillMove = SkillMove or class(Skill)
```

```
function SkillMove:Ctor(owner, typeId)
    Skill.Ctor(self, owner, typeId) --调用构造函数

    --位移相关变量
    self.__Path = nil
    self.__Speed = 0
    self.EndPosition = nil
    self.MoveComponent = nill;
end
```

[0068]

```
function Skill:Use(arg) --正式释放技能
    Skill. Use (arg)
    --创建位移组件  self.MoveComponent
end

function SkillMove:Update()
    Super.Update(self)
    -- self.MoveComponent 更新移动
end
```

[0069] 然后，我们在表里，相应的技能选择相应的技能处理模块就可以了。如下表：

[0070]

int	String	String	String@default	String	Int
技能 ID	技能名	处理模块名	图标	技能释放条件	特效音效 ID
1001	战士普通物理攻击	Skill	1001	Status: CheckBuff(1003)	1001
1002	冲锋	Skillmove			1002
1003					
1004					
…	…	…	…	…	…

[0071] 通过配表的方式，以及动态加载技能处理模块的方式，我们可以自由地增加类，并且在单个模块的类里“HardCode”不会影响我们的程序结构。通过这种设计方式大大增加程序的灵活性，以及实现复杂的技能。这个结构的特点是：1抽取公同特点，设计基类。2差异性放在子类处理。3通过配表索处不到的处理函数或模块，以达到设计的通用性和灵活性。

[0072] (T5)动作差异的处理，动作的播放的公同特点：

[0073] 1、开始状态；

[0074] 2、结束状态；

[0075] 3、动作里触发事件，事件再触发特效音效。

[0076] 动作播放的不同点：

[0077] 1、动作可以循环播放；

[0078] 2、动作有多段：1～n段，并且根据技能需要来播放。

[0079] 同样一个基类：Tracker，考虑到位移组件后面也需要继承此类，故设计提取一个状态管理的类，动作播放组件继承自Tracker类的接口，并增加动作播放等功能。在一些实

施例中，对于多段动画播放处理呢?先看一段动画播放与多段动画播放的区别在于:1多段有多个动画，动画是一个列表;2一段播放，符合条件时，则播下一段。那么继承自一段动画播放类:Attack，只需要处理好差异部分就可以了。动画，我们仍然只记录一个名字，如attack就好，当是2段时，就变成attack_01，attack_02，自动添加后缀来处理。参考设计的伪代码类如下：

```
local Super = tracker.Tracker
AttackSection = AttackSection or class(Super)

function AttackSection:Ctor(owner, skillId, baseAnim, eventProcessFunc, status)
    status = status or "Attack"
    Super.Ctor(self, owner, status)

    self.__BaseAnim = baseAnim
    self.__Section = nil
    self.__CurrentAnim = nil
    self.SkillId = skillId
    self._CurFrame = 0

    self.__EventProcessFunc = eventProcessFunc
end

function AttackSection:SecondBegin() --第二段开始
end

function AttackSection:ThirdBegin() --第三段开始
end
```

[0081] 然后，根据表里填的段的数量，采用不同的处理函数即可。

[0082] (T6)技能的位移技能位移的公同特点：

[0083] 1位移开始

[0084] 2位移移动

[0085] 3位移结束。

[0086] 技能位移的差异点：

[0087] 位移的方式不同。

[0088] 因为，我们可以封装一个基本的移动类：匀速位移类Move。接着，差异类封装：抛物线移动方式、匀变速移动方式、分段函数移动方式，整合到游戏里。那么，我们可以看到丰富的移动方式。当哪天策划或程序心血来潮要实验新的移动方式时，我们也可以写一个新的移动模块，不需要改动原来的结构，就可以看到激动人心的效果了。

[0089] (T7)其它附带的特效的表现

[0090] 当有一些很特殊的表现，无法使用框架去处理时，例使用"HardCode"(即硬编码)，本框架是目标是将HardCode放到合适的模块，让HardCode只是影响某个技能的小模块，而不影响技能的整体结构。例如，我们要设计一个有7段动作的技能时，要求:1-7个动作，第1个动作和第7个动作固定技能开始和结束时候播;第1个动开始时，角色往上，接着哪里有敌

人，就位移到哪里。没有时，程序随机一个位移来播；第1个动作播特效A，当碰到敌人时播特效B，第7个动作时播放特效C；这种特效的需求，当结构没有办法支持时，就想办法HardCode，首先把需要的特殊的参数填到表里。

[0091] (T8)技能结果的触发，技能结果的处理，与特效音效播放类似，采用事件触发机制，在动作里做上事件标签，如start，end，hit。其中hit表现检测是否技能打中了敌人。当一个技能需要触发2次伤害时，可以打两个检测相关的标签hit1，hit2。当检测到伤害敌人，则传给战斗结果处理模块，进行处理。用户可以根据自己游戏的设计需求，按照以下方式来实现。

[0092] 1设计技能表、动作表、特效、音效表等技能系统需要用到的表。并且在根据附带上程序所需要的参数。

[0093] 表的设计，参考本文“第二部分：技能系统配置表设计”里面的附图。

[0094] 2根据需求，编写相技能主要处理流程模块Skill，接着，根据需着，写编写差异性的技能模块，继承自Skill模块，如SkillMove等。

[0095] 3编写1段技能动作处理和播放Attack模块，保证一段动作的技能系统可以正常使用。

[0096] 4编写n段技能动作处理和播放模块。

[0097] 5编写均速位移模块。并根据此，编写各种各样的位移组件，以供使用。

[0098] 6编写事件触发模块，并索引特效音效进行播放。

[0099] 7整合进战斗系统：从输入到释放处理。

[0100] 以上所述，只是本发明的较佳实施例而已，本发明并不局限于上述实施方式，只要其以相同的手段达到本发明的技术效果，都应属于本发明的保护范围。在本发明的保护范围内其技术方案和/或实施方式可以有各种不同的修改和变化。

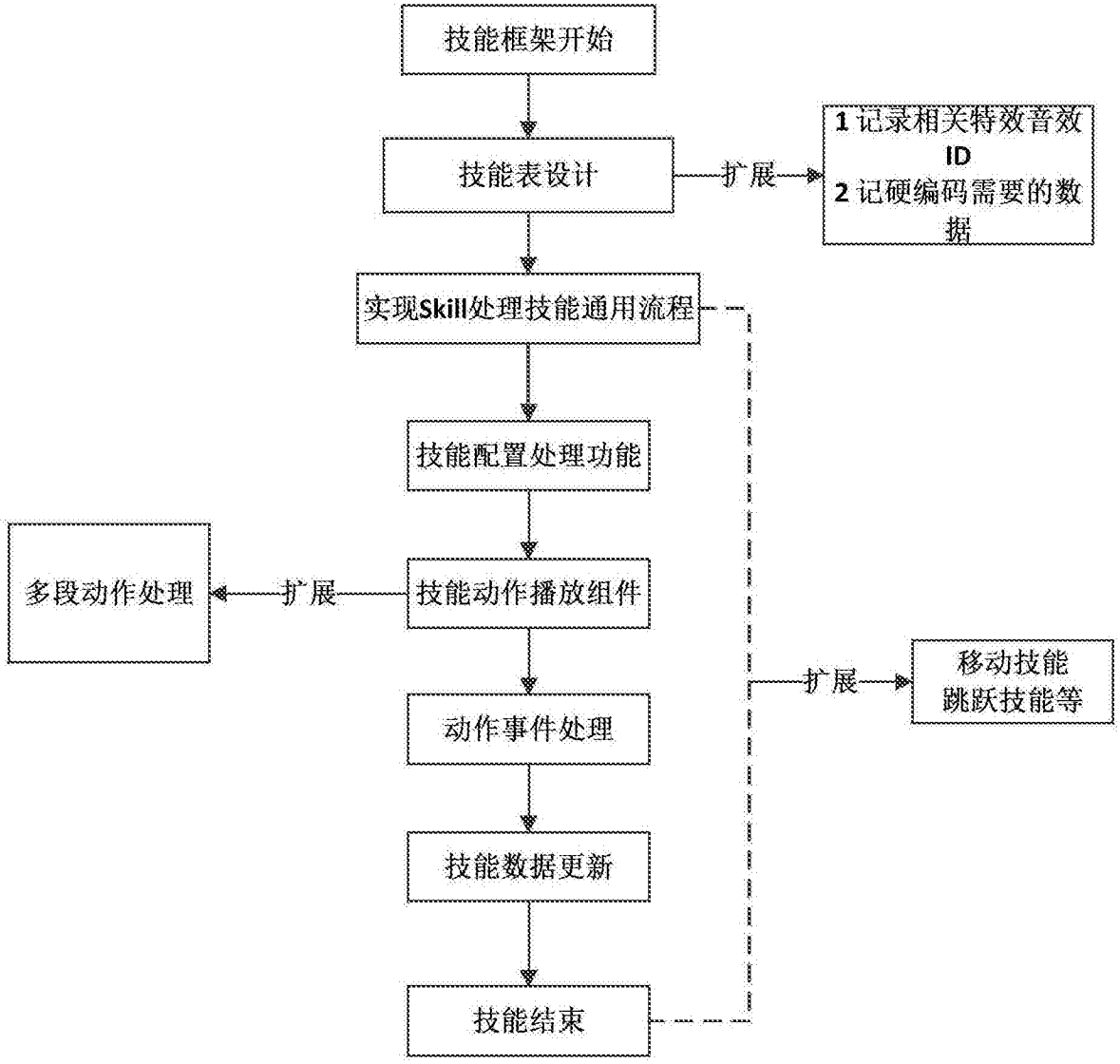

技能框架开始
技能表设计
扩展
1 记录相关特效音效ID
2 记硬编码需要的数据
实现Skill处理技能通用流程
技能配置处理功能
多段动作处理
扩展
技能动作播放组件
扩展
移动技能
跳跃技能等
动作事件处理
技能数据更新
技能结束

图1

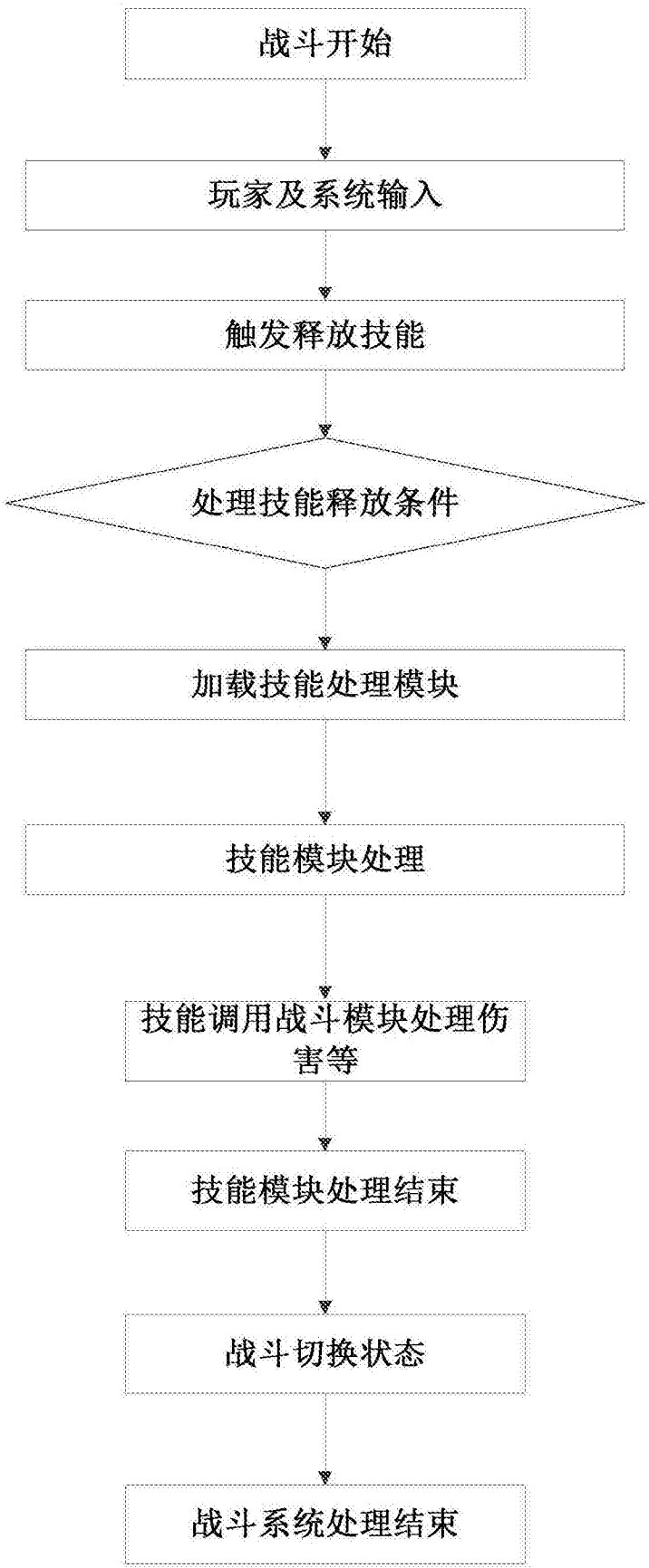

战斗开始
玩家及系统输入
触发释放技能
处理技能释放条件
加载技能处理模块
技能模块处理
技能调用战斗模块处理伤害等
技能模块处理结束
战斗切换状态
战斗系统处理结束

图2

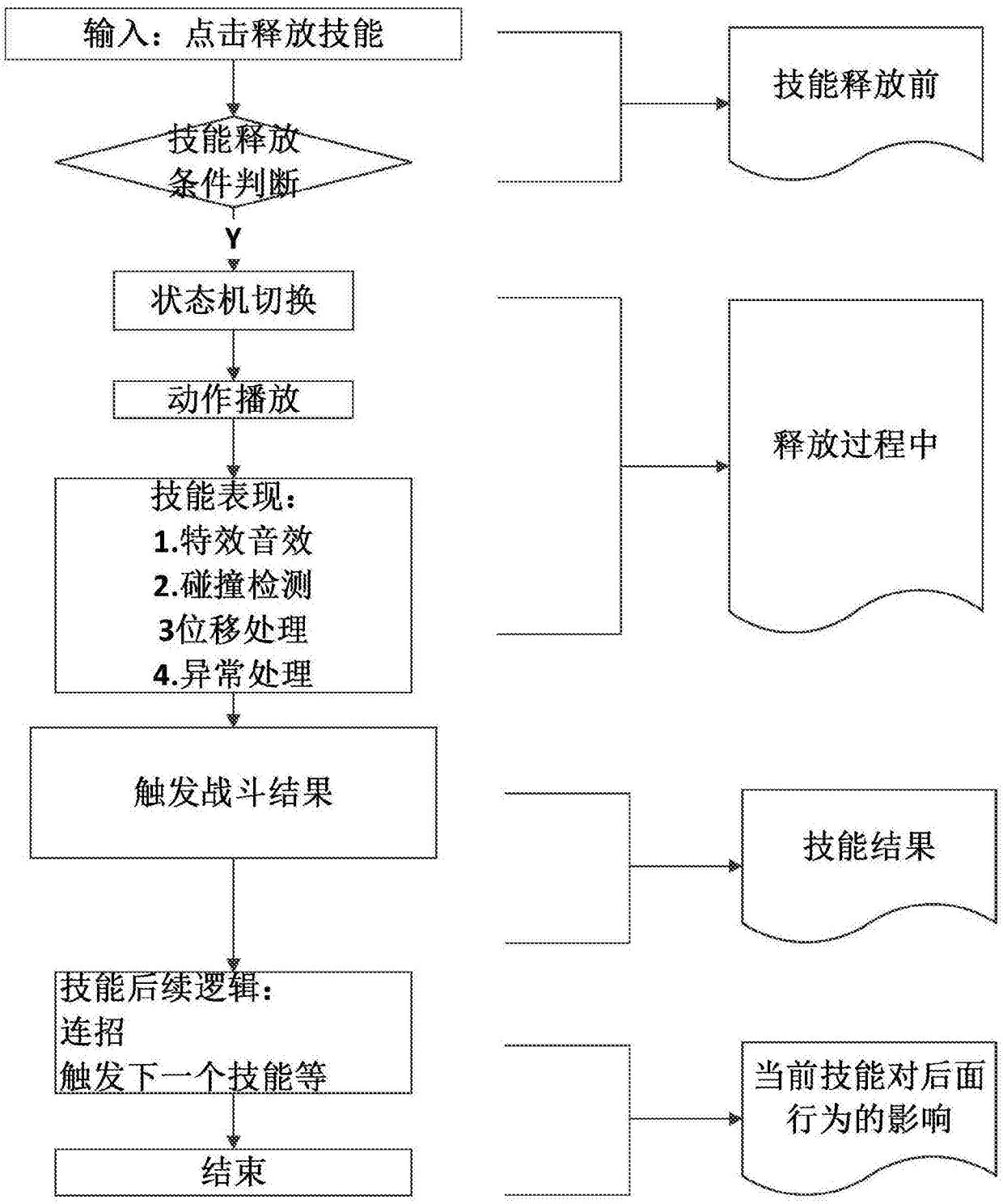

输入：点击释放技能
技能释放
条件判断
Y
状态机切换
动作播放
技能表现：
1.特效音效
2.碰撞检测
3位移处理
4.异常处理
触发战斗结果
技能后续逻辑：
连招
触发下一个技能等
结束
技能释放前
释放过程中
技能结果
当前技能对后面
行为的影响

图3