

数据处理方法及装置、设备、服务器、系统及存储介质

相关申请的交叉引用

本公开基于申请号为 201910684555.5、申请日为 2019 年 7 月 26 日的中国专利申请提出，并要求该中国专利申请的优先权，该中国专利申请的全部内容在此以全文引入的方式引入本公开。

技术领域

本公开实施例涉及数据处理技术领域，尤其涉及一种数据处理方法及装置、设备、服务器、系统及存储介质。

背景技术

Unity 是游戏开发中最主流的框架之一，不仅仅应用于游戏制作，还可以用于制作 3D（three dimensions，三维）/2D（two dimensions，二维）等丰富多彩的前端界面，并通过万维网图形库（WebGL，World Wide Web Graphics Library）发布在浏览器上。

然而，Python 作为一种动态的，面向对象的脚本语言，用于网页制作（如网页游戏制作）的后端；但在后端使用 Python 的前提下，操控 Unity 前端界面中，由于 Unity 接受的脚本语言为 C sharp；一般地，在用户需要使用 Python 操控游戏的情况下，需要在 Unity 前端中使用插件 IronPython 并编写对应接口。

IronPython 要求动态链接，也就是说 IronPython 只能在 Unity 的编辑版本（editor）的文本中使用；若将 Unity 发布到 WebGL（意味着游戏成为静态环境），则 IronPython 会失效，那么 Python 的操控会失效，致使网页无法正常运行。

发明内容

本公开实施例提供一种数据处理方法及装置、设备、服务器、系统及存储介质。

本公开实施例提供了一种数据处理方法，包括：Python 后端获取数据处理指令；所述 Python 后端根据所述数据处理指令，得到数据处理结果；所述 Python 后端根据所述数据处理结果，生成处理文件，以使得 Unity 前端使用预设接口读取所述处理文件，并根据所述处理文件进行页面显示。这样，有利于 Unity 前端读取 Python 后端的处理文件，能够及时地对 Python 后端获取到的数据处理指令进行响应，从而实现了 Python 后端与 Unity 前端的正常交互，进而实现了包括 Python 后端与 Unity 前端的网页正常运行。

在上述方案中，所述 Python 后端根据所述数据处理指令，得到数据处理结果，包括：所述 Python 后端将所述数据处理指令发送至服务器，以使得所述服务器根据所述数据处理指令得到所述数据处理结果并返回；接收所述服务器返回的所述数据处理结果。通过服务器处理数据处理指令，增加了对数据处理指令进行处理的灵活性。

在上述方案中，所述 Python 后端根据所述数据处理结果，生成处理文件，以使得所述 Unity 前端使用预设接口读取所述处理文件，并根据所述处理文件进行页面显示，包括：所述 Python 后端根据所述数据处理结果，生成被预设脚本语言可识别的处理文件，以使得所述 Unity 前端使用所述预设接口读取所述处理文件，并根据所述处理文件进行页面显示；其中，所述预设接口是采用预设脚本语言编写的。采用预设脚本语言编

写的预设接口,使得 Unity 前端可以从 Python 后端读取处理文件,有助于响应数据处理指令,使得 Unity 前端与 Python 后端能够交互,实现网页的正常运行。

在上述方案中,所述预设接口是采用 Javascript (JS,脚本语言)或 HTML5 脚本语言编写的。可以采用多种类型的脚本语言来编写预设接口,增加了预设接口的种类,方便开发。

在上述方案中,所述处理文件包括结果处理成功或失败的标识,以及界面动作序列。通过得到不同数据类型的处理文件,可以更加准确地响应数据处理指令,以达到网页正常运行的目的。

在上述方案中,所述 Python 后端获取数据处理指令,包括:所述 Python 后端根据接收到的 Python 代码,生成所述数据处理指令。这里采用 Python 代码可以生成数据处理指令,使得开发人员方便进行网页编辑或测试,以完善产品的性能。

本公开实施例提供了一种数据处理方法,包括:Unity 前端使用预设接口读取处理文件;其中,所述处理文件是 Python 后端根据数据处理结果生成的结果文件,所述数据处理结果是所述 Python 后端根据获取到的数据处理指令得到的;所述 Unity 前端根据所述处理文件进行页面显示。这样,有利于 Unity 前端读取 Python 后端的处理文件,能够及时地从 Python 后端读取处理文件,从而实现了 Python 后端与 Unity 前端的正常交互,进而实现了包括 Python 后端与 Unity 前端的网页正常运行。

在上述方案中,所述 Unity 前端根据所述处理文件进行页面显示,包括:所述 Unity 前端响应于所述处理文件中包括结果处理成功的标识,根据所述处理文件中的界面动作序列刷新页面。对处理成功的处理文件,通过界面动作序列刷新页面,使得页面能够显示处理成功的处理文件,以有利于页面的正常运行。

在上述方案中,所述 Unity 前端根据所述处理文件进行页面显示,包括:所述 Unity 前端响应于所述处理文件中包括结果处理失败的标识,生成错误信息,以提示所述数据处理指令操作失败。对处理失败的处理文件,生成错误信息以起到提示作用,使得页面能够及时地告知用户数据处理指令操作失败。

在上述方案中,所述方法还包括:所述 Unity 前端使用预设接口向所述 Python 后端传输当前页面状态文件,以使得所述 Python 后端向服务器发送所述当前页面状态文件,以使得所述服务器根据所述当前页面状态文件和所述数据处理指令得到所述数据处理结果。通过预设接口还可以传输当前页面状态文件,以使得 Unity 前端可以得到更多的 Python 后端的信息,以更好的进行页面显示。

在上述方案中,在所述 Unity 前端使用预设接口向所述 Python 后端传输当前页面状态文件之前,所述方法还包括:所述 Unity 前端根据当前页面状态,生成所述当前页面状态文件。Unity 前端还可以根据当前页面状态生成当前页面状态文件,以得到更加准确有效的数据处理结果。

在上述方案中,所述方法还包括:所述 Unity 前端使用预设接口读取初始化文件;所述 Unity 前端根据所述初始化文件进行页面初始化,得到初始页面并显示。Unity 前端通过预设接口还可以实现页面初始化的功能,加强了 Unity 前端的显示效果。

在上述方案中,所述 Unity 前端使用预设接口读取的处理文件,包括:所述 Unity 前端响应接收到的对所述处理文件的显示指令,使用预设接口读取所述处理文件;或者,所述 Unity 前端按照预设周期使用预设接口读取所述处理文件。Unity 前端可以通过多种方式使用预设接口来读取处理文件,增加了读取处理文件的灵活性,有助于及时地读取处理文件。

在上述方案中，所述预设接口是采用 Javascript 或 HTML5 脚本语言编写的。可以采用多种类型的脚本语言来编写预设接口，增加了预设接口的种类，方便开发。

本公开实施例提供一种数据处理方法，该方法应用于一服务器中，所述方法包括：接收到来自 Python 后端的数据处理指令；根据所述数据处理指令，得到数据处理结果；  
5 将所述数据处理结果发送至所述 Python 后端，以使得所述 Python 后端根据所述数据处理结果，生成处理文件，以使得 Unity 前端使用预设接口读取所述处理文件，并根据所述处理文件进行页面显示。这样，通过服务器来处理数据处理指令得到数据处理结果，增加了对数据处理指令进行处理的灵活性，为 Python 后端分担更多的处理任务，提高  
10 处理效率，并且，有利于 Unity 前端读取 Python 后端的处理文件，能够及时地对 Python 后端获取到的数据处理指令进行响应，从而借助服务器实现了 Python 后端与 Unity 前端的正常交互，进而实现了包括 Python 后端与 Unity 前端的网页正常运行。

在上述方案中，所述根据所述数据处理指令，得到数据处理结果，包括：接收到来自所述 Python 后端的当前页面状态文件；根据所述当前页面状态文件和所述数据处理指令得到所述数据处理结果；其中，所述当前页面状态文件是所述 Unity 前端使用预设  
15 接口向所述 Python 后端传输的。通过接收到的当前页面状态文件，使得服务器能够根据当前页面状态文件来处理数据处理指令，以得到数据处理结果，增加了处理数据处理指令的准确性。

在上述方案中，所述根据所述当前页面状态文件和所述数据处理指令得到所述数据处理结果，包括：判断所述数据处理指令对所述当前页面状态文件的处理是否符合预设  
20 条件；在所述数据处理指令对所述当前页面状态文件的处理符合预设条件的情况下，根据所述数据处理指令，得到结果处理成功的数据处理结果，以使得所述 Python 后端根据所述数据处理结果，生成结果处理成功的处理文件，以使得所述 Unity 前端使用预设接口读取所述结果处理成功的处理文件，并根据所述结果处理成功的处理文件对页面进行刷新。通过对数据处理指令对当前页面状态文件的处理是否符合预设条件的判断，针  
25 对符合预设条件的得到结果处理成功的处理文件，能够更加准确的得到数据处理结果，使得 Unity 前端能够显示出正确的处理结果。

在上述方案中，在判断所述数据处理指令对所述当前页面状态文件的处理是否符合预设条件之后，所述方法还包括：在所述数据处理指令对所述当前页面状态文件的处理不符合预设条件的情况下，根据所述数据处理指令，得到结果处理失败的数据处理结果，  
30 以使得所述 Python 后端根据所述数据处理结果，生成结果处理失败的处理文件，以使得所述 Unity 前端使用预设接口读取结果处理失败的处理文件，并根据结果处理失败的处理文件生成错误信息，以提示所述数据处理指令操作失败。通过对数据处理指令对当前页面状态文件的处理是否符合预设条件的判断，针对不符合预设条件的得到结果处理失败的处理文件，能够更加准确的得到数据处理结果，使得 Unity 前端能够显示出正确的  
35 的处理结果。

在上述方案中，所述处理文件中包括结果处理成功或失败的标识，以及界面动作序列。通过得到不同数据类型的处理文件，可以更加准确地响应数据处理指令，以达到网页正常运行的目的。

在上述方案中，所述预设接口是采用 Javascript 或 HTML5 脚本语言编写的。可以采用多种类型的脚本语言来编写预设接口，增加了预设接口的种类，方便开发。  
40

本公开实施例提供了一种数据处理装置，包括：获取模块，配置为获取数据处理指令；第一确定模块，配置为根据所述数据处理指令，得到数据处理结果；第一生成模块，配置为根据所述数据处理结果，生成处理文件，以使得 Unity 前端使用预设接口读取所

述处理文件，并根据所述处理文件进行页面显示。

本公开实施例还提供了一种数据处理装置，包括：读取模块，配置为使用预设接口读取处理文件；其中，所述处理文件是 Python 后端根据数据处理结果生成的结果文件，所述数据处理结果是所述 Python 后端根据获取到的数据处理指令得到的；显示模块，配置为根据所述处理文件进行页面显示。

本公开实施例还提供了一种数据处理装置，所述装置包括：接收模块，配置为接收到来自 Python 后端的数据处理指令；第二确定模块，配置为根据所述数据处理指令，得到数据处理结果；发送模块，配置为将所述数据处理结果发送至所述 Python 后端，以使得所述 Python 后端根据所述数据处理结果，生成处理文件，以使得 Unity 前端使用预设接口读取所述处理文件，并根据所述处理文件进行页面显示。

本公开实施例还提供了一种网页处理设备，所述网页处理设备包括：处理器、存储器和通信总线；其中，所述通信总线，配置为实现所述处理器和所述存储器之间的连接通信；所述处理器，配置为执行所述存储器中存储的数据处理程序，以实现上述一个或多个实施例中 Python 后端执行的所述的数据处理方法，和/或，上述一个或多个实施例中 Unity 前端执行的所述的数据处理方法。

本公开实施例提供了一种服务器，所述服务器包括：处理器、存储器和通信总线；其中，所述通信总线，配置为实现所述处理器和所述存储器之间的连接通信；所述处理器，配置为执行所述存储器中存储的数据处理程序，以实现上述一个或多个实施例中服务器执行的所述的数据处理方法。

本公开实施例提供一种数据处理系统，所述系统包括上述一个或多个实施例中的网页处理设备，以及实现上述一个或多个实施例中的服务器。

本公开实施例提供了一种计算机可读存储介质，所述计算机可读存储介质存储有一个或者多个程序，所述一个或者多个程序可以被一个或者多个处理器执行，以实现上述一个或多个实施例中 Python 后端执行的数据处理方法，或者实现上述一个或多个实施例中 Unity 前端执行的数据处理方法，或者实现上述一个或多个实施例中服务器执行的所述的数据处理方法。

本公开实施例提供了一种计算机程序，包括计算机可读代码，当所述计算机可读代码在电子设备中运行时，所述电子设备中的处理器执行用于实现上述一个或多个实施例中 Python 后端执行的数据处理方法，或者实现上述一个或多个实施例中 Unity 前端执行的数据处理方法，或者实现上述一个或多个实施例中服务器执行的所述的数据处理方法。

由此可见，在本公开实施例提供的一种数据处理方法及装置、设备、服务器、系统及存储介质，该方法包括：首先，Python 后端获取数据处理指令，然后，Python 后端根据数据处理指令，得到数据处理结果，最后，Python 后端根据数据处理结果，生成处理文件，以使得 Unity 前端使用预设接口读取处理文件，并根据处理文件进行显示；也就是说，在本公开实施例中，当 Python 后端获取到的数据处理指令之后，先对数据处理指令进行处理得到数据处理结果，再根据数据处理结果生成处理文件，这样，使得 Unity 前端可以使用预设接口来读取处理文件，有利于 Unity 前端读取 Python 后端的处理文件，能够及时地对 Python 后端获取到的数据处理指令进行响应，从而实现了 Python 后端与 Unity 前端的正常交互，进而实现了包括 Python 后端与 Unity 前端的网页正常运行。

应当理解的是，以上的一般描述和后文的细节描述仅是示例性和解释性的，而非限

制本公开。

#### 附图说明

此处的附图被并入说明书中并构成本说明书的一部分，这些附图示出了符合本公开的实施例，并与说明书一起用于说明本公开的技术方案。

- 5       图 1 为本公开实施例提供的一种数据处理方法的流程交互示意图；  
      图 2a 为本公开实施例提供的一种数据处理方法的实例的流程示意图；  
      图 2b 为本公开实施例提供的另一种数据处理方法的实例的流程示意图；  
      图 3 为本公开实施例提供的一种数据处理方法的流程示意图；  
      图 4 为本公开实施例提供的另一种数据处理方法的流程示意图；  
10      图 5 为本公开实施例提供的再一种数据处理方法的流程示意图；  
      图 6 为本公开实施例提供的一种数据处理装置的结构示意图；  
      图 7 为本公开实施例提供的另一种数据处理装置的结构示意图；  
      图 8 为本公开实施例提供的再一种数据处理装置的结构示意图；  
      图 9 为本公开实施例提供的一种网页处理设备的结构示意图；  
15      图 10 为本公开实施例提供的另一种网页处理设备的结构示意图；  
      图 11 为本公开实施例提供的一种服务器的结构示意图；  
      图 12 为本公开实施例提供的一种数据处理系统的结构示意图。

#### 具体实施方式

- 20       下面将结合本公开实施例中的附图，对本公开实施例中的技术方案进行清楚、完整地描述。

- 本公开实施例提供了一种数据处理方法，该数据处理方法可应用于一数据处理系统中，其中，该数据处理系统可以包括交互设备和服务器，交互设备可以包括 Unity 前端和 Python 后端；其中，Unity 前端具有强大的图形功能，可以用于交互设备的交互界面的设计和显示，Unity 前端还具有各种输出选项，可以支持任何平台的触摸屏表面；Python  
25      后端能够进行数据挖掘，科学计算，图像处理，人工智能等等数据处理，可以用于根据交互设备输入的 Python 代码进行数据处理，可以实现编辑 Unity 前端显示的页面的目的。

- 该数据处理系统可用于网页制作，例如网页游戏开发。下面以网页游戏为例进行说明。这里，Python 后端分别与服务器和 Unity 前端建立有通信连接，使得 Python 后端可以用于从服务器和 Unity 前端读取数据，还可以用于向服务器和 Unity 前端发送数据；  
30      服务器用于处理 Python 后端接收到的数据处理请求，Unity 前端用于前端显示，如显示网页游戏的界面。

      为了使得网页游戏能够正常运行，本公开实施例提供一种数据处理方法，该方法应用于上述数据处理系统中，图 1 为本公开实施例提供的一种数据处理方法的流程交互示意图，如图 1 所示，该数据处理方法可以包括：

- 35       S101: Python 后端 101 获取数据处理指令；

      目前，网页游戏作为一种休闲娱乐的方式，广受大众的追捧，一般地，利用 Unity 这一语言工具来制作网页游戏的前端显示，以实现网页游戏的前端显示界面，利用 Python 这一语言工具来实现网页游戏的后端逻辑操控，以操控网页游戏，即所有编程逻辑

辑都可以通过 Python 后端完成,在 Python 后端操作从 Unity 前端游戏逻辑中剥离,Unity 前端可以仅起显示作用;当使用 Unity 前端 102 和 Python 后端 101,则需要在 Unity 前端 102 中使用插件 IronPython 编写而对应的接口,使得 Unity 前端 102 能够解析并处理接收到的数据处理请求。

- 5       然而,由于插件 IronPython 的动态链接特性,使其只能应用于 Unity 前端 102 的编辑状态,所以,在将游戏发布至网页上的情况下插件 IronPython 失效,使得网页游戏无法正常运行。

10       为了实现该网页游戏的正常运行,首先,Python 后端 101 获取数据处理指令,在实际应用中,当网页游戏发布后,开发人员仍然可以使用 Python 后端 101 对网页游戏的界面进行操作,从而生成数据处理指令,该数据处理指令可以为对预设按钮的点击操作生成的数据处理指令,也可以为该数据处理指令可以为对特定区域的点击操作生成的数据处理指令,或者根据输入的代码自动生成对应的数据处理指令;并且,该数据处理指令可以为重启或者关闭网页游戏的指令,也可以为网页游戏中选取角色,选取皮肤,设置角色属性等等的数据处理指令;这里,本公开实施例对此不作具体限定。

- 15       为了获取到的数据处理指令,在本公开的一些实施例中,S101 可以包括:Python 后端 101 根据接收到的 Python 代码,生成数据处理指令。

20       一般来说,开发人员或用户可以通过输入设备输入 Python 代码,例如用户在浏览器的代码输入窗口输入一段 Python 代码,Python 后端 101 可以接收到该 Python 代码,从而 Python 后端 101 可以根据 Python 代码生成数据处理指令,例如,该 Python 代码可以作为一个 Python 运算表达式,生成的数据处理指令为携带有该 Python 运算表达式的指令。

S102: Python 后端 101 将数据处理指令发送至服务器 103;

25       Python 后端 101 在获取到数据处理指令之后,可以将该数据处理指令打包发送给服务器 103,例如根据 Http 协议(HyperText Transfer Protocol,超文本传输协议)对该数据处理指令进行打包加密形成数据包发送至服务器 103。其中,Python 后端 101 和服务器 103 之间也可以使用 FTP 协议(File Transfer Protocol,文件传输协议)、SOAP 协议(Simple Object Access Protocol,简单对象访问协议)或 TELNET(远程登录)协议等实现通信。

30       在本公开的一些实施例中,Python 后端 101 也可以自行根据获取到的数据处理指令进行处理,得到数据处理结果;这里,本公开实施例对此不作具体限定。

S103: 服务器 103 根据数据处理指令,得到数据处理结果;

35       在实施过程中,通常地,为了减少 Python 后端 101 的负担,将数据处理指令发送至服务器 103,使得服务器 103 对数据处理指令进行处理得到数据处理结果;例如,服务器 103 和 Python 后端 101 之间采用 Http 协议进行通信,服务器 103 可以接收 Python 后端 101 发送的数据包后,对数据包进行解析解密,得到该数据处理指令,在数据处理指令中为具有 Python 运算表达式的指令的情况下,服务器 103 的 Python 处理器对 Python 运算表达式进行计算得到数据处理结果。

40       为了得到更加准确的数据处理结果,在本公开的一些实施例中,S103 可以包括:服务器 103 接收到来自 Python 后端 101 的当前页面状态文件;服务器 103 根据当前页面状态文件和数据处理指令得到数据处理结果;其中,当前页面状态文件是 Unity 前端 102 使用预设接口向 Python 后端 101 传输的;其中,该预设接口为采用预设的脚本语言编写的接口。

一般来说,服务器 103 在接收到来自 Python 后端 101 的当前页面状态文件,该当前页面状态文件用于反映当前页面的界面信息,包括可修改的属性值,可修改的目标等等;当前页面状态文件是 Unity 前端 102 使用预设接口向 Python 后端 101 传输的,这样,可以得知当前页面的状态信息,然后,服务器 103 可以根据当前页面状态文件和数据处  
5 理指令来得到数据处理结果,如此,使得数据处理结果不仅仅根据数据处理指令得到的,还结合当前页面的状态信息得到数据处理结果,使得数据处理结果更加准确有效。

为了得到更加准确的数据处理结果,在本公开的一些实施例中,根据当前页面状态文件和数据处理指令得到数据处理结果,可以包括:服务器 103 判断数据处理指令对当前页面状态文件的处理是否符合预设条件;在数据处理指令对当前页面状态文件的处理  
10 符合预设条件的情况下,根据数据处理指令,得到结果处理成功的数据处理结果,以使得 Python 后端 101 根据数据处理结果,生成结果处理成功的处理文件,以使得 Unity 前端 102 使用预设接口读取结果处理成功的处理文件,并根据结果处理成功的处理文件对页面进行刷新;在数据处理指令对当前页面状态文件的处理不符合预设条件的情况下,根据数据处理指令,得到结果处理失败的数据处理结果,以使得 Python 后端 101 根据  
15 数据处理结果,生成结果处理失败的处理文件,以使得 Unity 前端 102 使用预设接口读取结果处理失败的处理文件,并根据结果处理失败的处理文件生成错误信息,以提示数据处理指令操作失败。

一般来说,服务器 103 先根据数据处理指令对当前页面状态文件进行处理,并判断  
20 数据处理指令对当前页面状态文件的处理是否符合预设条件,也就是说,检查数据处理指令是否能够对当前页面状态文件进行处理,若能够处理,服务器 103 得到结果处理成功的数据处理结果,若不能够处理,服务器 103 得到结果处理失败的数据处理结果。例如,该数据处理指令需要修改界面中的目标人物 A 的属性,而当前页面状态文件中并不存在目标人物 A 或并不存在需要修改的属性、或该修改的属性值超过允许修改范围,则服务器 103 会判定该数据处理指令不能够对当前页面状态文件进行处理,输出结果处理  
25 失败的数据处理结果。

那么,针对结果处理成功的数据处理结果,使得 Python 后端 101 可以得到结果处理成功的处理文件,进而使得 Unity 前端 102 读取结果处理成功的处理文件,以使用预设接口读取结果处理成功的处理文件,通过页面刷新显示处理文件;针对结果处理失败的数据处理结果,使得 Python 后端 101 可以得到结果处理失败的处理文件,进而使得  
30 Unity 前端 102 读取结果处理失败的处理文件,生成错误信息,以提示数据处理指令操作失败,其中,结果处理失败可能是需要修改的属性值不符合要求或者修改的目标不存在。

S104: 服务器 103 将数据处理结果返回至 Python 后端 101;

在一种可能的实现方式中,服务器 103 会将数据处理结果返回给 Python 后端 101,  
35 例如将数据处理结果根据 Http 协议进行加密打包后发送给 Python 后端 101,以使得 Python 后端 101 可以接收到该数据处理结果。

S105: Python 后端 101 根据数据处理结果,生成处理文件;

这里,服务器 103 将数据处理结果返回至 Python 后端 101,Python 后端 101 生成可  
40 被 Unity 前端 102 成功识别的处理文件。例如,Python 后端 101 获取 Http 数据包后,对数据包进行解密解析,得到该数据处理结果,然后使用预先设定的脚本语言,如 Javascript,生成可被预设接口读取的处理文件,如 json(JavaScript Object Notation, JS 对象简谱)格式文件。这样,使得 Unity 前端 102 使用预设接口读取自身可识别的处理文件,从而能够在网页游戏上对处理文件进行显示,即响应用户的操作,使得网页游戏可以正常运行。

为了实现网页游戏的正常运行,在本公开的一些实施例中,S105可以包括:Python后端101解析数据处理结果,生成被预设脚本语言可识别的处理文件,以使得Unity前端102使用预设接口读取所述处理文件,并根据处理文件进行页面显示;

- 5 这里,为了使得Python后端101得到的处理文件能够被Unity前端102读取到,Python后端101在解析数据处理结果后,生成能够被预设脚本语言可识别的处理文件,这样,可以使得Unity前端102通过预设接口从Python后端101读取处理文件并进行页面显示。

- 10 其中,预设接口是采用预设脚本语言编写的,预设接口为Unity前端102与外部进行交互的功能函数,其可实现识别并读取Python后端101生成的处理文件的功能。这样,使得Unity前端102通过预设接口能够从Python后端101读取被预设脚本语言可识别的处理文件,有利于网页游戏的正常运行。

- 15 在本公开的一些实施例中,上述预设接口是采用Javascript或HTML5(Hyper Text Markup Language 5,第5版本的超文本标记语言)脚本语言编写的;这样,只需要根据实际应用场景定义所需的中间协议就可以实现:Python后端101生成符合中间协议的处理文件,Unity前端102通过符合中间协议的预设接口从Python后端101即可识别并读取该处理文件,其中中间协议规定了Python后端101和Unity前端102之间通信的规则,例如规定Unity前端102对外的预设接口和Python后端101生成的处理文件的格式等。

- 20 其中,上述处理文件包括结果处理成功或者失败的标识,以及界面动作序列。也就是说,处理文件的内容不仅仅包括处理结果还包括动作序列,动作序列是一种封装多个动作的对象,在这个对象执行的情况下,被封装的动作会顺序执行。处理文件可以为结果处理成功的标识或者为结果处理失败的标识,还包括结果处理成功的界面动作序列和结果处理失败的界面动作序列,这样,使得Unity前端102可以根据处理结果标识判定是否需要根据界面动作序列更新当前页面。

- 25 S106:Unity前端102使用预设接口读取处理文件;Python后端101在得到处理文件之后,Unity前端102可使用预设接口读取处理文件,在实施过程中,S106可以包括:Unity前端102响应接收到的对处理文件的显示指令,使用预设接口读取处理文件;或者,Unity前端102按照预设周期使用预设接口读取处理文件。

- 30 一般来说,Unity前端是一个前端显示工具,可以定时或按照用户的指示从Javascript处读取状态参数等,更新自己的状态并将变化的过程显示在平台上;Unity前端102可以在接收到用户对于处理文件的显示指令之后,立即去使用预设接口读取处理文件,或者,Unity前端102按照预设周期,例如每隔1s使用预设接口读取处理文件,这里,本公开实施例对此不作具体限定。

- S107:Unity前端102根据处理文件进行页面显示。

- 35 Unity前端102在读取到处理文件之后,要根据处理文件进行页面显示,以向用户反馈对输入的Python代码的处理结果。

为了实现对处理文件的页面显示,在本公开的一些实施例中,S107可以包括:Unity前端102响应于处理文件为结果处理成功,通过刷新页面将处理文件进行页面显示。

- 40 这里,在处理文件为结果处理成功的情况下,此时,可以通过页面刷新的方式将处理文件显示在页面上,以供用户查看。

为了实现对处理文件的页面显示,在本公开的一些实施例中,S107可以包括:Unity

前端 102 响应于处理文件中包括结果处理失败的标识,生成错误信息,以提示数据处理指令操作失败。

这里,在处理文件为结果处理失败的情况下,此时,可以生成错误信息,用错误信息提示用户处理文件为结果处理失败,这样,使得用户知晓对数据处理指令的处理结果。

- 5 在其他实施例中,Unity 前端 102 在处理文件为结果处理失败的情况下,可以直接不更新当前页面,和/或显示更新失败的信息。

可见,在网页游戏接收到数据处理指令的情况下,得到的处理文件可以为结果处理成功或者结果处理失败,Unity 前端 102 可以根据读取到的处理文件进行不同的页面显示,以反馈给用户。

- 10 为了实现 Unity 前端 102 使用预设接口向 Python 后端 101 传输当前页面状态文件,需要 Unity 前端 102 先得到当前页面状态文件,为了得到当前页面状态文件,在本公开的一些实施例中,在 Unity 前端 102 使用预设接口向 Python 后端 101 传输当前页面状态文件之前,该方法还包括:Unity 前端 102 根据当前页面状态,生成当前页面状态文件。

- 15 一般来说,Unity 前端 102 先获取当前页面状态,然后再根据当前页面状态生成当前页面状态文件,从而可以将当前页面状态文件传输至 Python 后端 101,使得 Python 后端 101 将当前页面状态文件发送至服务器 103。

另外,Unity 前端 102 还可以进行初始化,在本公开的一些实施例中,该方法还可以包括:Unity 前端 102 使用预设接口读取初始化文件;Unity 前端 102 根据初始化文件进行页面初始化,得到初始页面并显示。

- 20 该预设接口为 Unity 前端 102 与外部(如服务器 103 和 Python 后端 101)进行交互的接口,如其可以实现与服务器 103 和 Python 后端 101 的信令和/或数据信息的传输功能。

- 25 一般来说,当服务器 103 通过 Python 后端 101 接收到用户 Python 代码,服务器 103 通过预设接口告知 Unity 前端 102 进行初始化,Unity 前端 102 使用相同的预设接口读取初始化文件,然后,根据初始化文件进行页面初始化,从而得到初始页面并显示,这样,使得 Unity 前端 102 可以在初始化页面的基础上对处理文件进行页面显示。

相关技术中,引入 Unity 技术作为前端,使用到网页开发库,在 Unity 前端中使用 IronPython 插件;本公开实施例中,Python 后端与 Unity 前端独立、功能不受限地开发,通过 Javascript 通信。

- 30 在本公开实施例中,Unity 前端平台制作的前端交互力更强,Python 后端与 Unity 前端交互的程序依赖性要求变小,只需要定义不同的中间协议就可以适应各种功能需求。

下面举实例来对上述一个或多个实施例中所述的数据处理方法进行说明。

图 2a 为本公开实施例提供的一种数据处理方法的实例的流程示意图,如图 2a 所示,在网页游戏的设计过程中,可以执行以下步骤:

- 35 首先,定义 Unity 前端 202a 对外的 Javascript 接口,以供在 Unity 开发环境外操控 Unity 前端 202a,将网页游戏中的游戏设置 204a 与 Unity 前端 202a 中的接口互相绑定,其中,绑定的接口可以通过 Javascript 编写的接口来访问,例如,在实际应用中,游戏设置 204a 可以包括:地图,血量,朝向和成功机制等等。

- 40 然后,定义 Python 后端 201a 与 Unity 前端 202a 通过 Javascript 的通信协议,在 Python 后端 201a 上编写代码,使得 Python 后端 201a 可以通过 Javascript 实现 Unity 前端 202a 与服务器 203a 之间的数据中转,例如,Python 后端 201a 获取数据处理指令后,将数据

处理指令通过 Http 协议发送至服务器 203a。

其次, 服务器 203a 实现对数据处理指令的处理; 例如, 在实际应用中, 服务器 203a 可以处理 Python 后端 201a 发送来的数据处理指令, 得到数据处理结果。

最后, Python 后端 201a 接收到数据处理结果, 生成处理文件, 例如, 所述数据处理结果可以包括结果处理成功和结果处理失败, 所述数据处理结果为结果处理成功时, 所述处理文件可以包括成功判别 (成功页面提示) 205a 和动作序列 206a, 所述处理文件可以是对地图, 位置, 朝向和能量的文件; Unity 前端 202a 使用 Javascript 编写的接口读取该处理文件, 例如, Unity 前端 202a 开放一个“刷新”接口给 Javascript 访问, Javascript 将这个接口绑定至某个游戏网页按钮, 用户点击该游戏网页按钮即可刷新游戏界面。

基于上述图 2a 建立好上述网页游戏之后, 该数据处理方法可以包括: 服务器 203a 接收到来自 Python 后端 201a 读取到的 Python 请求, 处理 Python 请求得到 Python 结果, 并通过 Javascript 告知 Unity 前端 202a 进行初始化, Python 后端 201a 通过 Javascript 读取 Python 结果, 将 Python 结果发送至 Unity 前端 202a, Unity 前端 202a 执行 Python 结果。

一般来说, 当用户点击游戏页面的重启按钮, 此时 Python 后端接收到一个 Python 请求, 其中, 该 Python 请求用于重启游戏, 此时, Javascript 可以访问该“重启”接口, 触发 Python 后端 201a 调用 Javascript 的读取指令读取 Python 请求, 并调用 Javascript 的发送指令发送 Python 请求至服务器 203a 的 Python 处理器; 其中, Python 请求可以作为一个 Python 运算表达式。

Python 处理器处理 Python 请求得到 Python 结果, Python 后端 201a 从服务器 203a 读取 Python 结果, 并将 Python 结果发送至 Unity 前端 202a, Unity 前端 202a 实现网页游戏重启。

图 2b 为本公开实施例提供的一种数据处理方法的实例的流程示意图, 如图 2b 所示, 在网页游戏的设计过程中, 可以执行以下步骤:

首先, 定义 Unity 前端 202b 对外的 Javascript 接口, 以供在 Unity 开发环境外操控 Unity 前端 202b, 将网页游戏中的游戏设置 204b 与 Unity 前端 202b 中的接口互相绑定, 其中, 绑定的接口可以通过 Javascript 编写的接口来访问, 例如, 在实际应用中, 游戏设置 204b 可以包括: 地图, 血量, 朝向和成功机制等等。

然后, 定义 Python 后端 201b 与 Unity 前端 202b 通过 Javascript 的通信协议, 在 Python 后端 201b 上编写代码, 使得 Python 后端 201b 可以通过 Javascript 实现 Unity 前端 202b 与服务器 203b 之间的数据中转, 例如, Python 后端 201b 获取数据处理指令后, 将数据处理指令通过 Http 协议发送至服务器 203b。

其次, 服务器 203b 实现对数据处理指令的处理; 例如, 在实际应用中, 服务器 203b 可以处理 Python 后端 201b 发送来的数据处理指令, 得到数据处理结果, 并根据数据处理结果, 生成处理文件, 例如, 所述数据处理结果可以包括结果处理成功和结果处理失败, 所述数据处理结果为结果处理成功时, 所述处理文件可以包括成功判别 (成功页面提示) 205b 和动作序列 206b, 所述处理文件可以是对地图, 位置, 朝向和能量的文件; Unity 前端 202b 使用 Javascript 编写的接口读取该处理文件, 例如, Unity 前端 202b 开放一个“刷新”接口给 Javascript 访问, Javascript 将这个接口绑定至某个游戏网页按钮, 用户点击该游戏网页按钮即可刷新游戏界面。

最后, Unity 前端使用 Javascript 接口读取所述处理文件, 并根据所述处理文件进行

页面显示。

基于上述图 2b 建立好上述网页游戏之后, 该数据处理方法可以包括:

服务器 203b 接收到来自 Python 后端 201b 读取到的 Python 请求, 处理 Python 请求得到 Python 结果, 并通过 Javascript 告知 Unity 前端 202b 进行初始化, 服务器 203b 将 Python 结果发送至 Unity 前端 202b, Unity 前端 202b 执行 Python 结果。

在本公开实施例提供的一种数据处理方法中, 参见图 2b, 在网页游戏的设计过程中, 还可以执行以下步骤:

步骤 301: 后端所有逻辑使用 Python 语言开发;

步骤 302: 定义 Python 后端与 Javascript 通信的协议: 比如: 后端运行输出一个特定格式的 json 文件, Javascript 在用户点击某按钮时读取文件, 与 Unity 前端交互;

步骤 303: 定义 Unity 前端对外的 Javascript 接口, 以供在 Unity 开发环境外操纵前端。

其中, 步骤 302 中的协议可以示例为如下步骤, 这是一个学生编码产品使用本公开实施例构建的协议架构:

步骤 3021: 服务器接收用户的输入操作;

步骤 3022: 服务器通过 JS 告知 Unity 前端进行初始化并根据输入运行 Python 后端;

步骤 3023: 服务器结束 Python 后端的运行之后, Python 后端输出可以被 JS 读取的运行结果; JS 指令告知 Unity 前端以运行结果为输入, 执行特定的操作。

用户可以利用本架构建立 Python 后端与 Unity 前端的系统, 通过定义不同的协议, Python 后端与 Unity 前端的交互可以根据需求变化。

本公开实施例还可以应用在教育机器人领域。

在本公开实施例提供的一种数据处理方法, 该方法应用于一网页处理设备中, 该处理设备包括 Python 后端, 该方法可以包括: Python 后端获取数据处理指令, Python 后端根据数据处理指令, 得到数据处理结果, Python 后端根据数据处理结果, 生成处理文件, 以使得 Unity 前端使用预设接口读取处理文件, 并根据处理文件进行页面显示。上述方法的实施例过程可参考上述实施例的内容, 此处不再重复。

在本公开实施例中, Python 后端根据获取到的数据处理指令得到数据处理结果, 并解析数据处理结果生成处理文件, 这样, Unity 前端可以使用预设接口读取处理文件并显示, 这样, 有利于网页游戏中 Unity 前端处理数据处理请求, 能够及时地对数据处理指令进行响应, 从而实现了网页游戏的正常运行。

下面以数据处理系统中所部属的各个设备侧对上述数据处理方法进行说明。

首先, 以 Python 后端侧对数据处理方法进行描述。

本公开实施例提供一种数据处理的方法, 图 3 为本公开实施例提供的一种数据处理方法的流程示意图, 参考图 3 所示, 该数据处理方法可以包括:

S301: Python 后端获取数据处理指令;

S302: Python 后端根据数据处理指令, 得到数据处理结果;

S303: Python 后端根据数据处理结果, 生成处理文件, 以使得 Unity 前端使用预设接口读取所述处理文件, 并根据处理文件进行页面显示。

在本公开的一些实施例中, S302 可以包括: Python 后端将数据处理指令发送至服

务器,以使得服务器根据数据处理指令得到数据处理结果并返回;接收服务器返回的数据处理结果。

在本公开的一些实施例中,S303 可以包括:Python 后端根据数据处理结果,生成被预设脚本语言可识别的处理文件,以使得 Unity 前端使用预设接口读取处理文件,并根据处理文件进行页面显示;其中,预设接口是采用预设脚本语言编写的。

其中,预设接口是采用 Javascript 或 HTML5 脚本语言编写的,处理文件包括结果处理成功或失败的标识,以及界面动作序列。

在本公开的一些实施例中,S301 可以包括:Python 后端根据接收到的 Python 代码,生成数据处理指令。

其次,以 Unity 前端侧对数据处理方法进行描述。

本公开实施例提供一种数据处理的方法,图 4 为本公开实施例提供的另一种数据处理方法的流程示意图,如图 4 所示,该数据处理方法可以包括:

S401: Unity 前端使用预设接口读取处理文件;

其中,处理文件是 Python 后端根据数据处理结果生成的结果文件,数据处理结果是 Python 后端根据获取到的数据处理指令得到的;

S402: Unity 前端根据处理文件进行页面显示。

在本公开的一些实施例中,S402 可以包括:Unity 前端响应于处理文件中包括结果处理成功的标识,根据处理文件中的界面动作序列刷新页面。

在本公开的一些实施例中,S402 可以包括:Unity 前端响应于处理文件中包括结果处理失败的标识,生成错误信息,以提示数据处理指令操作失败。

在本公开的一些实施例中,该方法还可以包括:Unity 前端使用预设接口向 Python 后端传输当前页面状态文件,以使得 Python 后端向服务器发送当前页面状态文件,以使得服务器根据当前页面状态文件和数据处理指令得到数据处理结果。

在本公开的一些实施例中,在 Unity 前端使用预设接口向 Python 后端传输当前页面状态文件之前,该方法还包括:Unity 前端根据当前页面状态,生成当前页面状态文件。

在本公开的一些实施例中,该方法还可以包括:Unity 前端使用预设接口读取初始化文件;Unity 前端根据初始化文件进行页面初始化,得到初始页面并显示。

在本公开的一些实施例中,S401 可以包括:Unity 前端响应接收到的对处理文件的显示指令,使用预设接口读取处理文件;或者,Unity 前端按照预设周期使用预设接口读取处理文件。其中,预设接口是采用 Javascript 或 HTML5 脚本语言编写的。

再次,以服务器侧对数据处理方法进行描述。

本公开实施例提供一种数据处理的方法,该方法应用于一服务器中,图 5 为本公开实施例提供的再一种数据处理方法的流程示意图,如图 5 所示,该数据处理方法可以包括:S501:接收到来自 Python 后端的数据处理指令;S502:根据数据处理指令,得到数据处理结果;S503:将数据处理结果发送至 Python 后端,以使得 Python 后端根据数据处理结果,生成处理文件,以使得 Unity 前端使用预设接口读取处理文件,并根据处理文件进行页面显示。

在本公开的一些实施例中,S502 可以包括:接收到来自 Python 后端的当前页面状态文件;根据当前页面状态文件和数据处理指令得到数据处理结果;其中,当前页面状态文件是 Unity 前端使用预设接口向 Python 后端传输的。

在本公开的一些实施例中，根据当前页面状态文件和数据处理指令得到数据处理结果，包括：判断数据处理指令对当前页面状态文件的处理是否符合预设条件；在数据处理指令对当前页面状态文件的处理符合预设条件的情况下，根据数据处理指令，得到结果处理成功的数据处理结果，以使得 Python 后端根据数据处理结果，生成结果处理成功  
5 的处理文件，以使得 Unity 前端使用预设接口读取结果处理成功的处理文件，并根据结果处理成功的处理文件对页面进行刷新。

在本公开的一些实施例中，在判断数据处理指令对当前页面状态文件的处理是否符合预设条件之后，所述方法还包括：在数据处理指令对当前页面状态文件的处理不符合  
10 预设条件的情况下，根据数据处理指令，得到结果处理失败的数据处理结果，以使得 Python 后端根据数据处理结果，生成结果处理失败的处理文件，以使得 Unity 前端使用预设接口读取结果处理失败的处理文件，并根据结果处理失败的处理文件生成错误信息，以提示数据处理指令操作失败。

其中，处理文件中包括结果处理成功或失败的标识，以及界面动作序列，预设接口是采用 Javascript 或 HTML5 脚本语言编写的。

15 本领域技术人员可以理解，在具体实施方式的上述方法中，各步骤的撰写顺序并不意味着严格的执行顺序而对实施过程构成任何限定，各步骤的执行顺序应当以其功能和可能的内在逻辑确定。

本公开实施例提供一种数据处理装置，该数据处理装置设置为可实现 Python 后端功能，图 6 为本公开实施例提供的一种数据处理装置的结构示意图，如图 6 所示，该装置  
20 可以包括：获取模块 61，第一确定模块 62 和第一生成模块 63；其中，

获取模块 61，配置为获取数据处理指令；

第一确定模块 62，配置为根据数据处理指令，得到数据处理结果；

第一生成模块 63，配置为根据数据处理结果，生成处理文件，以使得 Unity 前端使用预设接口读取处理文件，并根据处理文件进行页面显示。

25 在本公开的一些实施例中，第一确定模块 62，配置为：将数据处理指令发送至服务器，以使得服务器根据数据处理指令得到数据处理结果并返回；接收服务器返回的数据处理结果。

在本公开的一些实施例中，第一生成模块 63，配置为：根据数据处理结果，生成被预设脚本语言可识别的处理文件，以使得 Unity 前端使用预设接口读取处理文件，并根据  
30 处理文件进行页面显示；其中，预设接口是采用预设脚本语言编写的。

其中，预设接口是采用 Javascript 或 HTML5 脚本语言编写的，处理文件包括结果处理成功或失败的标识，以及界面动作序列。

在本公开的一些实施例中，获取模块 61，配置为：根据接收到的 Python 代码，生成数据处理指令。

35 在实际应用中，上述获取模块 61、第一确定模块 62 和第一生成模块 63 可由位于 Python 后端上的处理器实现，一般为中央处理器（CPU，Central Processing Unit）、微处理器（MPU，Microprocessor Unit）、数字信号处理器（DSP，Digital Signal Processing）或现场可编程门阵列（FPGA，Field Programmable Gate Array）等实现。

本公开实施例提供一种数据处理装置，该数据处理装置设置为可实现 Unity 前端功能，图 7 为本公开实施例提供的另一种数据处理装置的结构示意图，如图 7 所示，该数  
40 据处理装置可以包括：读取模块 71 和显示模块 72；其中，

读取模块 71, 配置为使用预设接口读取处理文件; 其中, 处理文件是 Python 后端根据数据处理结果生成的结果文件, 数据处理结果是 Python 后端根据获取到的数据处理指令得到的;

显示模块 72, 配置为根据处理文件进行页面显示。

- 5        在本公开的一些实施例中, 显示模块 72, 配置为: 响应于处理文件中包括结果处理成功的标识, 根据处理文件中的界面动作序列刷新页面。

在本公开的一些实施例中, 显示模块 72, 配置为: 响应于处理文件中包括结果处理失败的标识, 生成错误信息, 以提示数据处理指令操作失败。

- 10       在本公开的一些实施例中, 该装置还包括: 传输模块, 配置为使用预设接口向 Python 后端传输当前页面状态文件, 以使得 Python 后端向服务器发送当前页面状态文件, 以使得服务器根据当前页面状态文件和数据处理指令得到数据处理结果。

在本公开的一些实施例中, 该装置还包括: 第二生成模块, 配置为在使用预设接口向 Python 后端传输当前页面状态文件之前, 根据当前页面状态, 生成当前页面状态文件。

- 15       在本公开的一些实施例中, 该装置还包括: 初始化模块, 配置为: 使用预设接口读取初始化文件; 根据所述初始化文件进行页面初始化, 得到初始页面并显示。

在本公开的一些实施例中, 读取模块, 配置为: 响应接收到的对处理文件的显示指令, 使用预设接口读取处理文件; 或者, 按照预设周期使用预设接口读取处理文件。

其中, 预设接口是采用 Javascript 或 HTML5 脚本语言编写的。

- 20       在实际应用中, 上述读取模块 71、显示模块 72、传输模块、第二生成模块和初始化模块可由位于 Unity 前端上的处理器实现, 一般为 CPU、MPU、DSP 或 FPGA 等实现。

本公开实施例提供一种数据处理装置, 该数据处理装置设置为可实现服务器功能, 图 8 为本公开实施例提供的再一种数据处理装置的结构示意图, 如图 8 所示, 该数据处理装置可以包括: 接收模块 81、第二确定模块 82 和发送模块 83; 其中,

- 25       接收模块 81, 配置为接收到来自 Python 后端的数据处理指令;

第二确定模块 82, 配置为根据数据处理指令, 得到数据处理结果;

发送模块 83, 配置为将数据处理结果发送至 Python 后端, 以使得 Python 后端根据数据处理结果, 生成处理文件, 以使得 Unity 前端使用预设接口读取处理文件, 并根据处理文件进行页面显示。

- 30       在本公开的一些实施例中, 第二确定模块 82, 包括:

接收子模块, 配置为接收到来自 Python 后端的当前页面状态文件;

确定子模块, 配置为根据当前页面状态文件和数据处理指令得到数据处理结果;

其中, 当前页面状态文件是 Unity 前端使用预设接口向 Python 后端传输的。

- 35       在本公开的一些实施例中, 确定子模块, 配置为: 判断数据处理指令对当前页面状态文件的处理是否符合预设条件; 在数据处理指令对当前页面状态文件的处理符合预设条件的情况下, 根据数据处理指令, 得到结果处理成功的数据处理结果, 以使得 Python 后端根据数据处理结果, 生成结果处理成功的处理文件, 以使得 Unity 前端使用预设接口读取结果处理成功的处理文件, 并根据结果处理成功的处理文件对页面进行刷新。

在本公开的一些实施例中, 确定子模块, 还配置为: 在判断数据处理指令对当前页

面状态文件的处理是否符合预设条件之后，在数据处理指令对当前页面状态文件的处理不符合预设条件的情况下，根据数据处理指令，得到结果处理失败的数据处理结果，以使得 Python 后端根据数据处理结果，生成结果处理失败的处理文件，以使得 Unity 前端使用预设接口读取结果处理失败的处理文件，并根据结果处理失败的处理文件生成错误信息，以提示数据处理指令操作失败。

其中，处理文件中包括结果处理成功或失败的标识，以及界面动作序列，预设接口是采用 Javascript 或 HTML5 脚本语言编写的。

在实际应用中，上述接收模块 81、第二确定模块 82、发送模块 83、接收子模块和确定子模块可由位于服务器上的处理器实现，一般为 CPU、MPU、DSP 或 FPGA 等实现。

图 9 为本公开实施例提供的一种网页处理设备的结构示意图，如图 9 所示，本公开实施例提供了一种网页处理设备 900，该网页处理设备 900 包括处理器 91、存储器 92 和通信总线 93；其中，

通信总线 93，配置为实现处理器 91 和存储器 92 之间的连接通信；

处理器 91，配置为执行存储器 92 中存储的数据处理程序，以实现上述 Python 后端侧执行的一个或多个实施例所述的数据处理方法。

需要说明的是，实际应用时，网页处理设备 900 中的各个组件通过通信总线 93 耦合在一起。可理解，通信总线 93 配置为实现这些组件之间的连接通信。通信总线 93 除包括数据总线之外，还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见，在图 9 中将各种总线都标为通信总线 93。

图 10 为本公开实施例提供的另一种网页处理设备的结构示意图，如图 10 所示，本公开实施例提供了一种网页处理设备 1000，网页处理设备 1000 包括：处理器 101、存储器 102 和通信总线 103；其中，

通信总线 103，配置为实现处理器 101 和存储器 102 之间的连接通信；

处理器 101，配置为执行存储器 102 中存储的数据处理程序，以实现上述 Unity 前端侧执行的一个或多个实施例所述的数据处理方法。

需要说明的是，实际应用时，网页处理设备 1000 中的各个组件通过通信总线 103 耦合在一起。可理解，通信总线 103 配置为实现这些组件之间的连接通信。通信总线 103 除包括数据总线之外，还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见，在图 10 中将各种总线都标为通信总线 103。

本公开实施例还提供的一种网页处理设备，其结构可参考图 9 或图 10，其处理器可配置为同时实现上述 Python 后端侧和 Unity 前端侧执行的一个或多个实施例所述的数据处理方法。

图 11 为本公开实施例提供的一种服务器的结构示意图，如图 11 所示，本公开实施例提供了一种服务器 1100，服务器 1100 包括：处理器 111、存储器 112 和通信总线 113；其中，

通信总线 113，配置为实现处理器 111 和存储器 112 之间的连接通信；

处理器 111，配置为执行存储器 112 中存储的数据处理程序，以实现上述服务器侧执行的一个或多个实施例所述的数据处理方法。

需要说明的是，实际应用时，服务器 1100 中的各个组件通过通信总线 113 耦合在

一起。可理解，通信总线 113 配置为实现这些组件之间的连接通信。通信总线 113 除包括数据总线之外，还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见，在图 11 中将各种总线都标为通信总线 113。

图 12 为本公开实施例提供的一种数据处理系统的结构示意图，如图 12 所示，本公开实施例提供了一种数据处理系统 1200，数据处理系统 1200 包括网页处理设备 121，  
5 网页处理设备 122，以及的服务器 123；其中，

数据处理系统 1200 包括上述实现上述一个或多个实施例中 Python 后端执行的所述的数据处理方法的网页处理设备 121，实现上述一个或多个实施例中 Unity 前端执行的所述的数据处理方法的网页处理设备 122，以及实现上述一个或多个实施例中的服务器  
10 123；其中，网页处理设备 121 可以为上述一个或多个实施例所述的 Python 后端，网页处理设备 122 可以为上述一个或多个实施例所述的 Unity 前端，服务器 123 为上述一个或多个实施例所述的服务器。

本公开实施例还提供了一种计算机可读存储介质，所述计算机可读存储介质存储有一个或者多个程序，所述一个或者多个程序可以被一个或者多个处理器执行，以实现上  
15 述 Python 后端执行的一个或多个实施例所述的数据处理方法，或者 Unity 前端执行的一个或多个实施例所述的数据处理方法，以及服务器执行的一个或多个实施例所述的数据处理方法。计算机可读存储介质可以是易失性存储器（volatile memory），例如随机存取存储器（Random-Access Memory, RAM）；或者非易失性存储器（non-volatile memory），例如只读存储器（Read-Only Memory, ROM），快闪存储器（flash memory），硬盘（Hard  
20 Disk Drive, HDD）或固态硬盘（Solid-State Drive, SSD）；也可以是包括上述存储器之一或任意组合的各自设备，如移动电话、计算机、平板设备、个人数字助理等。

相应的，本公开实施例还提供一种计算机程序，包括计算机可读代码，当所述计算机可读代码在电子设备中运行时，所述电子设备中的处理器执行用于实现本公开实施例提供的任一数据处理方法。

25 上文对各个实施例的描述倾向于强调各个实施例之间的不同之处，其相同或相似之处可以互相参考，为了简洁，本文不再赘述。

本文中术语“和/或”，仅仅是一种描述关联对象的关联关系，表示可以存在三种关系，例如 A 和/或 B，可以表示：单独存在 A，同时存在 A 和 B，单独存在 B 这三种情况。

30 本领域内的技术人员应明白，本公开的实施例可提供为方法、系统、或计算机程序产品。因此，本公开可采用硬件实施例、软件实施例、或结合软件和硬件方面的实施例的形式。而且，本公开可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质（包括但不限于磁盘存储器和光学存储器等）上实施的计算机程序产品的形式。

35 本公开是参照根据本公开实施例的方法、设备（系统）、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程信号处理设备的处理器以产生一个机器，使得通过计算机或其他可编程信号处理设备的处理器执行的指令产生  
40 用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

这些计算机程序指令也可存储在能引导计算机或其他可编程信号处理设备以特定

方式工作的计算机可读存储器中,使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品,该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

- 5 这些计算机程序指令也可装载到计算机或其他可编程信号处理设备中,使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理,从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

以上所述,仅为本公开的较佳实施例而已,并非用于限定本公开的保护范围。