

一种页面渲染方法、装置、电子设备、计算机可读存储介质及计算机程序产品

相关申请的交叉引用

5 本申请基于申请号为 202310732158.7、申请日为 2023 年 6 月 19 日的中国专利申请提出，并要求以上中国专利申请的优先权，以上中国专利申请的全部内容在此引入本申请作为参考。

技术领域

本申请涉及图形处理技术，尤其涉及一种页面渲染方法、装置、电子设备、计算机可读存储介质及计算机程序产品。

10 背景技术

小程序是一种不需要下载安装即可使用的应用，从而能够为使用者在不同应用的使用上提供巨大的便利。相关技术中，是借助于嵌入式浏览器（例如 WebView）实现小程序的运行的。也即，通过浏览器进程对小程序的业务逻辑代码进行解释和执行，以及通过浏览器进程调用图形渲染函数，以对小程序的页面帧进行渲染。然而，采用浏览器进程对小程序的页面帧进行渲染，会使得小程序被强制关闭的风险较高。

发明内容

本申请实施例提供一种页面渲染方法、装置、电子设备、计算机可读存储介质及计算机程序产品，能够提高小程序的运行稳定性。

20 本申请实施例的技术方案是这样实现的：

本申请实施例提供一种页面渲染方法，应用于电子设备，包括：

响应于针对小程序的启动操作，通过应用程序客户端的客户端进程启动浏览器的浏览器进程，并通过所述浏览器进程执行所述小程序的业务逻辑代码，所述小程序在所述应用程序客户端中运行；

25 响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码，通过所述浏览器进程向客户端进程发送调用处理请求；其中，所述第一渲染函数是所述浏览器进程调用的图形渲染函数，所述第一渲染函数用于渲染所述小程序的页面帧；

响应于所述调用处理请求，通过所述客户端进程确定所述第一渲染函数相对应的第二渲染函数，并对所述第二渲染函数进行调用；其中，所述第二渲染函数和所述第一渲染函数的图形渲染功能相同；

30 通过所述第二渲染函数，对所述小程序的所述页面帧进行图形渲染，得到所述小程序的所述页面帧的渲染结果。

本申请实施例提供一种页面渲染装置，包括：

进程启动模块，配置为响应于针对小程序的启动操作，通过应用程序客户端的客户端进程启动浏览器的浏览器进程，并通过所述浏览器进程执行所述小程序的业务逻辑代

35 码，所述小程序在所述应用程序客户端中运行；

请求发送模块，配置为响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码，通过所述浏览器进程向客户端进程发送调用处理请求；其中，所述第一渲染函数是所述浏览器进程调用的图形渲染函数，所述第一渲染函数用于渲染所述小程序的页面帧；

5 调用处理模块，配置为响应于所述调用处理请求，通过所述客户端进程确定所述第一渲染函数相对应的第二渲染函数，并对所述第二渲染函数进行调用；其中，所述第二渲染函数和所述第一渲染函数的图形渲染功能相同；

图形渲染模块，配置为通过所述第二渲染函数，对所述小程序的所述页面帧进行图形渲染，得到所述小程序的所述页面帧的渲染结果。

10 本申请实施例提供一种电子设备，包括：

存储器，用于存储计算机可执行指令或计算机程序；

处理器，用于执行所述存储器中存储的计算机可执行指令或者计算机程序时，实现本申请实施例提供的页面渲染方法。

15 本申请实施例提供一种计算机可读存储介质，存储有计算机可执行指令或者计算机程序，用于引起处理器执行时，实现本申请实施例提供的页面渲染方法。

本申请实施例提供一种计算机程序产品，包括计算机程序或计算机可执行指令，所述计算机程序或计算机可执行指令被处理器执行时实现本申请实施例提供的页面渲染方法。

20 本申请实施例具有以下有益效果：电子设备会在小程序启动时，先通过应用程序客户端进程启动浏览器进程，通过浏览器进程执行小程序的业务逻辑代码，并在执行到业务逻辑代码中针对第一渲染函数的调用代码时，通过浏览器进程向客户端进程发送调用处理请求，以通过调用处理请求将浏览器进程对第一渲染函数的调用转换为客户端进程对第二渲染函数的调用，并通过第二渲染函数对小程序的页面帧进行图形渲染，由于第二渲染函数是与第一渲染函数具有相同图形渲染功能的渲染函数，如此，在保证图形渲染功能能够正常实现的前提下，将小程序的页面帧的图形渲染过程转移到应用程序客户端实现，使得浏览器只是作为业务逻辑代码的执行器，从而能够大大降低浏览器的内存消耗，减少浏览器进程被强制停止的可能性，最终也就降低了小程序被强制关闭的风险，从而提高小程序的运行稳定性。

附图说明

30 图 1 是小程序的页面帧的渲染示意图；
图 2 是本申请实施例提供的页面渲染系统的架构示意图；
图 3 是本申请实施例提供的图 2 中的终端的结构示意图；
图 4 是本申请实施例提供的页面渲染方法的流程示意图一；
图 5 是本申请实施例提供的页面渲染方法的流程示意图二；
35 图 6 是本申请实施例提供的页面渲染方法的流程示意图三；
图 7 是本申请实施例提供的对小游戏的游戏画面进行渲染的原理示意图；
图 8 是本申请实施例提供的对 WebGL 函数的调用进行记录的一种示意图；
图 9 是本申请实施例提供的对 WebGL 函数的调用进行记录的另一种示意图。

具体实施方式

40 为了使本申请的目的、技术方案和优点更加清楚，下面将结合附图对本申请作进一

步地详细描述，所描述的实施例不应视为对本申请的限制，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其它实施例，都属于本申请保护的范围。

在以下的描述中，涉及到“一些实施例”，其描述了所有可能实施例的子集，但是可以理解，“一些实施例”可以是所有可能实施例的相同子集或不同子集，并且可以在不冲突的情况下相互结合。

在以下的描述中，所涉及的术语“第一\第二”仅仅是区别类似的对象，不代表针对对象的特定排序，可以理解地，“第一\第二”在允许的情况下可以互换特定的顺序或先后次序，以使这里描述的本申请实施例能够以除了在这里图示或描述的以外的顺序实施。

本申请实施例中，术语“模块”或“单元”是指有预定功能的计算机程序或计算机程序的一部分，并与其他相关部分一起工作以实现预定目标，并且可以通过使用软件、硬件（如处理电路或存储器）或其组合来全部实现或部分实现。同样的，一个处理器（或多个处理器或存储器）可用来实现一个或多个模块或单元。此外，每个模块或单元都可以是包含该模块或单元功能的整体模块或单元的一部分。

除非另有定义，本文所使用的所有的技术和科学术语与属于本申请的技术领域的技术人员通常理解的含义相同。本文中所使用的术语只是为了描述本申请实施例的目的，不是旨在限制本申请。

对本申请实施例进行进一步详细说明之前，对本申请实施例中涉及的名词和术语进行说明，本申请实施例中涉及的名词和术语适用于如下的解释。

1) 小程序 (Mini Program)，也称为网络程序 (Web Program)，是一种基于面向前端的语言 (例如 JavaScript) 开发的，在超文本标记语言 (Hyper Text Markup Language, HTML) 页面中实现服务的程序，由客户端经由网络 (如互联网) 下载，并在客户端的浏览器或嵌入式浏览器中解释和执行的软件，节省在客户端中安装的步骤。例如，可以在社交网络客户端中下载、运行用于实现打卡，或者是游戏服务的小程序。

2) 客户端，是使用者需要下载并安装在终端 (例如，智能手机) 上才可以使用的应用程序。例如，浏览器客户端或者社交网络客户端。

3) 客户端进程，是终端上为客户端分配的进程，用于实现客户端的各项功能。

4) 嵌入式浏览器 (WebView)，是嵌入在原生应用 (指能够直接运行于操作系统的应用程序) 中的浏览器，原生应用可以使用嵌入式浏览器展示网络页面。不同的操作系统上会有不同的嵌入式浏览器，例如，一些操作系统上可以使用 UIWebView 和 WKWebView 展示网络页面。

5) 浏览器进程，是终端上位嵌入式浏览器分配的进程，用于实现浏览器进程的各项功能。

6) 页面帧，是指小程序的画面内容的渲染和显示的单位，即小程序的画面内容，是以一个页面帧为单位进行页面渲染，并且以一个页面帧为单位进行显示，从而通过多个页面帧连续显示，就可以动态地呈现小程序的画面内容。

7) 响应于，用于表示所执行的操作所依赖的条件或状态，当满足所依赖的条件或状态时，所执行的一个或多个操作可以是实时的，可以具有设定的延迟；在没有特别说明的情况下，所执行的多个操作不存在执行先后顺序的限制。

8) 渲染函数，是指 Web 开发或小程序开发中用于定义和描述页面如何显示的函数。它通常负责将数据转换成可视化的 HTML 结构，并将其呈现到用户界面上。在不同的框架和平台中，页面渲染函数的实现方式和概念可能有所不同，但其核心目的相似，即定义页面的显示逻辑。

9) 命令缓存，用于存储渲染函数的调用信息的缓存空间。

10) 业务逻辑代码, 是用于完成业务逻辑的代码, 在本申请实施例中, 是指小程序的业务逻辑代码, 也即是在小程序中实现各种功能和交互的代码部分。这些代码可以包括登录验证、页面跳转、数据请求和处理、组件操作等功能。一般来说, 小程序的业务逻辑代码主要包括以下几个方面:

5 1. 页面生命周期函数: 在小程序中每个页面都有自己的生命周期函数, 包括页面加载、页面显示、页面隐藏、页面卸载等不同阶段的回调函数, 可以在这些函数中执行各种逻辑操作。

 2. 数据请求和处理: 小程序需要与服务器进行数据交互, 可以通过内置的请求方法发送请求并处理返回的数据, 包括数据解析、数据展示等操作。

10 3. 事件监测和处理: 小程序中的页面可以监测用户的各种交互事件, 比如点击事件、滑动事件、输入事件等, 可以通过事件处理函数来实现相应的业务逻辑。

 4. 页面跳转和传参: 小程序中需要实现页面之间的跳转, 可以通过内置的应用程序编程接口 (Application Programming Interface, API) 实现页面跳转, 并可以通过参数传递数据。

15 5. 组件操作: 小程序中可以使用各种内置组件和自定义组件来实现页面的不同功能, 可以通过相关 API 控制组件的显示、隐藏、样式等操作。

 总的来说, 小程序的业务逻辑代码就是负责实现小程序各种功能和交互的代码部分, 通过这些代码可以实现小程序的各种复杂逻辑和交互效果。

20 小程序是一种不需要下载安装即可使用的应用, 从而能够为使用者在不同应用的使用上提供巨大的便利。例如, 使用者可以在不安装相应的客户端的情况下, 通过小程序实现考勤打卡, 或者是在不安装相应的客户端的情况下, 通过小程序实现游戏娱乐等等。

 相关技术中, 是借助于嵌入式浏览器 (例如 WebView) 实现小程序的运行的。也即, 通过浏览器进程对小程序的业务逻辑代码进行解释和执行, 以及通过浏览器进程调用图形渲染函数, 以对小程序的页面帧进行渲染。

25 示例性的, 图 1 是小程序的页面帧的渲染示意图。参见图 1, 小程序的代码通常会分为业务逻辑代码 (即小程序的创建者所编写的用于实现小程序的功能的代码) 11、调用客户端的应用程序编程接口 (Application Programming Interface, API) 12 和渲染函数 13 的调用接口, 渲染函数 13 用于对小程序的页面帧进行渲染。当客户端检测到针对小程序的启动操作之后, 客户端进程会启动操作系统的嵌入式浏览器的浏览器进程, 浏览器进程执行业务逻辑代码 11, 并调用渲染函数 13, 同时会通过 API 12 调用客户端进程, 以实现文件读写、信息登录等。

30 然而, 随着小程序使用时长的增加, 这种方式所需要的渲染的页面帧的数量也会增加, 会使得嵌入式浏览器的内存占用过高, 从而导致出现浏览器进程被终端强制停止, 最终使得小程序无法继续使用。由此可见, 相关技术中, 使用浏览器进程对小程序的页面帧进行渲染, 会使得小程序被强制关闭的风险较高。

 另外, 嵌入式浏览器还可能会随着终端的操作系统一起升级 (即操作系统升级时, 嵌入式浏览器也可能会相应地升级)。然而, 嵌入式浏览器的升级可能会带来一些运行故障 (Bug), 如此, 在使用浏览器进程对小程序的页面帧进行渲染时, 也可能出现故障, 从而影响小程序的正常使用。

40 本申请实施例提供一种页面渲染方法、装置、设备、计算机可读存储介质和计算机程序产品, 能够降低小程序被强制关闭的风险。下面说明本申请实施例提供的用于页面渲染的电子设备的示例性应用, 本申请实施例提供的电子设备可以实施为笔记本电脑, 平板电脑, 台式计算机, 机顶盒, 移动设备 (例如, 移动电话, 便携式音乐播放器, 个人数字助理, 专用消息设备, 便携式游戏设备) 等各种类型的终端, 也可以实施为服务

器。下面，将说明电子设备实施为终端时的示例性应用。

参见图 2，图 2 是本申请实施例提供的页面渲染系统的架构示意图。为实现支撑一个页面渲染应用，在页面渲染系统 100 中，终端(示例性示出了终端 400-1 和终端 400-2)通过网络 300 连接服务器 200，网络 300 可以是广域网或者局域网，又或者是二者的组合。在页面渲染系统 100 中，还设置有数据库 500，用于向服务器 200 提供数据支持。其中，数据库 500 可以独立于服务器 200，也可以配置在服务器 200 中。图 2 示出的是数据库 500 独立于服务器 200 的情况。

终端 400-1 和终端 400-2 分别用于响应于使用者在图形界面 410-1 和图形界面 410-2 上针对小程序的启动操作，通过客户端进程从服务器 200 下载小程序的程序代码以及启动浏览器的浏览器进程，并通过浏览器进程执行小程序的程序代码中的业务逻辑代码；响应于业务逻辑代码针对小程序的页面帧调用第一渲染函数，通过浏览器进程向客户端进程发送调用处理请求，其中，第一渲染函数是浏览器进程能够调用的图形渲染函数；响应于调用处理请求，通过客户端进程针对第一渲染函数确定相对应的第二渲染函数，并对第二渲染函数进行调用；其中，第二渲染函数是客户端进程能够调用的图形渲染函数；通过第二渲染函数，对小程序的页面帧进行图形渲染，得到小程序的页面帧的渲染结果；将每个页面帧的渲染结果依次展示在图形界面 410-1 和图形界面 410-2 上。

服务器 200 用于将小程序的程序代码下发给终端 400-1 和终端 400-2。

本申请实施例可以借助于云技术(Cloud Technology)实现，云技术是指在广域网内或局域网内将硬件、软件、网络等系列资源统一起来，实现数据的计算、存储、处理和共享的一种托管技术。

云计算是基于云计算商业模式应用的网络技术、信息技术、整合技术、管理平台、以及应用技术等的总称，可以组成资源池，按需所用，灵活便利。云计算技术将变成重要支撑。技术网络的系统后台服务需要大量的计算和存储资源，需要通过云计算实现。

示例性的，服务器 200 可以是独立的物理服务器，也可以是多个物理服务器构成的服务器集群或者分布式系统，还可以是提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、CDN、以及大数据和人工智能平台等基础云计算服务的云服务器。终端 400-1 和终端 400-2 可以是智能手机、平板电脑、笔记本电脑、台式计算机、智能音箱、智能手表、智能家电、车载终端等，但并不局限于此。终端以及服务器可以通过有线或无线通信方式进行直接或间接地连接，本申请实施例中不做限制。

参见图 3，图 3 是本申请实施例提供的图 2 中的终端(电子设备的一种实施)的结构示意图。图 3 所示的终端 400 包括：至少一个处理器 410、存储器 450、至少一个网络接口 420 和用户接口 430。终端 400 中的各个组件通过总线系统 440 耦合在一起。可理解，总线系统 440 用于实现这些组件之间的连接通信。总线系统 440 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 3 中将各种总线都标为总线系统 440。

处理器 410 可以是一种集成电路芯片，具有信号的处理能力，例如通用处理器、数字信号处理器(DSP, Digital Signal Processor)，或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等，其中，通用处理器可以是微处理器或者任何常规的处理器等。

用户接口 430 包括使得能够呈现媒体内容的一个或多个输出装置 431，包括一个或多个扬声器和/或一个或多个视觉显示屏。用户接口 430 还包括一个或多个输入装置 432，包括有助于用户输入的用户接口部件，比如键盘、鼠标、麦克风、触屏显示屏、摄像头、其他输入按钮和控件。

存储器 450 可以是可移除的，不可移除的或其组合。示例性的硬件设备包括固态存储器，硬盘驱动器，光盘驱动器等。存储器 450 可选地包括在物理位置上远离处理器 410 的一个或多个存储设备。

存储器 450 包括易失性存储器或非易失性存储器，也可包括易失性和非易失性存储器两者。非易失性存储器可以是只读存储器（ROM，Read Only Memory），易失性存储器可以是随机存取存储器（RAM，Random Access Memory）。本申请实施例描述的存储器 450 旨在包括任意适合类型的存储器。

在一些实施例中，存储器 450 能够存储数据以支持各种操作，这些数据的示例包括程序、模块和数据结构或者其子集或超集，下面示例性说明。

操作系统 451，包括用于处理各种基本系统服务和执行硬件相关任务的系统程序，例如框架层、核心库层、驱动层等，用于实现各种基础业务以及处理基于硬件的任务；

网络通信模块 452，用于经由一个或多个（有线或无线）网络接口 420 到达其他计算设备，示例性的网络接口 420 包括：蓝牙、无线相容性认证（WiFi）、和通用串行总线（USB，Universal Serial Bus）等；

呈现模块 453，用于经由一个或多个与用户接口 430 相关联的输出装置 431（例如，显示屏、扬声器等）使得能够呈现信息（例如，用于操作外围设备和显示内容和信息的用户接口）；

输入处理模块 454，用于对一个或多个来自一个或多个输入装置 432 之一的一个或多个用户输入或互动进行检测以及翻译所检测的输入或互动。

在一些实施例中，本申请实施例提供的页面渲染装置可以采用软件方式实现，图 3 示出了存储在存储器 450 中的页面渲染装置 455，其可以是程序和插件等形式的软件，包括以下软件模块：进程启动模块 4551、请求发送模块 4552、调用处理模块 4553、图形渲染模块 4554、脚本处理模块 4555 和映射生成模块 4556，这些模块是逻辑上的，因此根据所实现的功能可以进行任意的组合或进一步拆分。将在下文中说明各个模块的功能。

在另一些实施例中，本申请实施例提供的页面渲染装置可以采用硬件方式实现，作为示例，本申请实施例提供的页面渲染装置可以是采用硬件译码处理器形式的处理器，其被编程以执行本申请实施例提供的页面渲染方法，例如，硬件译码处理器形式的处理器可以采用一个或多个应用专用集成电路（ASIC，Application Specific Integrated Circuit）、DSP、可编程逻辑器件（PLD，Programmable Logic Device）、复杂可编程逻辑器件（CPLD，Complex Programmable Logic Device）、现场可编程门阵列（FPGA，Field-Programmable Gate Array）或其他电子元件。

在一些实施例中，终端或服务器（均为电子设备可能的实施）可以通过运行计算机程序来实现本申请实施例提供的页面渲染方法。举例来说，计算机程序可以是操作系统中的原生程序或软件模块：可以是本地（Native）应用程序（APP，Application），即需要在操作系统中安装才能运行的程序，如社交网络 APP 或者即时通信 APP；也可以是小程序，即只需要下载到浏览器环境中就可以运行的程序；还可以是能够嵌入至任意 APP 中的小程序。总而言之，上述计算机程序可以是任意形式的应用程序、模块或插件。

本申请实施例可以应用于应用软件、操作系统等的小程序的页面渲染场景中。下面，说明本申请实施例提供的页面渲染方法，如前所述，实现本申请实施例的页面渲染方法的电子设备可以为终端、服务器，又或者是二者的结合。因此在下文中不再重复说明各个步骤的执行主体。

将结合本申请实施例提供的电子设备的示例性应用和实施，说明本申请实施例提供的页面渲染方法。

参见图 4，图 4 是本申请实施例提供的页面渲染方法的流程示意图一，将结合图 4 示出的步骤进行说明。

S101、响应于针对小程序的启动操作，通过应用程序客户端的客户端进程启动浏览器的浏览器进程，并通过浏览器进程执行小程序的业务逻辑代码。

5 本申请实施例是在对小程序的页面帧进行图形渲染的场景下实现的，经过图形渲染之后的页面帧，用于对小程序的画面内容进行呈现。本申请实施例中，电子设备会实时检测是否接收到针对小程序的启动操作，当接收到针对小程序的启动操作，并触发小程序的运行之后，就会通过应用程序客户端的客户端进程启动浏览器的浏览器进程，并通过浏览器进程对小程序的业务逻辑代码进行解释和执行，从而实现小程序的业务功能。

10 本申请实施例中的小程序可以是游戏类的小程序（也可以称为小游戏），也可以是办公软件类的小程序，或者是音乐类等各种类型的小程序，本申请实施例在此不做限定。

需要说明的是，本申请实施例中，客户端进程是指嵌入了小程序的应用程序的客户端的进程，也即小程序是在该应用程序客户端中运行，从而，针对小程序的启动操作，是指在应用程序客户端（以下简称为客户端）上所触发的进入小程序的操作。该启动操作可以通过点击小程序的图标、扫描小程序的二维码等方式实现。其中，上述的应用程序可以是指社交网络应用程序，也可以是音乐应用程序、游戏应用程序等，从而，客户端可以是社交网络客户端、音乐客户端、游戏客户端等等。小程序即是可以嵌入在这些客户端中运行，以向安装了这些客户端的使用者提供小程序所具有的功能。

20 本申请实施例中的浏览器可以是客户端所提供的浏览器，也可以是电子设备的操作系统所提供的嵌入式浏览器，本申请实施例在此不做限定。浏览器进程是指电子设备针对浏览器所分配的进程。

小程序的业务逻辑代码，是用于实现小程序的具体功能的代码，其一般由小程序的创建者编写。小程序的业务逻辑代码会在小程序首次启动时，由客户端从服务器拉取得到，在非首次启动时，也可以从客户端的缓存文件中提取得到。

25 S102、响应于浏览器进程执行至业务逻辑代码中针对第一渲染函数的调用代码，通过浏览器进程向客户端进程发送调用处理请求。

为了将小程序的画面内容展示出来，业务逻辑代码会调用第一渲染函数对小程序的页面帧逐个或者并行进行渲染。需要说明的是，第一渲染函数是浏览器进程能够调用的图形渲染函数，而业务逻辑代码通过浏览器进程进行执行，当浏览器进程执行至业务逻辑代码中针对第一渲染函数的调用代码时，其中浏览器进程执行至业务逻辑代码中针对第一渲染函数的调用代码，表示浏览器进程已经执行了针对第一渲染函数的调用代码，此时如果通过浏览器进程调用第一渲染函数，以进行页面渲染。那么由于浏览器进程调用第一渲染函数进行页面渲染时，容易造成浏览器所占用的内存过大，导致浏览器进程被强制停止。为了避免该情况，本申请实施例中，响应于浏览器进程执行至业务逻辑代码中针对第一渲染函数的调用代码，电子设备会将浏览器进程内的页面渲染的函数调用，通过调用处理请求转发至客户端进程，以通过客户端实现页面渲染的函数调用。如此，浏览器就可以仅看作为业务逻辑代码的执行器，仅需要加载极少的页面，从而能够减少浏览器的内存消耗。

本申请实施例中，浏览器进程可以通过多种方式，向客户端进程发送调用处理请求。

40 参见图 5，图 5 是本申请实施例提供的页面渲染方法的流程示意图二。在本申请的一些实施例中，图 4 中 S102 的过程，即响应于浏览器进程执行至业务逻辑代码中针对第一渲染函数的调用代码，通过浏览器进程向客户端进程发送调用处理请求，可以通过 S1021 至 S1022 实现，如下：

S1021、响应于浏览器进程执行至业务逻辑代码中针对第一渲染函数的调用代码，

通过浏览器进程在命令缓存中写入第一渲染函数的调用信息。

电子设备执行至业务逻辑代码中针对第一渲染函数的调用代码时，会通过浏览器进程，将第一渲染函数的调用信息写入到命令缓存中，以通过命令缓存对第一渲染函数的调用进行记录。其中，浏览器进程可以将针对每个页面帧所调用的所有第一渲染函数的调用信息，均写入同一个命令缓存中，也可以将每个第一渲染函数的调用信息，分别写入针对其所分配的命令缓存中（即浏览器进程针对每个第一渲染函数均分配了命令缓存，每个命令缓存中只会写入所对应的第一渲染函数的调用信息）。

本申请实施例中，第一渲染函数的调用信息，至少需要包括第一渲染函数的函数标识和第一渲染函数的调用参数（也可以理解为第一渲染函数的入参）。其中，函数标识用于对不同的第一渲染函数进行区分，函数标识可以为第一渲染函数的函数名称，也可以为第一渲染函数的函数标识。第一渲染函数的调用参数可以依据第一渲染函数的具体的渲染功能变化，例如，第一渲染函数为图形绘制函数时，第一渲染函数的调用参数即为图形的各个点的坐标，第一渲染函数为着色函数时，第一渲染函数的调用参数即为颜色代码。

需要说明的是，本申请实施例中的命令缓存可以由浏览器进程本身创建的，也可以是由客户端进程创建并注入进行浏览器进程的。

在一些实施例中，在图 5 中 S1021 之前，即响应于浏览器进程执行至业务逻辑代码中针对第一渲染函数的调用代码，通过浏览器进程在命令缓存中写入第一渲染函数的调用信息之前，该方法还可以包括以下处理：通过客户端进程，生成命令缓存的脚本代码，并将命令缓存的脚本代码注入至浏览器进程中。

需要说明的是，命令缓存的脚本代码用于创建命令缓存，并控制浏览器进程将第一渲染函数的调用信息写入至命令缓存中。也即，客户端进程生成了一段可以自动执行的脚本代码，该脚本代码用于开辟一个可进行读写的命令缓存，同时对第一渲染函数的调用进行拦截，并控制浏览器进程将对应的调用信息进行记录。其中，将命令缓存的脚本代码注入至浏览器进程，可以通过在浏览器进程中创建线程，并使该线程执行命令缓存的脚本代码实现，也可以通过生成一个脚本代码所对应可执行文件，利用该可执行文件对浏览器进程中针对第一渲染函数的调用代码（或代码段）进行替换来实现，也就是说将针对第一渲染函数的调用代码替换为该可执行文件。

S1022、通过浏览器进程，针对命令缓存生成调用处理请求，并将调用处理请求发送至客户端进程。

电子设备通过浏览器进程在命令缓存中写入第一渲染函数的调用信息之后，就会通过浏览器进程，生成针对第一渲染函数的调用处理请求，并将写入了调用信息的命令缓存打包至调用处理请求中，通过跨进程通讯，将调用处理请求发送至客户端进程。如此，客户端进程在接收到调用处理请求之后，就能够从中解析得到命令缓存，从而得到第一渲染函数的调用信息。

需要说明的是，本申请实施例中，电子设备可以在浏览器进程将第一渲染函数的调用信息写入至命令缓存之后，立即通过浏览器进程针对命令缓存生成调用处理请求并发送至客户端进程，也可以在浏览器进程将第一渲染函数的调用信息写入命令缓存之后，在达到一定的请求发送条件时（即需要进行调用处理请求的发送时），再通过浏览器进程针对命令缓存生成调用处理请求并发送至客户端。

其中，请求发送条件可以是指第一渲染函数为存在返回值的同步渲染函数（例如 `gl.createBuffer` 函数），也可以是针对页面帧所调用的所有第一渲染函数的调用信息均已经被记录在命令缓存中，本申请实施例在此不做限定。

在一些实施例中，图 5 中 S1022，即通过浏览器进程，针对命令缓存生成调用处理

请求，并将调用处理请求发送至客户端进程，可以通过以下处理实现：当业务逻辑代码调用的第一渲染函数为存在返回值的同步渲染函数时，通过浏览器进程，针对命令缓存生成所述调用处理请求，并将调用处理请求发送至客户端进程。

需要说明的是，同步渲染函数必须要在得到返回值之后才能够继续运行，否则会一直处于暂停运行状态，以继续等待返回值。因此，当第一渲染函数为存在返回值的同步渲染函数时，需要及时将命令缓存提交给客户端进程，以使得客户端进程针对第一渲染函数生成对应的返回值并返回，从而使得第一渲染函数能够继续执行。

在另一些实施例中，图5中S1022，即通过浏览器进程，针对命令缓存生成调用处理请求，并将调用处理请求发送至客户端进程，还可以通过以下处理实现：当页面帧在渲染时所需要调用的多个第一渲染函数各自的调用信息均已被记录在命令缓存时，通过浏览器进程，针对命令缓存生成调用处理请求，并将调用处理请求发送至客户端进程。

可以理解的是，跨进程通讯需要消耗一定的时间才能实现，而一个页面帧可能需要调用多个，例如几十或上百个第一渲染函数，才能够完成图形渲染，这些第一渲染函数均可以看作是页面帧在渲染时所需要的调用的第一渲染函数。当这些第一渲染函数各自的调用信息，均已经被记录在命令缓存中时，电子设备才会通过浏览器进程，针对命令缓存生成调用处理请求，并将其发送给客户端进程。如此，能够将一个页面帧所需要的所有第一渲染函数的调用批量提交给客户端进程，从而能够减少跨进程通讯的次数，提升页面帧在图形渲染时的处理效率。

需要说明的是，本申请实施例中，将调用处理请求发送至客户端进程，可以通过XMLHttpRequest（XHR）请求实现，也可以基于WebSocket协议等技术实现，本申请实施例在此不做限定。

在一些实施例中，将调用处理请求发送至客户端进程，可以通过以下处理实现：通过浏览器进程，将调用处理请求的访问地址设置为客户端进程的服务地址；通过客户端进程，在服务地址对调用处理请求进行拦截，完成将调用处理请求发送至客户端进程。

也即，浏览器进程直接将客户端进程的服务地址作为调用处理请求的访问地址，然后客户端进程可以通过请求拦截控件（例如WKURLSchemeHandler控件），在其服务地址对浏览器进程发来的请求进行拦截，以实现浏览器进程和客户端进程之间的跨进程通讯，完成调用处理请求的发送。

S103、响应于调用处理请求，通过客户端进程确定第一渲染函数对应的第二渲染函数，并对第二渲染函数进行调用。

电子设备通过客户端进程对调用处理请求进行响应，确定出与第一渲染函数具有相同图形渲染功能的第二渲染函数，且第二渲染函数是客户端进程能够调用的图像渲染函数。接着，电子设备会通过客户端进程调用第二渲染函数，从而实现将对第一渲染函数的调用转换为对第二渲染函数的调用，以便于后续利用第二渲染函数实现对小程序的页面帧的渲染。

也就是说，本申请实施例中，第一渲染函数和第二渲染函数所能够进行的图形渲染功能相同，而在被调用的方式上存在不同。例如，若第一渲染函数为创建缓冲区对象的函数（例如WebGL中的creatBuffer函数）时，第二渲染函数也为缓冲区创建对象的函数（例如OpenGL中用于创建缓冲区对象的函数）。

需要说明的是，第二渲染函数可以是客户端进程根据第一渲染函数的功能，从已有的可被客户端进程调用的图形渲染函数中筛选得到的。在实现时，可以预先建立具有相同图形渲染功能的，浏览器进程能够调用的各个图形渲染函数，与客户端进程能够调用的图形渲染函数之间的对应关系表，然后从该对应关系表中查询第一渲染函数对应的第二渲染函数。在一些实施例中，第二渲染函数也可以是客户端进程对第一渲染函数进行

转换改写之后得到的，本申请实施例在此不做限定。

本申请实施例中，当调用信息包括调用参数时，对第二渲染函数进行调用，可以通过以下处理实现：通过客户端进程，从命令缓存中解析得到第一渲染函数的调用参数，并将第一渲染函数的调用参数转换为第二渲染函数的调用参数；通过客户端进程，将第

5 电子设备通过客户端进程，从调用处理请求中解析得到命令缓存，并从命令缓存中解析得到第一渲染函数的调用参数，并将其转换成第二渲染函数可用的调用参数，最后将转换得到的调用参数作为第二渲染函数的入参，传递至第二渲染函数中，以实现对第二渲染函数的调用。例如，当第一渲染函数的调用参数为需要绘制的图形各个坐标点时，

10 电子设备会通过客户端进程将其按照第二渲染函数参数格式进行转换，并将转换结果输入进第二渲染参数中，完成调用。

需要说明的是，在将第一渲染函数的调用参数转换为第二渲染函数可用的调用参数时，可以参考第二渲染函数对应的官方文档和指南，以确保正确理解和使用不同函数的参数及其含义。逐一比较第一渲染函数和第二渲染函数中需要转换的调用参数，注意对

15 应参数的差异和相似之处，进行适当的修改和调整，并且调用参数可能在第一渲染函数和第二渲染函数中的顺序有细微的差异，为了确保参数传递的顺序是正确的，因此还需要将转换得到的调用参数按照第二渲染函数要求的函数参数顺序进行排列，从而确保第二渲染函数能够正常运行并实现相同的图形渲染效果。

S104、通过第二渲染函数，对小程序的页面帧进行图形渲染，得到小程序的页面帧

20 的渲染结果。

电子设备通过第二渲染函数，针对小程序的每个页面帧进行图形渲染的处理，例如针对每个页面帧进行图案绘制、着色等等，在完成图形渲染之后，就能够得到每个页面帧的渲染结果。

可以理解的是，相比于相关技术中，通过浏览器进程调用图形渲染函数对小程序的

25 页面帧进行渲染，会造成浏览器的内存占用过高，从而使得小程序被强制关闭的风险较高的问题，本申请实施例中，电子设备会在使用者启动小程序时，先通过客户端进程启动浏览器进程，通过浏览器进行对小程序的业务逻辑代码进行执行，并在业务逻辑代码需要调用第一渲染函数时，通过浏览器进程向客户端进程发送调用处理请求，以通过调用处理请求将浏览器进程对第一渲染函数的调用转换为客户端进程对第二渲染函数的

30 调用，并通过第二渲染函数对小程序的页面帧进行图形渲染，如此，能够将小程序的页面帧的图形渲染过程转移到客户端实现，使得浏览器只是作为业务逻辑代码的执行器，而无需调用图形处理器所需的资源开销，从而能够大大降低浏览器的内存消耗，减少浏览器进程被强制停止的可能性，最终也就降低了小程序被强制关闭的风险。另外，将小程序的页面帧的图形渲染过程转移到客户端实现，实现了将浏览器和小程序的页面帧的

35 图形渲染的解绑，如此，即使浏览器的升级可能会带来运行故障，也不会对小程序的页面帧的图形渲染带来影响，从而能够保证小程序正常使用。

需要说明的是，在上述方案中，当第一渲染函数为存在返回值的同步渲染函数时，浏览器进程会立即将命令缓存携带在调用处理请求中发送给客户端进程，以实现命令缓存的提交，并且在得到返回值之后，才会继续后续执行。可见，这种方式由于需要等待

40 返回值的返回，从而仍旧需要花费一些时间才能够继续将后续的第一渲染函数的调用信息写入命令缓存中，并提交命令缓存，从而使得运行效率受到影响。

对此，参见图6，图6是本申请实施例提供的页面渲染方法的流程示意图三。在本申请的一些实施例中，在图5中S1022之前，即通过浏览器进程，针对命令缓存生成调用处理请求，并将调用处理请求发送至客户端进程之前，该方法还可以包括以下处理：

S1023、当第一渲染函数为存在返回值的同步渲染函数时，响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码，通过浏览器进程，针对第一渲染函数返回虚拟返回值，并将虚拟返回值和第一渲染函数的调用信息，共同写入至命令缓存中。

也就是说，当第一渲染函数为存在返回值的同步渲染函数时，电子设备会通过浏览器进程，先针对第一渲染函数生成虚拟返回值，并将该虚拟返回值返回给第一渲染函数，以使得第一渲染函数能够继续运行，如此，浏览器进程便无需针对第一渲染函数进行返回值的等待，从而能够继续对后续其他的第一渲染函数的调用信息进行记录，并在针对页面帧完成对所有的第一渲染函数的调用信息的记录之后，一次性地将命令缓存提交至客户端进程。其中，虚拟返回值可以是浏览器进程，依据第一渲染函数的返回值的格式随机生成的，也可以是第一渲染函数的历史返回值，本申请实施例在此不做限定。

需要说明的是，为了使得后续客户端进程能够正确将对第一渲染函数的调用转换为对第二渲染函数的调用，电子设备会通过浏览器进程，将第一渲染函数的调用信息（即函数标识、调用参数等）以及虚拟返回值写入到命令缓存中，以供后续客户端进程从命令缓存中解析得到第一渲染函数的调用信息和虚拟返回值。

当命令缓存中存在第一渲染函数的虚拟返回值时，在本申请的一些实施例中，通过第二渲染函数，对小程序的页面帧进行图形渲染，得到小程序的页面帧的渲染结果之后，该方法还包括：通过客户端进程，从调用处理请求中解析得到命令缓存，并从命令缓存中提取得到虚拟返回值；通过客户端进程，将第二渲染函数的返回值作为第一渲染函数的真实返回值，并建立真实返回值与虚拟返回值之间的映射关系。

其中，映射关系用于在未来时间针对命令缓存中的虚拟返回值确定对应的真实返回值。

需要说明的是，由于本申请实施例中，针对第一渲染函数的调用被转换为针对第二渲染函数的调用，从而第二渲染函数与第一渲染函数具有相同的功能，因此，第二渲染函数的返回值就能够作为第一渲染函数的真实返回值。例如，当第一渲染函数的功能为创建缓冲区对象时，第二渲染函数的功能也会为创建缓冲区对象，从而第二渲染函数的返回值可以为缓冲区对象的标识（例如 bufferID），而第一渲染函数的返回值也应该为缓冲区对象的标识，因此，电子设备会通过客户端进程，直接将第二渲染函数的返回值确定为第一渲染函数的真实返回值，并通过建立真实返回值和虚拟返回值之间的映射关系，将真实返回值和虚拟返回值进行绑定，如此，当浏览器进程在未来时间针对第一渲染函数再次向客户端进程传递虚拟返回值时，客户端进程可以通过该映射关系，直接为第一渲染函数确定真实返回值，以便于后续功能的进行。

下面，将说明本申请实施例在一个实际的应用场景中的示例性应用。

本申请实施例是在对小游戏（称为小程序）的游戏画面（称为页面帧）进行渲染的场景下实现的。在本申请实施例中，对小游戏的游戏画面的渲染，由 WebView（称为浏览器）切换到了社交应用的客户端实现。

图 7 是本申请实施例提供的对小游戏的游戏画面进行渲染的原理示意图。参见图 7，本申请实施例中，将浏览器进程 71（也可称为 WebView 进程）内的第一渲染函数 73（示例性地，第一渲染函数可以为 WebGL 函数）的调用，通过跨进程通讯，转发到客户端进程 72 来实现，而小游戏的业务逻辑代码，即 JS 代码 74，仍旧在浏览器进程 71 中运行，同时会通过应用程序编程接口 75 调用客户端进程 72，以实现文件读写、信息登录等。

然而，WebGL 函数的接口的数量很多，JS 代码针对每一帧的游戏画面可能需要几十乃至上百次的调用，若每次调查 WebGL 函数，都需要通过跨进程通讯以转发到客户

端进程来实现，那么使得小游戏的游戏画面的渲染的运行性能非常差。因此，本申请实施例中，会通过浏览器进程，将 WebGL 函数的调用记录到命令缓存中，当需要提交时，通过跨进程通讯将 WebGL 函数的调用一次性批量提交给客户端进程，以进行批量处理。

在实现时，会由客户端进程在浏览器进程中注入命令缓存的脚本代码，该脚本代码

5 用于拦截 JS 代码的 WebGL 函数的调用，以及将 WebGL 函数的调用记录到命令缓存中。

示例性的，图 8 是本申请实施例提供的对 WebGL 函数的调用进行记录的一种示意图。当 JS 代码执行 WebGL 函数时，浏览器进程会将 WebGL 函数的调用，即 WebGL 函数的函数 ID（称为函数标识）以及调用参数，写入到命令缓存中，以便于后续进行批量提交。

10 当一帧结束或者是遇到同步函数（称为同步渲染函数）的调用时，需要将命令缓存提交给客户端进程。本申请实施例中，命令缓存可以转换为二进制的数，从而跨进程通讯可以使用 XHR 方式实现，即浏览器进程会通过 XHR 请求携带命令缓存的数据，提交给客户端进程，客户端进程通过 WKURLSchemeHandler 拦截该 XHR 请求（称为调用处理请求）。当然，跨进程通讯也可以通过 WebScket、Prompt、postMessage 等方式实现。

15 正常情况下，当调用有返回值的同步函数，例如调用 `gl.createBuffer`，时，浏览器进程会对命令缓存进行一次提交，并等待返回值，从而使得渲染效率不是很高。对此，本申请实施例中，可以在调用有返回值的同步函数时，将调用写入命令缓存中，而不触发同步提交。此时，在调用该同步函数时，浏览器进程会返回一个假 BufferID（称为虚拟返回值），并将假 BufferID 与函数调用一起写入至命令缓存中。客户端进程在解析命令缓存，并生成真 BufferID（称为真实返回值）之后，会对真 BufferID 和假 BufferID 进行映射，之后，当浏览器进程再次传递假 BufferID 时，客户端进程可以直接依据该映射关系将假 BufferID 替换为真 BufferID，以实现功能。

20 示例性的，图 9 是本申请实施例提供的对 WebGL 函数的调用进行记录的另一种示意图。在命令缓存中，除了记录了一些不具有返回值的 WebGL 函数的函数标识（也即函数 ID）和调用参数之外，针对创建命令缓存（`createBuffer`）函数还记录了虚拟返回值（也即假 BufferID），以便将其传递给客户端进程。

客户端进程在渲染时，即客户端进程在收到浏览器进程传递的指令表明需要创建画布，并通过 `getContext` 调用 WebGL 函数时（此时客户端进程针对该指令并不会进行执行），会在客户端创建对应的绘制图层（例如 `EAGLView`），并创建 `EAGLContext`。之后，客户端进程接收到命令缓存之后，会将 WebGL 函数的调用转换为 OpenGL ES 的函数（称为第二渲染函数），从而实现渲染。

30 最终，浏览器只是作为 JS 代码的执行器，此时，对于浏览器而言，其只需要加载极小，甚至空白的 HTML 页面（能够保证 JS 代码的正常执行即可），没有 GPU 资源相关的开销，如此，就能够大大减少 WebView 本身的内存消耗。同时，由于不再需要浏览器进行渲染，从而能够避免浏览器升级的故障针对小游戏的游戏画面的渲染带来的影响。

下面继续说明本申请实施例提供的页面渲染装置 455 的实施为软件模块的示例性结构，在一些实施例中，如图 3 所示，存储在存储器 450 的页面渲染装置 455 中的软件模块可以包括：

40 进程启动模块 4551，配置为响应于针对小程序的启动操作，通过应用程序客户端的客户端进程启动浏览器的浏览器进程，并通过所述浏览器进程执行所述小程序的业务逻辑代码，所述小程序在所述应用程序客户端中运行；

请求发送模块 4552，配置为响应于所述浏览器进程执行至所述业务逻辑代码中针对

第一渲染函数的调用代码,通过所述浏览器进程向客户端进程发送调用处理请求;其中,所述第一渲染函数是所述浏览器进程调用的图形渲染函数,所述第一渲染函数用于渲染所述小程序的页面帧;

5 调用处理模块 4553,配置为响应于所述调用处理请求,通过所述客户端进程确定所述第一渲染函数相对应的第二渲染函数,并对所述第二渲染函数进行调用;其中,所述第二渲染函数和所述第一渲染函数的图形渲染功能相同;

图形渲染模块 4554,配置为通过所述第二渲染函数,对所述小程序的所述页面帧进行图形渲染,得到所述小程序的所述页面帧的渲染结果。

10 在本申请的一些实施例中,所述请求发送模块 4552,还配置为响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码,通过所述浏览器进程在命令缓存中写入所述第一渲染函数的调用信息;通过所述浏览器进程,针对所述命令缓存生成所述调用处理请求,并将所述调用处理请求发送至所述客户端进程。

15 在本申请的一些实施例中,所述请求发送模块 4552,还配置为当所述所述第一渲染函数为存在返回值的同步渲染函数时,通过所述浏览器进程,针对所述命令缓存生成所述调用处理请求,并将所述调用处理请求发送至所述客户端进程。

在本申请的一些实施例中,所述请求发送模块 4552,还配置为当所述页面帧在渲染时所需要调用的多个第一渲染函数各自的调用信息均已被记录在所述命令缓存时,通过所述浏览器进程,针对所述命令缓存生成所述调用处理请求,并将所述调用处理请求发送至所述客户端进程。

20 在本申请的一些实施例中,所述页面渲染装置 455 还包括:脚本处理模块 4555,配置为通过所述客户端进程,生成所述命令缓存的脚本代码,并将所述命令缓存的脚本代码注入至所述浏览器进程中;所述命令缓存的脚本代码用于创建所述命令缓存,并控制所述浏览器进程将所述第一渲染函数的调用信息写入至所述命令缓存中。

25 在本申请的一些实施例中,所述脚本处理模块 4555,还配置为在所述浏览器进程中创建线程,利用所述线程执行所述命令缓存的脚本代码;或者,生成所述命令缓存的脚本代码对应的可执行文件,将针对所述第一渲染函数的调用代码替换为所述可执行文件。

30 在本申请的一些实施例中,所述请求发送模块 4552,还配置为通过所述浏览器进程,将所述调用处理请求的访问地址设置为所述客户端进程的服务地址;通过所述客户端进程,在所述服务地址对所述调用处理请求进行拦截,完成将所述调用处理请求发送至所述客户端进程。

35 在本申请的一些实施例中,所述请求发送模块 4552,还配置为当所述所述第一渲染函数为存在返回值的同步渲染函数时,响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码,通过所述浏览器进程,针对所述第一渲染函数返回虚拟返回值,并将所述虚拟返回值和所述第一渲染函数的调用信息,共同写入至所述命令缓存中。

40 在本申请的一些实施例中,所述页面渲染装置 455 还包括:映射生成模块 4556,配置为通过所述客户端进程,从所述调用处理请求中解析得到所述命令缓存,并从所述命令缓存中提取得到所述虚拟返回值;通过所述客户端进程,将所述第二渲染函数的返回值作为所述第一渲染函数的真实返回值,并建立所述真实返回值与所述虚拟返回值之间的映射关系;其中,所述映射关系用于在未来时间针对所述命令缓存中的所述虚拟返回值确定对应的真实返回值。

在本申请的一些实施例中,所述调用信息包括:调用参数;所述调用处理模块 4553,还配置为通过所述客户端进程,从所述命令缓存中解析得到所述第一渲染函数的调用参

数，并将所述第一渲染函数的调用参数转换为第二渲染函数的调用参数；通过所述客户端进程，将所述第二渲染函数的调用参数，传递至所述第二渲染函数，完成将对所述第二渲染函数的调用。

本申请实施例提供了一种计算机程序产品，该计算机程序产品包括计算机程序或计算机可执行指令，该计算机程序或计算机可执行指令存储在计算机可读存储介质中。电子设备的处理器从计算机可读存储介质读取该计算机可执行指令，处理器执行该计算机可执行指令，使得该电子设备执行本申请实施例上述的页面渲染方法。

本申请实施例提供一种存储有计算机可执行指令的计算机可读存储介质，其中存储有计算机可执行指令，当计算机可执行指令被处理器执行时，将引起处理器执行本申请实施例提供的页面渲染方法，例如，如图4示出的页面渲染方法。

在一些实施例中，计算机可读存储介质可以是FRAM、ROM、PROM、EPROM、EEPROM、闪存、磁表面存储器、光盘、或CD-ROM等存储器；也可以是包括上述存储器之一或任意组合的各种设备。

在一些实施例中，计算机可执行指令可以采用程序、软件、软件模块、脚本或代码的形式，按任意形式的编程语言（包括编译或解释语言，或者声明性或过程性语言）来编写，并且其可按任意形式部署，包括被部署为独立的程序或者被部署为模块、组件、子例程或者适合在计算环境中使用的其它单元。

作为示例，计算机可执行指令可以但不一定对应于文件系统中的文件，可以可被存储在保存其它程序或数据的文件的一部分，例如，存储在超文本标记语言（HTML，Hyper Text Markup Language）文档中的一个或多个脚本中，存储在专用于所讨论的程序的单个文件中，或者，存储在多个协同文件（例如，存储一个或多个模块、子程序或代码部分的文件）中。

作为示例，计算机可执行指令可被部署为在一个电子设备上执行，或者在位于一个地点的多个电子设备上执行，又或者，在分布在多个地点且通过通信网络互连的多个电子设备上执行。

综上所述，通过本申请实施例，电子设备会在使用者启动小程序时，先通过客户端进程启动浏览器进程，通过浏览器进行对小程序的业务逻辑代码进行执行，并在业务逻辑代码需要调用第一渲染函数时，通过浏览器进程向客户端进程发送调用处理请求，以通过调用处理请求将浏览器进程对第一渲染函数的调用转换为客户端进程对第二渲染函数的调用，并通过第二渲染函数对小程序的页面帧进行图形渲染，如此，能够将小程序的页面帧的图形渲染过程转移到客户端实现，使得浏览器只是作为业务逻辑代码的执行器，从而能够大大降低浏览器的内存消耗，减少浏览器进程被强制停止的可能性，最终也就降低了小程序被强制关闭的风险。另外，将小程序的页面帧的图形渲染过程转移到客户端实现，实现了将浏览器和小程序的页面帧的图形渲染的解绑，如此，即使浏览器的升级可能会带来运行故障，也不会对小程序的页面帧的图形渲染带来影响，从而能够保证小程序正常使用。

以上所述，仅为本申请的实施例而已，并非用于限定本申请的保护范围。凡在本申请的精神和范围之内所作的任何修改、等同替换和改进等，均包含在本申请的保护范围之内。

权利要求书

- 1、一种页面渲染方法，应用于电子设备，所述电子设备中包括应用程序客户端和浏览器，所述方法包括：
- 5 响应于针对小程序的启动操作，通过应用程序客户端的客户端进程启动浏览器的浏览器进程，并通过所述浏览器进程执行所述小程序的业务逻辑代码，所述小程序在所述应用程序客户端中运行；
- 响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码，通过所述浏览器进程向客户端进程发送调用处理请求；其中，所述第一渲染函数是所述浏览器进程调用的图形渲染函数，所述第一渲染函数用于渲染所述小程序的页面帧；
- 10 响应于所述调用处理请求，通过所述客户端进程确定所述第一渲染函数相对应的第二渲染函数，并对所述第二渲染函数进行调用；其中，所述第二渲染函数和所述第一渲染函数的图形渲染功能相同；
- 通过所述第二渲染函数，对所述小程序的所述页面帧进行图形渲染，得到所述小程序的所述页面帧的渲染结果。
- 15 2、根据权利要求1所述的方法，其中，所述响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码，通过所述浏览器进程向客户端进程发送调用处理请求，包括：
- 响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码，通过所述浏览器进程在命令缓存中写入所述第一渲染函数的调用信息；
- 20 通过所述浏览器进程，针对所述命令缓存生成所述调用处理请求，并将所述调用处理请求发送至所述客户端进程。
- 3、根据权利要求1或2所述的方法，其中，所述通过所述浏览器进程，针对所述命令缓存生成所述调用处理请求，并将所述调用处理请求发送至所述客户端进程，包括：
- 25 当所述第一渲染函数为存在返回值的同步渲染函数时，通过所述浏览器进程，针对所述命令缓存生成所述调用处理请求，并将所述调用处理请求发送至所述客户端进程。
- 4、根据权利要求1至3任一项所述的方法，其中，所述通过所述浏览器进程，针对所述命令缓存生成所述调用处理请求，并将所述调用处理请求发送至所述客户端进程，包括：
- 30 当在渲染所述页面帧时需要调用的多个第一渲染函数各自的调用信息均已被记录在所述命令缓存时，通过所述浏览器进程，针对所述命令缓存生成所述调用处理请求，并将所述调用处理请求发送至所述客户端进程。
- 5、根据权利要求1至4任一项所述的方法，其中，所述响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码，通过所述浏览器进程在命令缓存中写入所述第一渲染函数的调用信息之前，所述方法还包括：
- 35 通过所述客户端进程，生成所述命令缓存的脚本代码；
- 将所述命令缓存的脚本代码注入至所述浏览器进程中；所述命令缓存的脚本代码用于创建所述命令缓存；
- 控制所述浏览器进程将所述第一渲染函数的调用信息写入至所述命令缓存中。
- 6、根据权利要求1至5任一项所述的方法，所述将所述命令缓存的脚本代码注入至所述浏览器进程中，包括：
- 40 在所述浏览器进程中创建线程，利用所述线程执行所述命令缓存的脚本代码；或者，生成所述命令缓存的脚本代码对应的可执行文件，将针对所述第一渲染函数的调用代码替换为所述可执行文件。

7、根据权利要求1至6任一项所述的方法，其中，所述将所述调用处理请求发送至所述客户端进程，包括：

通过所述浏览器进程，将所述调用处理请求的访问地址设置为所述客户端进程的服务地址；

5 通过所述客户端进程，在所述服务地址对所述调用处理请求进行拦截，完成将所述调用处理请求发送至所述客户端进程。

8、根据权利要求1至7任一项所述的方法，其中，所述通过所述浏览器进程，针对所述命令缓存生成所述调用处理请求，并将所述调用处理请求发送至所述客户端进程之前，所述方法还包括：

10 当所述第一渲染函数为存在返回值的同步渲染函数时，响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码，通过所述浏览器进程，针对所述第一渲染函数返回虚拟返回值，并将所述虚拟返回值和所述第一渲染函数的调用信息，共同写入至所述命令缓存中。

9、根据权利要求1至8任一项所述的方法，其中，所述通过所述第二渲染函数，对所述小程序的所述页面帧进行图形渲染，得到所述小程序的所述页面帧的渲染结果之后，所述方法还包括：

通过所述客户端进程，从所述调用处理请求中解析得到所述命令缓存，并从所述命令缓存中提取得到所述虚拟返回值；

20 通过所述客户端进程，将所述第二渲染函数的返回值作为所述第一渲染函数的真实返回值，并建立所述真实返回值与所述虚拟返回值之间的映射关系；

其中，所述映射关系用于针对所述命令缓存中的所述虚拟返回值确定对应的真实返回值。

10、根据权利要求1至9任一项所述的方法，其中，所述调用信息包括：调用参数；所述对所述第二渲染函数进行调用，包括：

25 通过所述客户端进程，从所述命令缓存中解析得到所述第一渲染函数的调用参数，并将所述第一渲染函数的调用参数转换为第二渲染函数的调用参数；

通过所述客户端进程，将所述第二渲染函数的调用参数，传递至所述第二渲染函数，完成对所述第二渲染函数的调用。

11、一种页面渲染装置，所述装置包括：
30 进程启动模块，配置为响应于针对小程序的启动操作，通过应用程序客户端的客户端进程启动浏览器的浏览器进程，并通过所述浏览器进程执行所述小程序的业务逻辑代码，所述小程序在所述应用程序客户端中运行；

请求发送模块，配置为响应于所述浏览器进程执行至所述业务逻辑代码中针对第一渲染函数的调用代码，通过所述浏览器进程向客户端进程发送调用处理请求；其中，所述
35 所述第一渲染函数是所述浏览器进程调用的图形渲染函数，所述第一渲染函数用于渲染所述小程序的页面帧；

调用处理模块，配置为响应于所述调用处理请求，通过所述客户端进程确定所述第一渲染函数相对应的第二渲染函数，并对所述第二渲染函数进行调用；其中，所述第二渲染函数和所述第一渲染函数的图形渲染功能相同；

40 图形渲染模块，配置为通过所述第二渲染函数，对所述小程序的所述页面帧进行图形渲染，得到所述小程序的所述页面帧的渲染结果。

12、一种电子设备，所述电子设备包括：
存储器，用于存储计算机可执行指令或计算机程序；
处理器，用于执行所述存储器中存储的计算机可执行指令或者计算机程序时，实现

权利要求 1 至 10 任一项所述的页面渲染方法。

13、一种计算机可读存储介质，存储有计算机可执行指令或者计算机程序，所述计算机可执行指令被处理器执行时实现权利要求 1 至 10 任一项所述的页面渲染方法。

14、一种计算机程序产品，包括计算机程序或计算机可执行指令，，所述计算机程序或计算机可执行指令被处理器执行时实现权利要求 1 至 10 任一项所述的页面渲染方法。