(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 111930374 A
(43)申请公布日 2020.11.13

(21)申请号 202010997873.X

(22)申请日 2020.09.21

(71)申请人 北京易真学思教育科技有限公司
地址 100043 北京市石景山区实兴大街30号院3号楼2层A-2667房间

(72)发明人 陈刚 林锋 王苗苗

(74)专利代理机构 上海知锦知识产权代理事务所(特殊普通合伙) 31327
代理人 高彦

(51)Int.Cl.
G06F 8/34(2018.01)
G06F 8/73(2018.01)
G09B 19/00(2006.01)

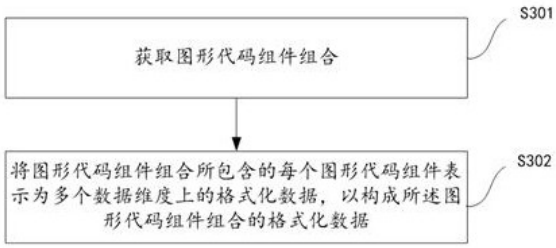
权利要求书2页 说明书14页 附图6页

(54)发明名称

数据格式化方法、装置、编程系统、设备及存储介质

(57)摘要

本申请的数据格式化方法、装置、编程系统、设备及存储介质，所述数据格式化方法包括：获取图形代码组件组合；将图形代码组件组合所包含的每个图形代码组件表示为多个数据维度上的格式化数据，以构成所述图形代码组件组合的格式化数据。本申请方案通过考虑图形代码组件上下文的格式化方式生成格式化数据，以改变现有图形编程教育软件中原代码的链表数据格式和积木格式异构的情形，降低代码解析的难度；此外，通过格式化数据可以实现更加灵活的作品检查机制。



CN 111930374 A

1.一种数据格式化方法,其特征在于,用于对由图形代码组件组成的图形代码组件组合进行格式化;所述数据格式化方法包括:

获取图形代码组件组合;

将图形代码组件组合所包含的每个图形代码组件表示为多个数据维度上的格式化数据,以构成所述图形代码组件组合的格式化数据。

2.根据权利要求1所述的数据格式化方法,其特征在于,所述多个数据维度包括:执行角色、事件、顺序逻辑、及类型;在图形代码组件具有参数的情况下,所述多个数据维度还包括参数。

3.根据权利要求2所述的数据格式化方法,其特征在于,所述格式化的方式包括:

获取图形代码组件组合中每个图形代码组件的执行角色信息;其中,各所述图形代码组件按各自的执行角色分组形成各执行角色组;

获取每个图形代码组件在所属执行角色组下相关的事件信息;其中,每个执行角色组下的各图形代码组件按各自相关的事件信息分组形成各事件组;

依排列顺序提取事件组下每个图形代码组件的类型、表示其嵌套关系的深度信息、以及具有参数的图形代码组件的参数;

根据所提取的每个图形代码组件的信息,形成每个图形代码组件的格式化数据,并按所述排列顺序形成对应的图形代码组件组合的格式化数据。

4.根据权利要求3所述的数据格式化方法,其特征在于,所述依排列顺序提取事件组下每个图形代码组件的类型、表示其嵌套关系的深度信息、以及具有参数的图形代码组件的参数,包括:

逐一访问每个图形代码组件;

将判断需要递归处理的图形代码组件,加入到第一递归队列;

从第一递归队列外的每个图形代码组件中提取信息并置入对应的格式化数据;

从第一递归队列中的每个图形代码组件中提取信息并置入对应的格式化数据。

5.根据权利要求4所述的数据格式化方法,其特征在于,所述从第一递归队列外的每个图形代码组件中提取信息,包括:

逐一筛取每个图形代码组件的有效参数;

将判断需要递归处理的图形代码组件放入第二递归队列。

6.根据权利要求5所述的数据格式化方法,其特征在于,所述第二递归队列递归处理的优先级高于第一递归队列。

7.根据权利要求4至6中任一项所述的数据格式化方法,其特征在于,所述需要递归处理的图形代码组件,包括以下至少一种:存在嵌套关系的图形代码组件;存在可选参数的图形代码组件。

8.一种数据格式化装置,其特征在于,用于对由图形代码组件组成的图形代码组件组合进行格式化;所述数据格式化装置包括:

获取模块,用于获取图形代码组件组合;

格式化单元,用于将图形代码组件组合所包含的每个图形代码组件表示为多个数据维度上的格式化数据,以构成所述图形代码组件组合的格式化数据。

9.一种图形编程系统,其特征在于,包括:

图形编程单元，用于向用户提供图形编程界面，用于接收用户操作而将所操作图形代码组件组合为图形编程作品；

格式化单元，由如权利要求8所述的数据格式化装置实现，用于对所述图形编程作品或其相关的图形代码组件组合进行格式化，以得到对应的格式化数据。

10.一种计算机设备，包括存储器和处理器，所述存储器上存储有可在所述处理器上运行的计算机程序，其特征在于，所述处理器运行所述计算机程序时执行权利要求1至7中任一项所述的数据格式化方法的步骤。

11.一种计算机可读存储介质，其上存储有计算机程序，其特征在于，所述计算机程序运行时执行权利要求1至7中任一项所述的数据格式化方法的步骤。

数据格式化方法、装置、编程系统、设备及存储介质

技术领域

[0001] 本申请实施例涉及编程教学技术领域，尤其涉及数据格式化方法、装置、编程系统、设备及存储介质。

背景技术

[0002] 基于当前的教育中对逻辑思维训练的需求，各种逻辑思维类的电子教学工具持续在涌现。尤其在当今社会对于计算机网络技术的高度依赖的情形下，编程教学课程所面对用户的年纪也越来越小。

[0003] 为此，已出现了一些较为直观且趣味的图形编程教学软件，例如麻省理工学院的简易图形化编程工具(Scratch)等，提供“搭积木”的编程方式，即用户可以将图形化的代码积木通过点击和拖拽等操作进行组合，而完成编程作品。

[0004] Scratch已经发展到了3.0版本。然而，Scratch 3.0 的用户界面有着明确的功能分区，由于采用了积木式的编程方式，所以用户代码也比较清晰。但是 Scratch 3.0 数据层的源码整体采用 JSON 数据格式进行存储，在该 JSON 文件中，角色脚本区内的每个积木都被分配了一个随机的唯一积木标识(key)，彼此之间使用链表的方式连接，当产生嵌套之后代码更加复杂，这就使得作品源码难以阅读，更无法直接进行二次开发。此外，Scratch作品运行时，每个积木是单独解析的，不同积木的数据格式存在较大的差异。

[0005] 综上所述，在 Scratch 采用的链表数据格式和积木格式异构的双重影响下，想要直接对 Scratch 的作品源码进行明文解析非常困难。

发明内容

[0006] 有鉴于此，本申请实施例中提供数据格式化方法、装置、编程系统、设备及存储介质，解决现有技术中的技术问题。

[0007] 本申请实施例提供了一种数据格式化方法，用于对由图形代码组件组成的图形代码组件组合进行格式化；所述数据格式化方法包括：

获取图形代码组件组合；

将图形代码组件组合所包含的每个图形代码组件表示为多个数据维度上的格式化数据，以构成所述图形代码组件组合的格式化数据。

[0008] 可选的，所述多个数据维度包括：执行角色、事件、顺序逻辑、及类型；在图形代码组件具有参数的情况下，所述多个数据维度还包括参数。

[0009] 可选的，所述格式化的方式包括：

获取图形代码组件组合中每个图形代码组件的执行角色信息；其中，各所述图形代码组件按各自的执行角色分组形成各执行角色组；

获取每个图形代码组件在所属执行角色组下相关的事件信息；其中，每个执行角色组下的各图形代码组件按各自相关的事件信息分组形成各事件组；

依排列顺序提取事件组下每个图形代码组件的类型、表示其嵌套关系的深度信息、以

及具有参数的图形代码组件的参数；

根据所提取的每个图形代码组件的信息，形成每个图形代码组件的格式化数据，并按所述排列顺序形成对应的图形代码组件组合的格式化数据。

[0010] 可选的，所述依排列顺序提取事件组下每个图形代码组件的类型、表示其嵌套关系的深度信息、以及具有参数的图形代码组件的参数，包括：

逐一访问每个图形代码组件；

将判断需要递归处理的图形代码组件，加入到第一递归队列；

从第一递归队列外的每个图形代码组件中提取信息并置入对应的格式化数据；

从第一递归队列中的每个图形代码组件中提取信息并置入对应的格式化数据。

[0011] 可选的，所述从第一递归队列外的每个图形代码组件中提取信息，包括：

逐一筛取每个图形代码组件的有效参数；

将判断需要递归处理的图形代码组件放入第二递归队列。

[0012] 可选的，所述第二递归队列递归处理的优先级高于第一递归队列。

[0013] 可选的，所述需要递归处理的图形代码组件，包括以下至少一种：存在嵌套关系的图形代码组件；存在可选参数的图形代码组件。

[0014] 本申请实施例还提供了一种数据格式化装置，用于对由图形代码组件组成的图形代码组件组合进行格式化；所述数据格式化装置包括：

获取模块，用于获取图形代码组件组合；

格式化单元，用于将图形代码组件组合所包含的每个图形代码组件表示为多个数据维度上的格式化数据，以构成所述图形代码组件组合的格式化数据。

[0015] 本申请实施例还提供了一种图形编程系统，包括：

图形编程单元，用于向用户提供图形编程界面，用于接收用户操作而将所操作图形代码组件组合为图形编程作品；

格式化单元，由所述的数据格式化装置实现，用于对所述图形编程作品或其相关的图形代码组件组合进行格式化，以得到对应的格式化数据。

[0016] 本申请实施例还提供了一种计算机设备，包括存储器和处理器，所述存储器上存储有可在所述处理器上运行的计算机程序，所述处理器运行所述计算机程序时执行任一项所述的数据格式化方法的步骤。

[0017] 本申请实施例还提供了一种计算机可读存储介质，其上存储有计算机程序，所述计算机程序运行时执行任一项所述的数据格式化方法的步骤。

[0018] 与现有技术相比，本申请实施例的技术方案具有以下有益效果：

本申请实施例中，通过考虑图形代码组件上下文的格式化方式生成格式化数据，以改变现有图形编程教育软件中原代码的链表数据格式和积木格式异构的情形，降低代码解析的难度；此外，通过格式化数据可以实现更加灵活的作品检查机制。

附图说明

[0019] 图1是Scratch的图形用户界面示意图。

[0020] 图2是Scratch中一个积木组合的示意图。

[0021] 图3是本申请实施例中数据格式化方法的流程示意图。

[0022] 图4是本申请实施例中在具体数据维度上执行数据格式化方法的流程示意图。

[0023] 图5是本申请实施例中格式化数据的原理示意图。

[0024] 图6是本申请实施例中使用递归队列执行数据格式化方法的流程示意图。

[0025] 图7是本申请实施例中对应图6的实际代码的格式化的流程图。

[0026] 图8是图7中S708的具体实施流程图。

[0027] 图9是本申请实施例中数据格式化装置的结构示意图。

[0028] 图10是本申请实施例中数据格式化装置中格式化模块的结构示意图。

[0029] 图11是本申请实施例中图形编程系统的结构示意图。

[0030] 图12是本申请实施例中计算机设备的结构示意图。

具体实施方式

[0031] 图形编程教学软件已被应用于孩童、青少年的编程教学中。例如麻省理工学院的简易图形化编程工具Scratch，用户可以通过在Scratch的图形用户界面中操作各种“编程积木”，并加以组合以形成图形编程作品。

[0032] 如图1所示，展示Scratch的图形用户界面示意图。

[0033] 在图1所示的Scratch的图形用户界面示意图中，左侧栏位101展示可以选择使用的代码积木，例如“运动”、“外观”、“声音”、“事件”、“控制”等各类型的代码积木，图中展示选择“外观”选项而展示的各种外观相关的代码积木；图示中部的编程区域102展示通过组合所选择的代码积木构成的图形编程作品，示例性地展示有“当旗帜(图中以图形表示)被点击”则执行角色“移动100步”至“碰到边缘就反弹”，将执行角色的旋转方式设为“左右翻转”(图中的倒三角表示下拉列表，通过操作在下拉列表可以选择动作)，执行角色“右转15度”说“你好!”播放声音“喵”并展示下一个造型；图中右侧栏位展示右下方的执行角色区域103、及其上方的动画展示区域(一般称为“舞台区”)104，其中可以展示执行图形编程作品的代码后所对应呈现的动画结果，所述执行角色可以是卡通猫“可多”，对应于上述“喵”的声音。当然，执行角色可以是多个，例如在右侧的执行角色区域104可以分开展示，例如执行角色有猫和狗，用户可以选择对应的执行角色，而在中部的编程区域102可以对应所选择的执行角色进行代码积木的编程，图中角色“猫”的纹路图案与“狗”不同，表示其被选中。

[0034] 在Scratch中，有各类型的积木，其形态各有不同。其中，大多数编写程序都是以事件积木开始，例如图1中展示的最上面一块的“当旗帜被点击”的代码积木就是事件积木。

[0035] 在一些示例中，本文所称的“图形代码组件”可以是Scratch中的代码积木，图形代码组件组合即可以是Scratch中拼装积木得到的积木组合，而图形编程作品则可以是由一或多个积木组合形成的作品，例如图1所展示的内容。需说明的是，图1所展示的一个积木组的图形编程作品只是一种示例，实际上图形编程作品并不限制积木组的数量，例如有一个或多个执行角色(例如猫、狗等)，每个执行角色可以执行一或多个事件，则会对应形成一个或多个事件相关的积木组，每个积木组以一个事件积木为起始。

[0036] 在图2中，展示一个积木组合，功能如下：当旗帜被点击后，角色会向前走10步，走完10步后如果进入了区域，会说“你好”。

[0037] 在Scratch 3.0中，源码整体采用 JSON 数据格式进行存储，JSON(JavaScript Object Notation)是一种轻量级的数据交换格式。

[0038] 示例性地,图2的积木组合对应的 JSON代码如下代码段1所示:

```
{
"积木1":{
"opcode":"event_whenflagclicked",//事件
"next":"积木2",//下一个是“积木2”
"parent":null,//不存在父积木
"inputs":{
        },
"fields":{
        }
    },
"积木2":{
"opcode":"motion_movesteps", //移动动作
"next":"积木3",//下一个是“积木3”
"parent":"积木1",//父积木是“积木1”
"inputs":{
"STEPS":[ 1, [4,"10"]]//移动步数是“10”
        },
"fields":{
        }
    },
"积木3":{
"opcode":"control_if", //如果...那么
"next":null,
"parent":"积木2",
"inputs":{
"CONDITION":[ 2, "积木4"],//条件:如果“积木4”
"SUBSTACK":[ 2, "积木5"]//分支:那么“积木5”
        },
"fields":{
        }
    },
"积木4":{
"opcode":"sensing_touchingcolor", //碰到颜色
"next":null,
"parent":"积木3",
"inputs":{
"COLOR":[1,[ 9,"#a3cc33"]]//颜色是 "#a3cc33"
        },
```

```
"fields":{
        }
    },
"积木5":{
"opcode":"looks_say", //外观:“说”
"next":null,
"parent":"积木3",
"inputs":{
"MESSAGE":[1, [ 10,"你好!"]]//说:“你好”
        },
"fields":{
        }
    }
```

其中,每个代码积木都被分配了一个随机的唯一积木标识(称为key),为查看清晰,在上述代码中简单地通过“积木1”~“积木5”等来替代性地表示积木标识,如果在实际实施中,积木标识可以是一串字母、数字、符号构成的随机字符串;可以看到,每个当前代码积木所对应的数据结构体中,通过“next”字段中的key来标注当前积木相邻的后一块积木,同时通过“parent”字段中的key标注当前积木的前一块积木,通过此方式确定积木的先后顺序;如果向前积木“parent”的值为“null”即“空”,则表示该积木为组合的首个积木,可以例如为事件积木。

[0039] 可以看到,“opcode”字段为积木的名称(也就是积木的“类型”),parent记录上一个积木的积木标识。积木的参数被分散在“inputs”和“fields”字段中,比如“碰到颜色”这块积木的参数名为 COLOR ,而“移动”积木的参数名为STEPS。

[0040] 除了彼此间的顺序关系,上述积木组合还存在更为复杂的“嵌套”关系,比如“如果…那么”积木中存在两个嵌套关系:“碰到颜色”是“如果…那么”的条件积木,“说”是满足条件时要被执行的积木。在原始的 JSON 文件中,“opcode”为“contorl_if”的积木代表用户界面的“如果…那么”积木,该积木的“inputs”字段中,“CONDITION”和“SUBSTACK”所对应的数组内部出现了两个随机的积木标识,通过这两个积木标识在代码中找到了“opcode”为“sensing_touching color”和“looks_say”的积木,这两块积木就是用户界面上的“碰到颜色”和“说”积木,代表“如果…那么”中嵌套了这两块积木。

[0041] 比较上述两段中对于参数的说明可见,在积木原始 JSON 中,嵌套关系跟参数在代码格式上非常相似,容易混淆,不利于代码的明文解析。

[0042] 在本申请实施例中,可以提供对图形代码组件组合(如积木组合)进行格式化的方案。

[0043] 如图3所示,展示本申请实施例中数据格式化方法的流程示意图。所述数据化格式方法用于对由图形代码组件组成的图形代码组件组合进行格式化;所述数据格式化方法包括:

步骤S301:获取图形代码组件组合。

[0044] 步骤S302:将图形代码组件组合所包含的每个图形代码组件表示为多个数据维度

上的格式化数据，以构成所述图形代码组件组合的格式化数据。

[0045] 在一些示例中，所述格式化指的是将积木组合所包含的每个积木表示为多个数据维度上的格式化数据。在具体实现的示例中，所述多个数据维度包括：执行角色、事件、顺序逻辑及类型；其中，在积木具有参数的情况下，所述多个数据维度还包括参数。

[0046] 其中，执行角色例如为图1中的猫“可多”或其它在执行角色库中的执行角色，图1实施例中的积木组合都可以是由“可多”为执行角色的。

[0047] 再例如，图1实施例中积木组合，可以看到其中首块积木为事件积木，之后的代码积木都归属于它，故通过所述事件积木即可确定后续代码积木的所属事件，通过确定代码积木所属事件的信息，可以唯一确定代码积木所归属于本事件相关的积木组合。

[0048] 再例如，代码积木的名称“opcode”就代表了它的名称(也即类型)，例如″motion_movesteps″表示移动。

[0049] 所述顺序逻辑包含了代码积木的顺序及嵌套关系。

[0050] 关于顺序关系，指的是积木组合中各个代码积木之间的先后排列顺序，例如图2中从上至下的各积木排列顺序，这个排列顺序同时也对应了积木组合的代码顺序，例如前述代码段1所示。

[0051] 关于嵌套关系，举例来说，图2中的“如果....那么”代码积木，其中嵌套了“碰到颜色”和“说”的代码积木。

[0052] 可选的，可以通过每个积木的深度信息来表示其嵌套位置，从嵌套外层到内层，每深一层嵌套则对积木的深度信息的取值可以加1来表示。例如，假设将每个不存在嵌套关系代码积木，通过深度信息的取值为“1”来表示，比如图2中“当旗帜被点击”、“移动”的深度信息的取值为“1”；被嵌套的积木“碰到颜色”和“说”的深度信息的取值可以设置为比“1”更深一级的“2”，如果“碰到颜色”换成其他嵌套积木，则被其所嵌套的积木的深度信息的取值就可以为更深一级的“3”。

[0053] 再例如，图2中所示例的，其中有的积木没有参数，例如“当旗帜被点击”就不是无参数的积木，而“移动”积木就是有参数的积木，这些参数可以归到对应的积木的格式化数据中。

[0054] 在Scratch中，用户是在对应执行角色所打开的执行角色区域中进行代码积木的拼装，所以相应的代码积木都会归到执行角色下，且积木组合的积木一般为事件积木，表示执行角色下的一个事件，再往下是所属的其它积木，图形代码积木之间可能存在有嵌套关系，其中有的是有参数积木而有的是无参数积木。

[0055] 在一些示例中，对应这样的上下级关系，可以进行如图4所示的格式化的流程示意图对应的方法，其具体包括：

步骤S401：获取每个图形代码组件的执行角色信息；其中，各所述图形代码组件按各自的执行角色分组形成各执行角色组。

[0056] 举例来说，例如图形编程作品包含以小猫“可多”为执行角色的第一图形代码组件组合，那么这个第一图形代码组件组合中的每个图形代码组件就会归在小猫“可多”这一执行角色组；同理，图形编程作品中若还包含以小狗为执行角色的第二图形代码组件组合，那么其中的每个图形代码组件就会归在小狗这一执行角色组中。

[0057] 步骤S402：获取每个图形代码组件在所属执行角色组下相关的事件信息；其中，每

个执行角色组下的各图形代码组件按各自相关的事件信息分组形成各事件组。

[0058] 举例来说，图2中的图形代码组件组合的执行角色是小猫“可多”，则归属于小猫“可多”的执行角色组中，进一步的，该图形代码组件组合的首个图形代码组件是事件类型的，“当旗帜被点击”，则以其为事件形成事件组，后续的图形代码组件都归在此事件组中。

[0059] 步骤S403：依排列顺序提取事件组下每个图形代码组件的类型、表示其嵌套关系的深度信息、以及具有参数的图形代码组件的参数。

[0060] 所述排列顺序为每个图形代码组件在所在的图形代码组件组合中的先后顺序。

[0061] 如之前实施例所述，类型通过从“opcode”提取的名称表示；嵌套关系通过深度信息表示，例如“当旗帜被点击”、“移动”、“如果....那么”的深度信息的取值可以设为“1”，而被“如果....那么”嵌套积木嵌套的“碰到颜色”和“说”的深度信息的取值可以为“2”；另外，参数例如为图2中“移动”中的参数“10”，“碰到颜色”中的颜色代码“#a3cc33”等。

[0062] 步骤S404：根据所提取的每个图形代码组件的信息，形成每个图形代码组件的格式化数据，并按所述排列顺序形成对应的图形代码组件组合的格式化数据。

[0063] 图2的图形代码组件组合对应于之前代码段1，在经过格式化后，得到的图形代码组件组合中每个图形代码组件的格式化数据表示成[名称，[参数列表]，深度信息]，而图形代码组件组合的格式化数据中集合了这些格式化数据，可以表示为数组形式。举例来说，代码段1可以格式化为以下代码段2形式：

```
{
        "event_whenflagclicked":
        [
                ["event_whenflagclicked", 1],
                ["motion_movesteps",["10"], 1],
                ["control_if", 1],
                ["sensing_touchingcolor", ["#a3cc33"], 2],
                ["looks_say", ["你好!"], 2]
    ]
}
```

具体说明上述代码段2中各个积木的格式化数据的含义。

[0064] 例如，“event_whenflagclicked”，表示这一组积木组合的事件积木是“当旗帜被点击”。“event_whenflagclicked”下方展示的数组中的每个元素为一个积木的格式化数据。

[0065] 其中，积木的格式化数据的形式为[名称，[参数列表]，深度信息]，名称对应的是积木的opcode，参数列表包含积木的各项参数，深度信息用于表示积木的嵌套关系。

[0066] 在代码段2的示例中，各个图形编码组件(即积木)的格式化数据的排列顺序同图2中对应的图形编码组件的排列顺序是一致的，即图形代码组件组合中图形代码组件的排列顺序的信息也被还原在了对应的格式化数据中。

[0067] 可以理解的是，在该排列顺序下，根据深度信息的变化可以对应得到嵌套关系的变化，例如一个嵌套关系的发生或退出。举例来说，最外层的图形代码组件的深度信息为“1”，按在图形代码组件组合中图形代码组件的排列顺序，出现嵌套关系后深度增加“1”，每

深一层嵌套深度增加“1”，每解除一层嵌套则深度减少“1”；若还原到深度信息为1时，说明退出了最外层的嵌套关系；例如从“如果...那么”到“碰到颜色”和“说”，深度信息从“1”变化到“2”，若该图形代码组件组合中后续的图形代码组件出现深度信息为“1”时，说明退出了“如果...那么”的嵌套。同理，若之后再出现组合中的图形代码组件的深度信息增加，说明又出现了新的嵌套关系。

[0068] 进一步具体来讲，数组中的第一个元素是格式化后的事件积木的格式化数据，本示例中事件积木是“当旗帜被点击”，无参数，深度为1。

[0069] 数组中的第二个元素对应“移动”积木的格式化数据，其有一个参数，为 10，同时跟事件积木是拼接的关系，所以深度信息相同也为1。

[0070] 数组中的第三个元素是“如果…那么”积木的格式化数据，该积木包含嵌套积木同时没有自身的参数，由于嵌套关系只通过相邻关系+深度信息表达，所以保持了“如果…那么”积木格式的简洁。第三个元素的深度信息为 1 表明与第二块积木仍是拼接关系。

[0071] 数组中的第四个元素是“碰到颜色”积木的格式化数据，包含一个参数“a3cc33”，是 RGB 格式的颜色。该积木深度信息为 2，而相邻的上一个积木深度信息为1，表明“碰到颜色”积木是“如果…那么”积木的嵌套内积木。

[0072] 数组中的第五个元素是“说”积木的格式化数据，包含一个参数“你好!”，深度为2，表明“说”积木和“碰到颜色”积木都是“如果…那么”积木的嵌套内积木。如果后面还有其他积木，如果深度变为 3，说明进入了更深一层的嵌套，如果深度变为 1，说明结束了“如果…那么”积木的嵌套，回到了“如果…那么”积木的同一深度。

[0073] 为更直观说明图4方法的原理，可以参考图5所示，其中右侧展示执行角色组“小猫”（如小猫“可多”）、“小狗”，“小猫”的执行角色组下级有事件组“当旗帜被点击”，事件组当旗帜被点击下级有各个图形代码组件的格式化数据，如“当旗帜被点击”、“移动”、“如果...那么”、“碰到颜色”、及“说”。

[0074] 图5中左侧展示的是图形代码组件“当旗帜被点击”和“移动”的格式化数据更具体的数据结构示意图。图形代码组件“当旗帜被点击”是无参数的，其包含操作码“event_whenflagclicked”（即“事件_当旗帜被点击”）和深度信息“1”；而“移动”是有参数的，其包含操作码“motion_movesteps”（即“动作_移动_步”）、参数（移动步数“10”）、及深度信息“1”。

[0075] 如图6所示，展示上述步骤S403在可能的实施方式中实际的处理流程，包括：

步骤S601：逐一访问每个图形代码组件；

步骤S602：将判断需要递归处理的图形代码组件，加入到第一递归队列；

步骤S603：从第一递归队列外的每个图形代码组件中提取信息并置入对应的格式化数据；

步骤S604：从第一递归队列中的每个图形代码组件中提取信息并置入对应的格式化数据。

[0076] 在一些示例中，需要递归处理的图形代码组件可以包括以下至少一种：存在嵌套关系的图形代码组件；存在可选参数的图形代码组件。

[0077] 举例来说，存在嵌套关系的积木例如图2中的“如果...那么”积木；存在可选参数的图形代码组件，例如为下拉列表积木，如图1中的“将旋转方式设为”积木。

[0078] 以下结合之前实施例中,现有Scratch的JSON代码即代码段1、及其转化后的格式化数据的代码段2,在图7实施例中展示上述处理过程实际实现的流程图。

[0079] 在Scratch 中,有效的积木组合的首个积木必须是事件积木,故找到链表中的事件积木“event”作为启动积木,依顺序格式化处理每个当前积木;当每个当前积木处理完后,会读取当前积木中next指向的下一块积木继续格式化处理;一旦碰到需要递归处理的积木,会生成递归任务插入到“递归队列”中,例如以下流程中的第一递归队列即递归队列1、和第二递归队列即递归队列2,并在处理 next 指向的下一积木之前处理;如果递归队列中存在任务时,会优先处理递归队列中的任务,处理递归会开启新的流程,整个处理过程中的深度信息会+1。

[0080] 以对Scratch的代码进行格式化为例,在图7中,展示对实际代码进行格式化的流程,具体包括:

步骤S701:开始步骤。

[0081] 步骤S702:inputs第一次格式化。

[0082] 举例来说,判断当前积木是否包含需归到递归队列的积木标识(key),若有则加入到递归队列1;对于在 Scratch 中包含具有下拉列表的积木,下拉列表的积木可能是由多块积木拼接成的,在步骤 S702中可以把此多块积木拼接成一块处理。例如图1中的“播放声音”积木和其中的“喵”,“喵”由下拉列表所选择,看似“喵”所是“播放声音”积木的参数,但实际上“喵”可能是属于“播放声音”积木以外的另一积木中的参数,因此,可以在步骤 S702中将这两个积木拼接为一块。

[0083] 步骤S703:fields格式化。

[0084] 举例来说,Scratch 中 inputs 和 fields 两个字段都可以携带数据,比如在S702中拼接带有下拉列表的两块积木为一块时,下拉列表选中的参数可能会保存在拼接成一块积木的fields 中;为了统一处理,在S703中可以把fields合并到 inputs中,即将参数合并到inputs中。

[0085] 步骤S704:inputs第二次格式化。

[0086] 合并了 fields 之后,对 inputs作第二次格式化,此流程主要是为了特殊处理事件积木的参数和“自定义积木”。

[0087] 步骤S705:判断inputs是否为空;若否,则至步骤S706;若是,则至步骤S708;

步骤S706、生成无参数的积木的格式化数据。

[0088] 如果经过 S705判断inputs列表为空,说明该积木没有参数,按照格式生成无参数的积木的格式化数据,加入到格式化结果的数组中。

[0089] S707、S712 处理递归队列1:递归队列1是全局共享的队列,也就是说在解析当前积木时,递归队列1可能已经包含了待处理的任务。在 S702 中,如果发现inputs中包含需要递归处理的积木,那么会把要处理的积木key和深度组合成一个“递归任务”加入到递归队列1队尾。使用“队列+递归”的优势是嵌套积木和被嵌套积木在最终的输出数组中是相邻的,保留了上下文,同时根据深度信息又可以推测出嵌套的层级关系。

[0090] 递归队列 1 主要是用来处理那些已知的递归积木,往往非常明显,比如“重复执行”和“如果...那么...否则”都是明显的嵌套积木,判断方式是被嵌套的积木key 会出现在此类积木的inputs字段中。例如,当遍历重复执行积木的 inputs 列表时,发现字段

“SUBSTACK”时,内部的值一定是被嵌套积木的key,“如果...那么”积木也同样有这个key。不过,嵌套积木虽然发现的早,但是此时当前积木的参数可能还没有处理完,所以需要提前把 SUBSTACK 中的 key 先加到递归队列1中,等待当前的积木完全处理完毕后,再去检查递归队列1是否有需要处理的积木。

[0091] S708、inputs 第三次格式化。

[0092] 本步骤的主要目的是对每个积木的 inputs 中的每一组数据进行所需参数的筛选,并加入到参数列表,且在此流程中若还遇到包含需要递归处理的积木,则将积木加入到第二递归队列,即递归队列2中;需要筛除的递归队列 2 的作用是收集那些不是特别明显的需要递归处理的积木,通常是积木中放入了其他积木,例如“移动10步”积木中的“10”参数替换成其它积木,例如“响度”等;可选的,递归队列 2 的优先级高于递归队列 1,即在处理递归队列时,先处理递归队列2中的积木,再处理递归队列1中的积木。

[0093] 在一些实施例中,步骤S708中,可以逐一筛取每个图形代码组件的有效参数,将判断需要递归处理的图形代码组件放入第二递归队列。

[0094] 如图8所示,展示实施例中S708的处理流程图。

[0095] 步骤S801:开始;

步骤S802:将inputs中的各input排序后逐个遍历;

在具体实施中,之所以对inputs排序是因为:Scratch 中包含多组参数的积木,有时候参数中标识key的顺序不是固定的;由“与”连接的两个积木,这两个积木中的任意一个先执行得到的结果会不同,故此处确定一种排序后再继续处理。所以需要对其中的各个input进行排序。

[0096] 在遍历每个input的过程中:

步骤S803:判断input长度是否为3;若否,则进入S804;若是,则进入S805;

在具体实施中,input 是 inputs 中的每一个字段对应的值。由于某些积木上可以嵌套其他积木,比如“移动10步”积木10步位置为一圆形框,在Scratch中称为数字框,其中除了放置参数以外,还可以放置其他圆形的积木,例如“响度”积木插入“移动10步”中10步的位置,这会使得“移动10步”积木的inputs 发生变化,“响度”积木的key 会被插到“移动10步” inputs 中的STEPS字段中,在这种情况下,对应的数据长度会变成 3。

[0097] 具体的,“移动10步”的input对应为"STEPS":[1, [4,"10"]],假设响度积木的key为“abc”,则“移动“响度”步”对应的input为"STEPS":[3,"abc",[4,"10"]],可见[ ]中的数据长度变为了3,因此需要特殊判断。

[0098] 步骤S804:去除input中的空值,标记input的末尾元素last。

[0099] 步骤S806:判断last是否是字符;若是,说明可能是key或字符形式的参数,则进入S807;若否,则进入S808;

步骤S807:判断last是否为某个积木的标识(key);若是,说明last需要递归处理(可能有嵌套),则进入S809;若否,说明last只是普通参数,则进入S810;

步骤S809:将标识对应的积木加到递归队列2;之后进入S811;

在具体实施中,之所以选择input的末尾元素,是因为所需提取的参数一般就在input的最后,例如COLOR":[1,[9,"#a3cc33"]]中的"#a3cc33";又例如,在下拉列表的积木中,比如“碰到“鼠标指针””,“鼠标指针”是下拉选择的参数,其中涉及拼接的“碰到”积木以及

存放“鼠标指针”参数的其它积木，“碰到”积木中input的末尾元素是这个其它积木的key，在递归队列中进行递归处理就可以跳转到该其它积木，而“鼠标指针”参数就存放在该其它积木的fields中，在JSON代码中，“鼠标指针”的参数例如表示为“_mouse_”，对应的字段例如表示为：“TOUCHINGOBJECTMENU”；在fields和inputs合并后从inputs提取或直接从合并前的fields直接提取input末尾元素的方式，就可以从该其它积木提取到“鼠标指针”参数。

[0100] 步骤S811：inputs遍历是否结束；若是，则结束；若否，则回到步骤S802；

步骤S810：将last拼接到参数列表中；之后进入S811；

步骤S808：标识是否是广播；若是，则进入S812；若否，则进入S813：

其中，广播是 Scratch 中的一类特殊积木，例如“广播消息”积木，其结构跟其他积木不同，所以需要特殊处理。

[0101] 步骤S812：把last的第一个元素last[1]加入到参数列表，即广播类型积木的参数；之后进入S811；

步骤S813：把last的末尾元素加到参数列表；之后进入S811；

步骤S805：input的第一个元素input[1]是否是其它积木的key；若是，input带有key的都可能与递归相关，这里表示数字框中放入了原生的变量，则进入S814；若否，则进入S815；

步骤S814：将input[1]加入到递归队列2；之后进入步骤S811；

步骤S815：input[1]是否是数组；若是，则进入步骤S816；若否，则则进至S804，进行参数的识别即可。

[0102] 步骤S816：表示数字框中放入了自定义的变量(位于数组中)，并将input[1]加入到递归队列2。

[0103] 在某些情形下，input[1]和last[1]可能会相同，例如"STEPS":[1, [4,"10"]]中，1、4都是Scratch中的内部值，可以略去，则在"STEPS":["10"]中，last[1]和input[1]得到的参数都是“10”，如果如前述S803得到的input的数据长度为3，例如"STEPS":[3,"abc",[4,"10"]]，那么last[1]和input[2]皆为“10”，"abc"可以表示放入移动积木中的其它积木的key。

[0104] 回到图7的流程，在S708之后，还包括：

步骤S709、生成当前的含参数的积木的格式化数据：S708 结束后会生成当前积木的参数列表，按照格式组合出含参数的积木的格式化数据 [名称，参数列表]，深度]，加入到最终要输出的数组中。

[0105] 步骤S710、对递归队列2 进行处理。

[0106] 步骤S711：对递归队列1进行处理。

[0107] 因为递归队列2 与原积木的上下文更加紧密，所以先处理递归队列 2，再处理递归队列 1。

[0108] 如图9所示，展示本申请实施例中提供的数据格式化装置的结构示意图。所述数据格式化装置用于对由图形代码组件组成的图形代码组件组合进行格式化。本实施例中的数据格式化装置900，对应于图3实施例中的数据格式化方法，因此本实施例中的具体实施细节可以参考上述实施例，此处不对技术细节进行重复赘述。

[0109] 所述数据格式化装置900包括：

获取模块910，用于获取图形代码组件组合；

格式化单元920，用于将图形代码组件组合所包含的每个图形代码组件表示为多个数据维度上的格式化数据，以构成所述图形代码组件组合的格式化数据。

[0110] 可选的，所述多个数据维度包括：执行角色、事件、顺序逻辑、及类型；在图形代码组件具有参数的情况下，所述多个数据维度还包括参数。

[0111] 可选的，如图10所示，所述格式化单元920具体可以包括：

第一提取模块921，用于获取图形代码组件组合中每个图形代码组件的执行角色信息；其中，各所述图形代码组件按各自的执行角色分组形成各执行角色组；

第二提取模块922，用于获取每个图形代码组件在所属执行角色组下相关的事件信息；其中，每个执行角色组下的各图形代码组件按各自相关的事件信息分组形成各事件组；

第三提取模块923，依排列顺序提取事件组下每个图形代码组件的类型、表示其嵌套关系的深度信息、以及具有参数的图形代码组件的参数；

生成模块924，用于根据所提取的每个图形代码组件的信息，形成每个图形代码组件的格式化数据，并按所述排列顺序形成对应的图形代码组件组合的格式化数据。

[0112] 可选的，所述第三提取模块923，可以包括：

处理控制模块9231，用于控制对每个图形代码组件的逐一处理，将判断需要递归处理的图形代码组件，加入到第一递归队列；

第一处理模块9232，用于从第一递归队列外的每个图形代码组件中提取信息并置入对应的格式化数据；

第二处理模块9233，用于从第一递归队列中的每个图形代码组件中提取信息并置入对应的格式化数据。

[0113] 可选的，所述第二处理模块9233还可以包括：

参数筛选子模块92331，用于逐一筛取每个图形代码组件的有效参数；

处理控制子模块92332，用于将判断需要递归处理的图形代码组件放入第二递归队列。

[0114] 在本实施例中，将可选实现的功能模块通过虚框标注，以表示并非以此为限。

[0115] 可选的，所述第二递归队列递归处理的优先级高于第一递归队列。

[0116] 可选的，所述需要递归处理的图形代码组件，包括以下至少一种：存在嵌套关系的图形代码组件；存在可选参数的图形代码组件。

[0117] 如图11所示，展示本申请实施例中的一种图形编程系统的结构示意图。所述图形编程系统110可以包括：

图形编程单元111，用于向用户提供图形编程界面，用于接收用户操作而将所操作图形代码组件组合为图形编程作品；

格式化单元112，由例如图9所示的数据格式化装置900实现，用于对所述图形编程作品或其相关的图形代码组件组合进行格式化，以得到对应的格式化数据。

[0118] 在一些示例中，所述图形编程系统可以是基于Scratch实现，包含所述图形编程单元，以用于与用户交互来实现积木组合的编程；而前述实施例中的数据格式化装置可以作为组件、模块形式集成到图形编程系统中，以用于对图形编程作品或其相关的参考答案、比对模板等中的图形代码组件组合进行格式化。

[0119] 如图12所示，展示本申请实施例中的计算机设备的结构示意图。

[0120] 所述计算机设备120包括存储器121和处理器122，所述存储器121上存储有可在所

述处理器122上运行的计算机程序，所述处理器122运行所述计算机程序时执行前述例如图3或4实施例的数据格式化方法或其子流程(图6、图7、图8等)的步骤。

[0121] 在一些示例中，所述处理器122可以是实现计算功能的组合，例如包含一个或多个微处理器组合，数字信号处理(Digital Signal Processing，DSP)、ASIC等；所述存储器121可能包含高速RAM存储器，也可能还包括非易失性存储器(Non-volatile Memory)，例如至少一个磁盘存储器。

[0122] 在一些示例中，所述计算机设备120可以实现于例如服务器、服务器组、台式机、笔记本电脑、智能手机、平板电脑、智能手环、智能手表、或其它智能设备、或这些智能设备通信连接而形成的处理系统。

[0123] 本申请实施例还可以提供计算机可读存储介质，其上存储有计算机程序，所述计算机程序运行时执行例如图3或4实施例中的数据格式化方法或其子流程(图6、图7、图8等)中的步骤。

[0124] 即，上述本发明实施例中的数据格式化方法被实现为可存储在记录介质(诸如CD ROM、RAM、软盘、硬盘或磁光盘)中的软件或计算机代码，或者被实现通过网络下载的原始存储在远程记录介质或非暂时机器可读介质中并将被存储在本地记录介质中的计算机代码，从而在此描述的方法可被存储在使用通用计算机、专用处理器或者可编程或专用硬件(诸如ASIC或FPGA)的记录介质上的这样的软件处理。可以理解，计算机、处理器、微处理器控制器或可编程硬件包括可存储或接收软件或计算机代码的存储组件(例如，RAM、ROM、闪存等)，当所述软件或计算机代码被计算机、处理器或硬件访问且执行时，实现在此描述的数据格式化方法。此外，当通用计算机访问用于实现在此示出的数据格式化方法的代码时，代码的执行将通用计算机转换为用于执行在此示出的数据格式化方法的专用计算机。

[0125] 与现有技术相比，本申请实施例的技术方案具有以下有益效果：

本申请实施例中，通过考虑图形代码组件上下文的格式化方式生成格式化数据，以改变现有图形编程教育软件中原代码的链表数据格式和积木格式异构的情形，降低代码解析的难度；此外，通过格式化数据可以实现更加灵活的作品检查机制。

[0126] 在上述实施例中，可以全部或部分地通过软件、硬件、固件或者其任意组合来实现。当使用软件实现时，可以全部或部分地以计算机程序产品的形式实现。计算机程序产品包括一个或多个计算机程序。在计算机上加载和执行计算机程序指令时，全部或部分地产生按照本申请的流程或功能。计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。计算机程序可以存储在计算机可读存储介质中，或者从一个计算机可读存储介质向另一个计算机可读存储介质传输。

[0127] 例如，前述图9、10实施例等中的各个功能模块可以是软件实现；或者也可以是软硬件配合实现，例如通过计算机设备实施例中的处理器运行存储器的计算机程序实现；或者，也可以是通过硬件电路实现。

[0128] 此外，在本申请各个实施例中的各功能模块可以集成在一个处理部件中，也可以是各个模块单独物理存在，也可以两个或两个以上模块集成在一个部件中。上述集成的部件既可以采用硬件的形式实现，也可以采用软件功能模块的形式实现。上述集成的部件如果以软件功能模块的形式实现并作为独立的产品销售或使用时，也可以存储在一个计算机可读存储介质中。该存储介质可以是只读存储器，磁盘或光盘等。

[0129] 例如，前述图9、10所示实施例中各个功能模块、子模块可以是独立、单一的程序实现，也可以是一程序中的不同程序段分别实现，在某些实施场景中，这些功能模块可以位于一个物理设备，也可以位于不同的物理设备但相互通信耦合。

[0130] 在本说明书的描述中，参考术语“一个实施例”、“一些实施例”、“示例”、“具体示例”、或“一些示例”等的描述意指结合该实施例或示例描述的具体特征、结构、材料或者特点包括于本申请的至少一个实施例或示例中。而且，描述的具体特征、结构、材料或者特点可以在任一个或多个实施例或示例中以合适的方式结合。此外，在不相互矛盾的情况下，本领域的技术人员可以将本说明书中描述的不同实施例或示例以及不同实施例或示例的特征进行结合和组合。

[0131] 此外，术语“第一”、“第二”仅用于描述目的，而不能理解为指示或暗示相对重要性或者隐含指明所指示的技术特征的数量。由此，限定有“第一”、“第二”的特征可以明示或隐含地包括至少一个该特征。在本申请的描述中，“多个”的含义是两个或两个以上，除非另有明确具体的限定。

[0132] 流程图中或在此以其他方式描述的任何过程或方法描述可以被理解为，表示包括一个或更多个用于实现特定逻辑功能或过程的步骤的可执行指令的代码的模块、片段或部分。并且本申请的优选实施方式的范围包括另外的实现，其中可以不按所示出或讨论的顺序，包括根据所涉及的功能按基本同时的方式或按相反的顺序，来执行功能。

[0133] 在流程图中表示或在此以其他方式描述的逻辑和/或步骤，例如，可以被认为是用于实现逻辑功能的可执行指令的定序列表，可以具体实现在任何计算机可读介质中，以供指令执行系统、装置或设备(如基于计算机的系统、包括处理器的系统或其他可以从指令执行系统、装置或设备取指令并执行指令的系统)使用，或结合这些指令执行系统、装置或设备而使用。

[0134] 例如，前述图3、4等实施例中的数据格式化方法等，其中的各个步骤的顺序可以在具体场景中加以变化，并非以上述描述为限。

[0135] 虽然本说明书实施例披露如上，但本发明并非限定于此。任何本领域技术人员，在不脱离本说明书实施例的精神和范围内，均可作各种更动与修改，因此本发明的保护范围应当以权利要求所限定的范围为准。

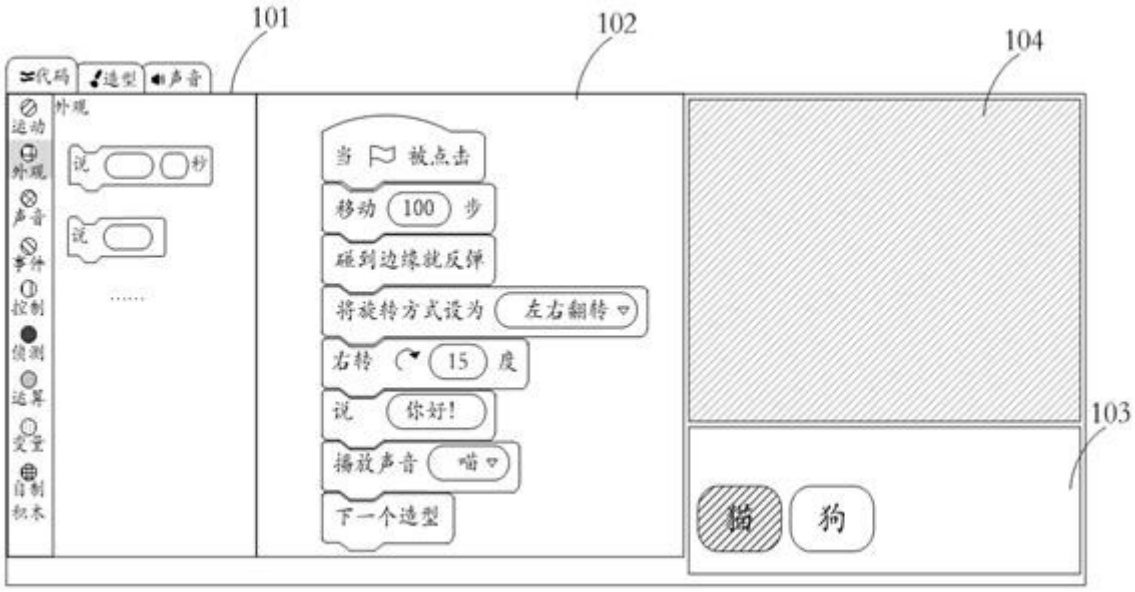

101
102
104
103
代码
造型
声音
运动
外观
声音
事件
控制
侦测
运算
变量
自制积木
外观
说 秒
说
.....
当 被点击
移动 100 步
碰到边缘就反弹
将旋转方式设为 左右翻转
右转 15 度
说 你好!
播放声音 喵
下一个造型
猫
狗

图1

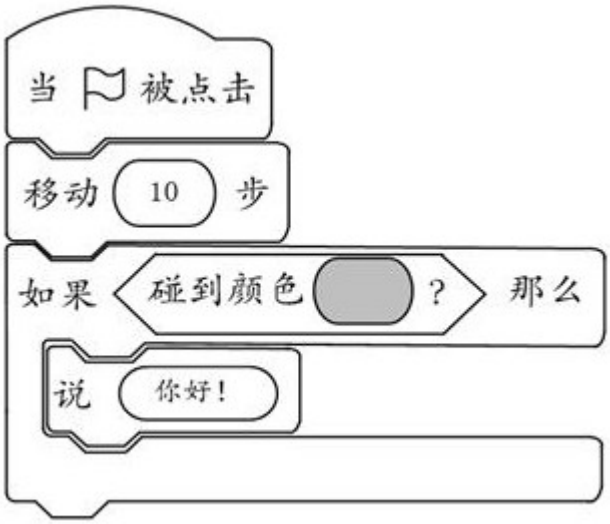

当 被点击
移动 10 步
如果 碰到颜色 ? 那么
说 你好!

图2

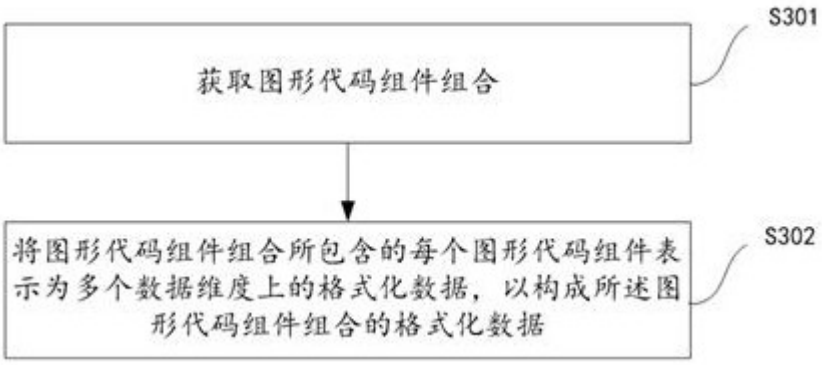

S301
获取图形代码组件组合
S302
将图形代码组件组合所包含的每个图形代码组件表示为多个数据维度上的格式化数据，以构成所述图形代码组件组合的格式化数据

图3

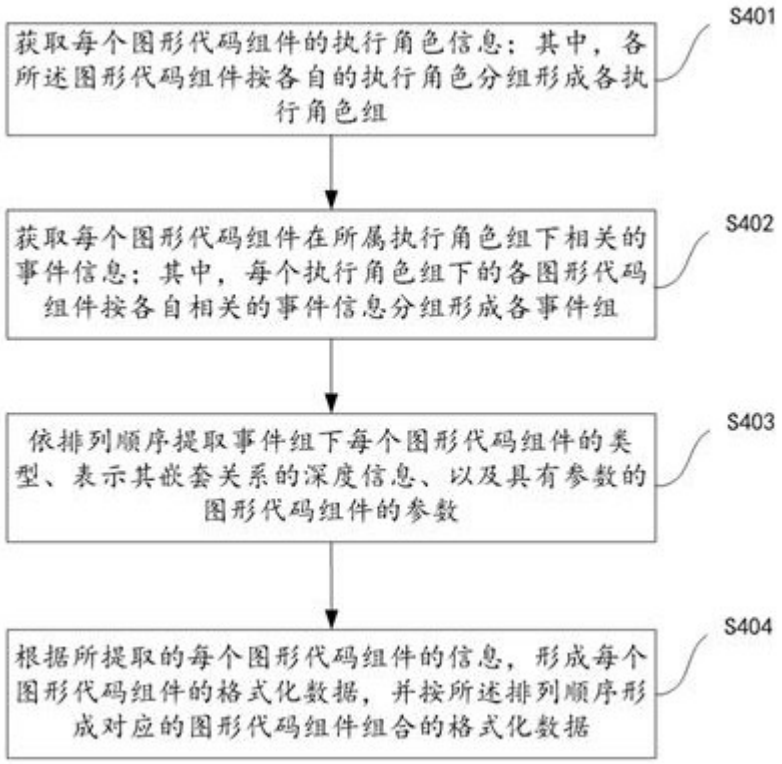

S401
获取每个图形代码组件的执行角色信息；其中，各所述图形代码组件按各自的执行角色分组形成各执行角色组
S402
获取每个图形代码组件在所属执行角色组下相关的事件信息；其中，每个执行角色组下的各图形代码组件按各自相关的事件信息分组形成各事件组
S403
依排列顺序提取事件组下每个图形代码组件的类型、表示其嵌套关系的深度信息、以及具有参数的图形代码组件的参数
S404
根据所提取的每个图形代码组件的信息，形成每个图形代码组件的格式化数据，并按所述排列顺序形成对应的图形代码组件组合的格式化数据

图4

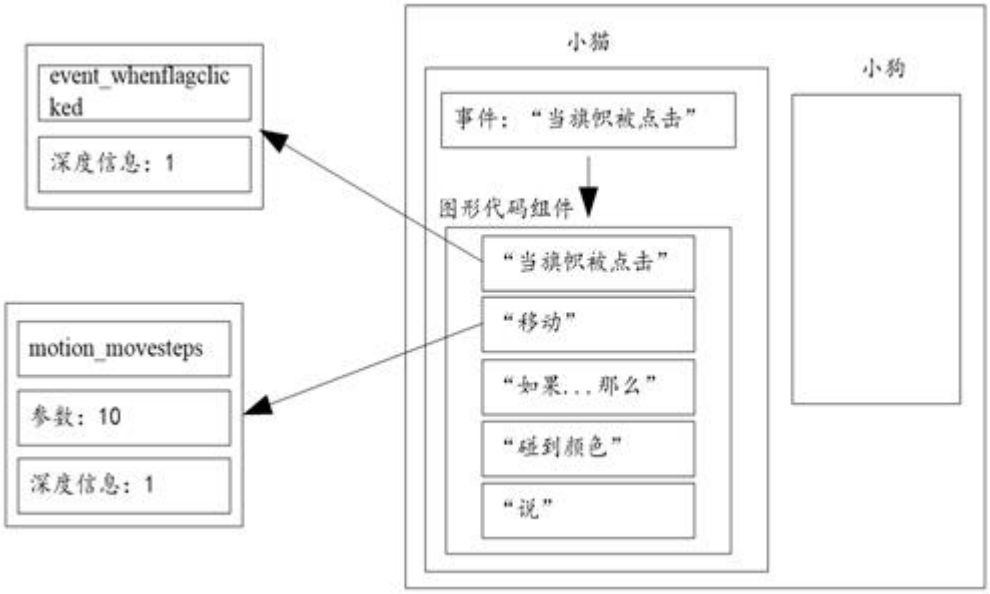

event_whenflagclicked
深度信息：1
motion_movesteps
参数：10
深度信息：1
小猫
事件：“当旗帜被点击”
图形代码组件
“当旗帜被点击”
“移动”
“如果...那么”
“碰到颜色”
“说”
小狗

图5

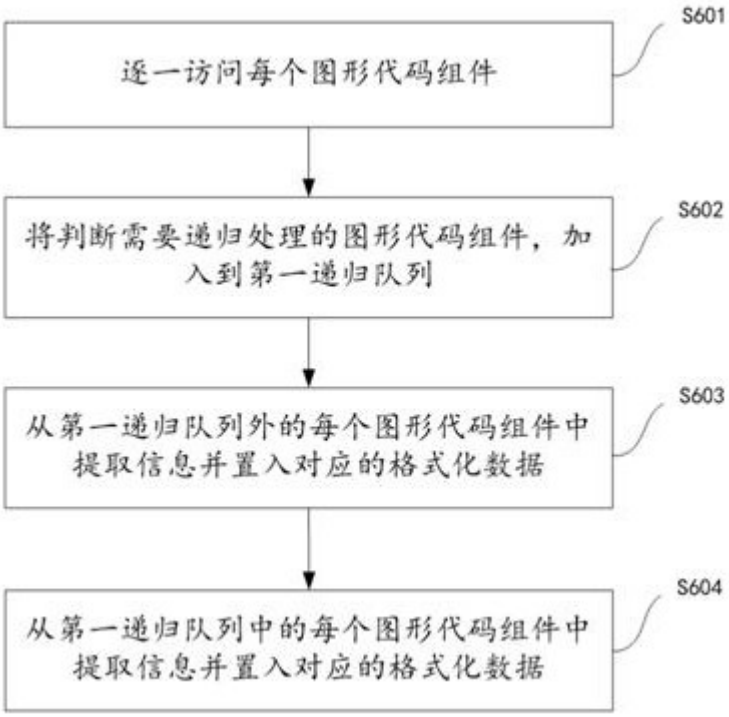

S601
逐一访问每个图形代码组件
S602
将判断需要递归处理的图形代码组件，加入到第一递归队列
S603
从第一递归队列外的每个图形代码组件中提取信息并置入对应的格式化数据
S604
从第一递归队列中的每个图形代码组件中提取信息并置入对应的格式化数据

图6

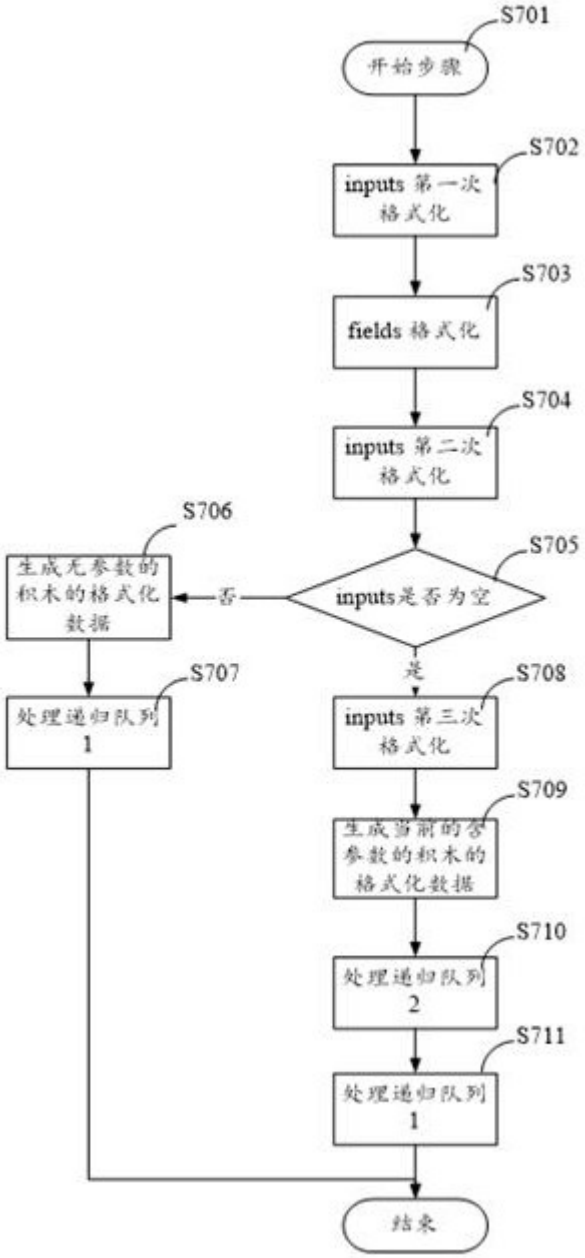

S701
开始步骤
S702
inputs 第一次格式化
S703
fields 格式化
S704
inputs 第二次格式化
S705
inputs是否为空
否
S706
生成无参数的积木的格式化数据
S707
处理递归队列1
是
S708
inputs 第三次格式化
S709
生成当前的含参数的积木的格式化数据
S710
处理递归队列2
S711
处理递归队列1
结束

图7

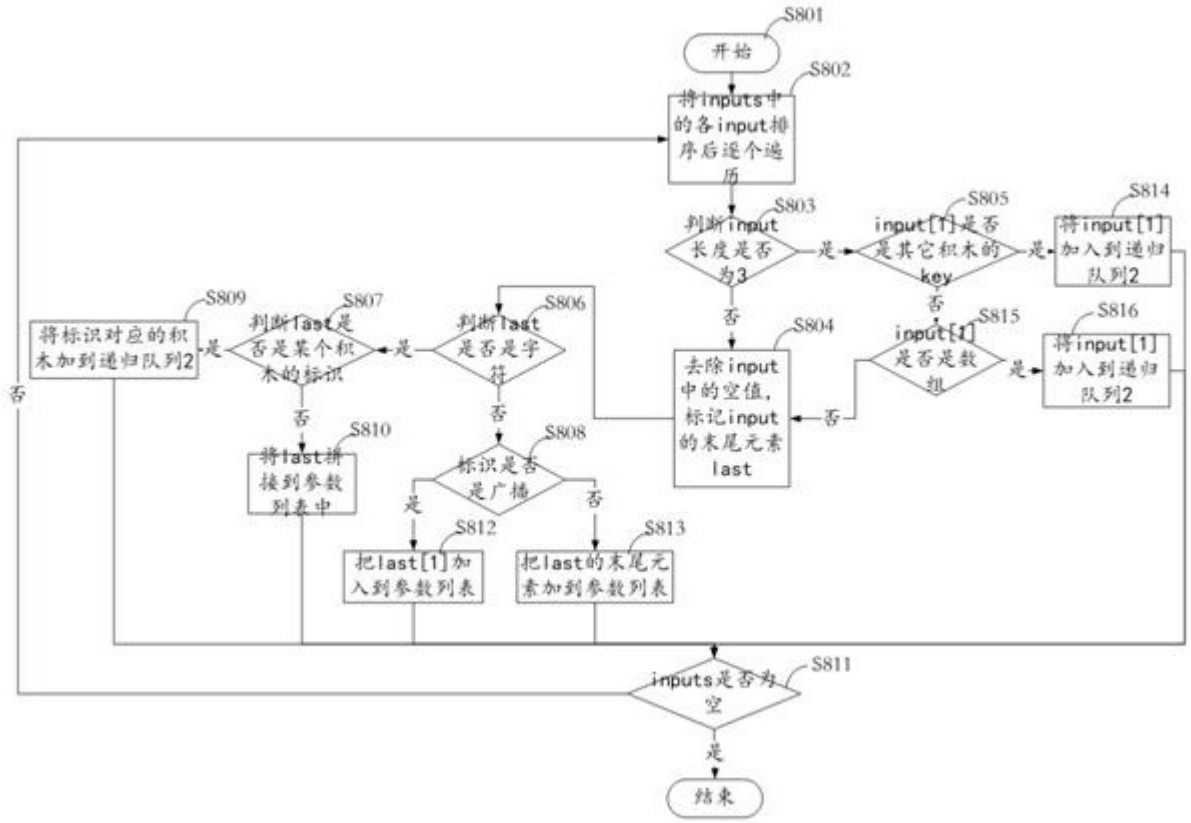

S801
开始
S802
将inputs中的各input排序后逐个遍历
S803
判断input长度是否为3
是
S805
input[1]是否是其它积木的key
是
S814
将input[1]加入到递归队列2
否
S804
去除input中的空值，标记input的末尾元素last
否
S815
input[1]是否是数组
是
S816
将input[1]加入到递归队列2
否
S806
判断last是否是字符
是
S807
判断last是否是某个积木的标识
是
S809
将标识对应的积木加到递归队列2
否
否
S808
标识是否是广播
是
S812
把last[1]加入到参数列表
否
S813
把last的末尾元素加到参数列表
否
S810
将last拼接到参数列表中
S811
inputs是否为空
是
结束

图8

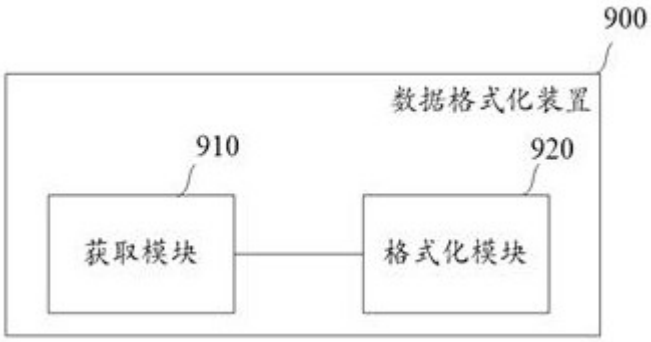

900
数据格式化装置
910
获取模块
920
格式化模块

图9

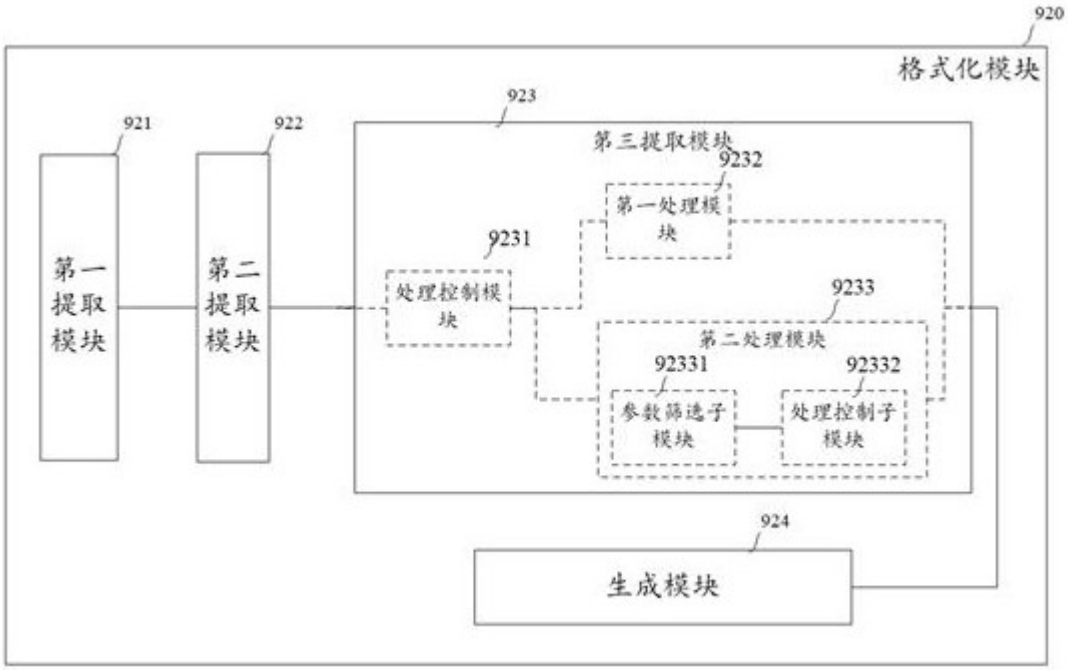

920
格式化模块
921
第一提取模块
922
第二提取模块
923
第三提取模块
9231
处理控制模块
9232
第一处理模块
9233
第二处理模块
92331
参数筛选子模块
92332
处理控制子模块
924
生成模块

图10

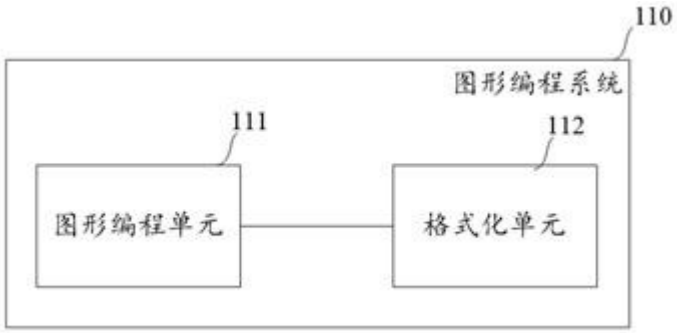

110
图形编程系统
111
图形编程单元
112
格式化单元

图11

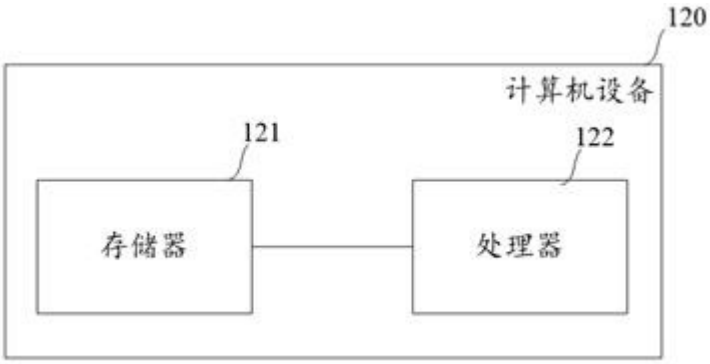

120
计算机设备
121
存储器
122
处理器

图12