(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117609646 A
(43)申请公布日 2024.02.27

(21)申请号 202311587356.5

(22)申请日 2023.11.24

(71)申请人 中国电信股份有限公司技术创新中心
地址 102209 北京市昌平区北七家镇未来科技城南区中国电信北京信息科技创新园11层1118室、1116室
申请人 中国电信股份有限公司

(72)发明人 万亭君 董石磊

(74)专利代理机构 北京律智知识产权代理有限公司 11438
专利代理师 王珊珊

(51)Int.Cl.
*G06F 16/957*(2019.01)
*G06T 15/00*(2011.01)

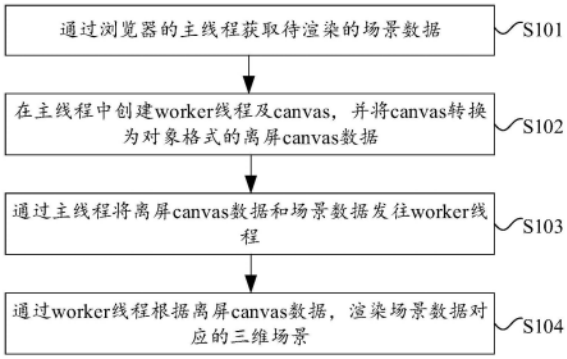
权利要求书2页 说明书11页 附图4页

(54)发明名称

场景渲染方法、装置、电子设备及存储介质

(57)摘要

本公开提供了一种场景渲染方法、装置、电子设备及存储介质，涉及软件技术领域。该方法包括：通过浏览器的主线程获取待渲染的场景数据；在主线程中创建worker线程及画布canvas，并将canvas转换为对象格式的离屏canvas数据；通过主线程将离屏canvas数据和场景数据发往worker线程；通过worker线程根据离屏canvas数据，渲染场景数据对应的三维场景。通过worker线程根据离屏canvas数据及场景数据渲染三维场景的方式，实现了将计算量大的三维渲染任务转移到worker线程，避免了在主线程中进行三维渲染任务，减少了浏览器阻塞和页面克顿，提升用户体验。




CN 117609646 A

1.一种场景渲染方法,其特征在于,包括:

通过浏览器的主线程获取待渲染的场景数据;

在所述主线程中创建worker线程及画布canvas,并将所述canvas转换为对象格式的离屏canvas数据;

通过所述主线程将所述离屏canvas数据和所述场景数据发往所述worker线程;

通过所述worker线程根据所述离屏canvas数据,渲染所述场景数据对应的三维场景。

2.根据权利要求1所述的方法,其特征在于,所述通过所述worker线程根据所述离屏canvas数据,渲染所述场景数据对应的三维场景,包括:

通过所述worker线程解析所述离屏canvas数据,得到canvas参数;

在所述worker线程中创建所述canvas参数对应的离屏canvas;

通过所述worker线程在所述离屏canvas中创建初始化三维场景;

通过所述worker线程在所述初始化三维场景中渲染所述场景数据,得到所述三维场景。

3.根据权利要求1所述的方法,其特征在于,所述将所述canvas转换为对象格式的离屏canvas数据,包括:

通过主线程调用offCanvas＝canvas.value.transferControlToOffscreen()方法,将所述canvas转换为对象格式的离屏canvas数据。

4.根据权利要求1所述的方法,其特征在于,所述通过所述主线程将所述离屏canvas数据和所述场景数据发往所述worker线程,包括:

通过所述主线程调用worker.postMessage()方法,将所述离屏canvas数据和所述场景数据发往所述worker线程;

在所述通过所述worker线程根据所述离屏canvas数据,渲染所述场景数据对应的三维场景之前,还包括:

通过所述worker线程调用worker.onmessage()方法接收离屏canvas数据和所述场景数据。

5.根据权利要求1所述的方法,其特征在于,所述三维场景是在第一相机camera下的场景,所述方法还包括:

在所述worker线程中,将所述第一camera的参数转换为对象格式的第一camera参数,并将所述第一camera参数发往所述主线程;

在所述主线程中构建所述第一camera参数对应的第二camera;

在所述主线程中构造第二camera对应的轨道控制器OrbitControls;

在所述主线程中通过所述OrbitControls获取对所述三维场景的调整操作,得到对所述第二camera调整后的第三camera;

在所述主线程中,将所述第三camera的参数转换为对象格式的第二camera参数,并将所述第二camera参数发往所述worker线程;

通过所述worker线程根据所述第二camera参数更新所述三维场景。

6.根据权利要求5所述的方法,其特征在于,还包括:

在所述主线程中使用worker.onmessage()方法配置监听方法;

在所述主线程中调用所述监听方法接收所述第一camera参数。

7.根据权利要求5所述的方法,其特征在于,所述通过所述worker线程根据所述第二camera参数更新所述三维场景,包括:

通过所述worker线程调用updateCamera()方法解析所述第二camera参数,并根据解析得到的数据更新所述第一camera,得到第四camera;

通过所述worker线程根据所述第四camera,重新对所述场景数据进行渲染,得到更新后的三维场景。

8.一种场景渲染装置,其特征在于,包括:

获取模块,用于通过浏览器的主线程获取待渲染的场景数据;

创建及转换模块,用于在所述主线程中创建worker线程及画布canvas,并将所述canvas转换为对象格式的离屏canvas数据;

发送模块,用于所述worker线程管理模块,还用于通过所述主线程将所述离屏canvas数据和所述场景数据发往所述worker线程;

渲染模块,用于通过所述worker线程根据所述离屏canvas数据,渲染所述场景数据对应的三维场景。

9.一种电子设备,其特征在于,包括:

处理器;以及

存储器,用于存储所述处理器的可执行指令;

其中,所述处理器配置为经由执行所述可执行指令来执行权利要求1~7中任意一项所述的场景渲染方法。

10.一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,所述计算机程序被处理器执行时实现权利要求1~7中任意一项所述的场景渲染方法。

# 场景渲染方法、装置、电子设备及存储介质

## 技术领域

[0001] 本公开涉及软件技术领域,尤其涉及一种场景渲染方法、装置、电子设备及存储介质。

## 背景技术

[0002] 随着软件技术的发展,三维场景展示的需求剧增,而三维场景的展示离不开对三维场景的渲染。而三维场景通常采用Three js(JavaScript(一种编程语言)编写的WebGL(Web Graphics Library,一种3D绘图协议)第三方库)技术结合JavaScript语言进行开发。

[0003] 相关技术中,使用浏览器主线程渲染三维场景,用户对三维场景的视角、位置、缩放比例进行调整后,主线程再次进行更新渲染。

[0004] 然而,渲染三维场景需要占据主线程,而渲染三维场景需要消耗大量的时间和性能,造成主线程的卡顿及后续任务的阻塞,进而导致页面卡顿。

[0005] 需要说明的是,在上述背景技术部分公开的信息仅用于加强对本公开的背景的理解,因此可以包括不构成对本领域普通技术人员已知的现有技术的信息。

## 发明内容

[0006] 本公开提供一种场景渲染方法、装置、电子设备及存储介质,至少在一定程度上克服了相关技术中浏览器页面中渲染三维场景而导致页面卡顿的问题。

[0007] 本公开的其他特性和优点将通过下面的详细描述变得显然,或部分地通过本公开的实践而习得。

[0008] 根据本公开的一个方面,提供一种场景渲染方法,包括:通过浏览器的主线程获取待渲染的场景数据;在所述主线程中创建worker线程及画布canvas,并将所述canvas转换为对象格式的离屏canvas数据;通过所述主线程将所述离屏canvas数据和所述场景数据发往所述worker线程;通过所述worker线程根据所述离屏canvas数据,渲染所述场景数据对应的三维场景。

[0009] 在本公开的一个实施例中,所述通过所述worker线程根据所述离屏canvas数据,渲染所述场景数据对应的三维场景,包括:通过所述worker线程解析所述离屏canvas数据,得到canvas参数;在所述worker线程中创建所述canvas参数对应的离屏canvas;通过所述worker线程在所述离屏canvas中创建初始化三维场景;通过所述worker线程在所述初始化三维场景中渲染所述场景数据,得到所述三维场景。

[0010] 在本公开的一个实施例中,所述将所述canvas转换为对象格式的离屏canvas数据,包括:通过主线程调用offCanvas=canvas.value.transferControlToOffscreen()方法,将所述canvas转换为对象格式的离屏canvas数据。

[0011] 在本公开的一个实施例中,所述通过所述主线程将所述离屏canvas数据和所述场景数据发往所述worker线程,包括:通过所述主线程调用worker.postMessage()方法,将所述离屏canvas数据和所述场景数据发往所述worker线程;在所述通过所述worker线程根

据所述离屏canvas数据,渲染所述场景数据对应的三维场景之前,还包括:通过所述worker线程调用worker.onmessage()方法接收离屏canvas数据和所述场景数据。

[0012] 在本公开的一个实施例中,所述三维场景是在第一相机camera下的场景,所述方法还包括:在所述worker线程中,将所述第一camera的参数转换为对象格式的第一camera参数,并将所述第一camera参数发往所述主线程;在所述主线程中构建所述第一camera参数对应的第二camera;

[0013] 在所述主线程中构造第二camera对应的轨道控制器OrbitControls;在所述主线程中通过所述OrbitControls获取对所述三维场景的调整操作,得到对所述第二camera调整后的第三camera;在所述主线程中,将所述第三camera的参数转换为对象格式的第二camera参数,并将所述第二camera参数发往所述worker线程;通过所述worker线程根据所述第二camera参数更新所述三维场景。

[0014] 在本公开的一个实施例中,还包括:在所述主线程中使用worker.onmessage()方法配置监听方法;在所述主线程中调用所述监听方法接收所述第一camera参数。

[0015] 在本公开的一个实施例中,所述通过所述worker线程根据所述第二camera参数更新所述三维场景,包括:通过所述worker线程调用updateCamera()方法解析所述第二camera参数,并根据解析得到的数据更新所述第一camera,得到第四camera;通过所述worker线程根据所述第四camera,重新对所述场景数据进行渲染,得到更新后的三维场景。

[0016] 根据本公开的另一个方面,提供一种场景渲染装置,包括:获取模块,用于通过浏览器的主线程获取待渲染的场景数据;创建及转换模块,用于在所述主线程中创建worker线程及画布canvas,并将所述canvas转换为对象格式的离屏canvas数据;发送模块,用于所述worker线程管理模块,还用于通过所述主线程将所述离屏canvas数据和所述场景数据发往所述worker线程;渲染模块,用于通过所述worker线程根据所述离屏canvas数据,渲染所述场景数据对应的三维场景。

[0017] 在本公开的一个实施例中,所述渲染模块,用于通过所述worker线程解析所述离屏canvas数据,得到canvas参数;在所述worker线程中创建所述canvas参数对应的离屏canvas;通过所述worker线程在所述离屏canvas中创建初始化三维场景;通过所述worker线程在所述初始化三维场景中渲染所述场景数据,得到所述三维场景。

[0018] 在本公开的一个实施例中,所述创建及转换模块,用于通过主线程调用offCanvas=canvas.value.transferControlToOffscreen()方法,将所述canvas转换为对象格式的离屏canvas数据。

[0019] 在本公开的一个实施例中,所述发送模块,用于通过所述主线程调用worker.postMessage()方法,将所述离屏canvas数据和所述场景数据发往所述worker线程;所述装置还包括:接收模块,用于通过所述worker线程调用worker.onmessage()方法接收离屏canvas数据和所述场景数据。

[0020] 在本公开的一个实施例中,所述三维场景是在第一相机camera下的场景,所述装置还包括:转换及发送模块,用于在所述worker线程中,将所述第一camera的参数转换为对象格式的第一camera参数,并将所述第一camera参数发往所述主线程;构建模块,用于在所述主线程中构建所述第一camera参数对应的第二camera;构造模块,用于在所述主线程中构造第二camera对应的轨道控制器OrbitControls;记录及调整模块,用于在所述主线程中

通过所述OrbitControls获取对所述三维场景的调整操作,得到对所述第二camera调整后的第三camera;所述转换及发送模块,还用于在所述主线程中,将所述第三camera的参数转换为对象格式的第二camera参数,并将所述第二camera参数发往所述worker线程;其中,所述渲染模块,还用于通过所述worker线程根据所述第二camera参数更新所述三维场景。

[0021] 在本公开的一个实施例中,所述装置还包括:配置模块,用于在所述主线程中使用worker.onmessage()方法配置监听方法;在所述主线程中调用所述监听方法接收所述第一camera参数。

[0022] 在本公开的一个实施例中,所述渲染模块,用于通过所述worker线程调用updateCamera()方法解析所述第二camera参数,并根据解析得到的数据更新所述第一camera,得到第四camera;通过所述worker线程根据所述第四camera,重新对所述场景数据进行渲染,得到更新后的三维场景。

[0023] 根据本公开的再一个方面,提供一种电子设备,包括:处理器;以及存储器,用于存储所述处理器的可执行指令;其中,所述处理器配置为经由执行所述可执行指令来执行上述任一所述的场景渲染方法。

[0024] 根据本公开的又一个方面,提供一种计算机可读存储介质,其上存储有计算机程序,所述计算机程序被处理器执行时实现上述任一所述的场景渲染方法。

[0025] 根据本公开的又一个方面,提供一种计算机程序产品,所述计算机程序产品包括计算机程序或计算机指令,所述计算机程序或所述计算机指令由处理器加载并执行,以使计算机实现上述任一所述的场景渲染方法。

[0026] 本公开的实施例所提供的技术方案至少包括以下有益效果:

[0027] 本公开的实施例所提供的技术方案,通过在主线程中创建worker线程及canvas,并将canvas的离屏canvas数据及场景数据发往worker线程,通过worker线程根据离屏canvas数据及场景数据渲染三维场景的方式,实现了将计算量大的三维渲染任务转移到worker线程,避免了在主线程中进行三维渲染任务,减少了浏览器阻塞和页面克顿,提升用户体验。

[0028] 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,并不能限制本公开。

## 附图说明

[0029] 此处的附图被并入说明书中并构成本说明书的一部分,示出了符合本公开的实施例,并与说明书一起用于解释本公开的原理。显而易见地,下面描述中的附图仅仅是本公开的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0030] 图1示出本公开一个实施例中的场景渲染方法流程图;

[0031] 图2示出本公开另一个实施例中的场景渲染方法流程图;

[0032] 图3示出本公开一个实施例中的场景渲染信令图;

[0033] 图4示出本公开一个实施例中的场景渲染装置示意图;

[0034] 图5示出本公开另一个实施例中的场景渲染装置示意图;

[0035] 图6示出本公开一个实施例中的电子设备的结构框图。

## 具体实施方式

[0036] 现在将参考附图更全面地描述示例实施方式。然而,示例实施方式能够以多种形式实施,且不应被理解为限于在此阐述的范例;相反,提供这些实施方式使得本公开将更加全面和完整,并将示例实施方式的构思全面地传达给本领域的技术人员。所描述的特征、结构或特性可以以任何合适的方式结合在一个或更多实施方式中。

[0037] 此外,附图仅为本公开的示意性图解,并非一定是按比例绘制。图中相同的附图标记表示相同或类似的部分,因而将省略对它们的重复描述。附图中所示的一些方框图是功能实体,不一定必须与物理或逻辑上独立的实体相对应。可以采用软件形式来实现这些功能实体,或在一个或多个硬件模块或集成电路中实现这些功能实体,或在不同网络和/或处理器装置和/或微控制器装置中实现这些功能实体。

[0038] 应当理解,本公开的方法实施方式中记载的各个步骤可以按照不同的顺序执行,和/或并行执行。此外,方法实施方式可以包括附加的步骤和/或省略执行示出的步骤。本公开的范围在此方面不受限制。

[0039] 需要注意,本公开中提及的“第一”、“第二”等概念仅用于对不同的装置、模块或单元进行区分,并非用于限定这些装置、模块或单元所执行的功能的顺序或者相互依存关系。

[0040] 需要注意,本公开中提及的“一个”、“多个”的修饰是示意性而非限制性的,本领域技术人员应当理解,除非在上下文另有明确指出,否则应该理解为“一个或多个”。

[0041] 为了便于理解,下面首先对本公开涉及到的几个名词进行解释如下:

[0042] JavaScript:简称“JS”,是一种具有函数优先的轻量级,解释型或即时编译型的编程语言。JavaScript语言的一个特点是单线程,即同一时间仅能处理一个任务,作为浏览器脚本语言,JavaScript的主要用途是与用户互动,这也决定了它只能是单线程,否则会带来复杂的同步问题。

[0043] Threejs:一个WebGL引擎,基于JavaScript,可直接运行GPU(Graphics Processing Unit,图形处理器)驱动游戏与图形驱动应用于浏览器。其库提供的特性与API(Application Programming Interface,应用程序编程接口)可在浏览器中绘制三维场景。

[0044] Canvas(画布):Canvas API是在HTML5(超文本5.0)中新增的标签用于在网页实时生成图像,并且可以操作图像内容,它是一个可以用JavaScript操作的位图(bitmap)。

[0045] worker线程:浏览器执行环境是单线程的,一旦出现主线程耗时操作,就会造成浏览器卡死,用户点击没响应等情况。为了利用多核CPU(Central Processing Unit,中央处理器)的计算能力,HTML5提出Web Worker标准,允许JS脚本创建多个线程,但是子线程完全受主线程控制,且不得操作DOM(Document Object Model,文档对象模型)。所以新标准并没有改变JS单线程的本质。

[0046] 本公开提供了一种场景渲染方法、装置、电子设备及存储介质,该方法可以应用在具有浏览器的电子设备上,该电子设备,包括但不限于智能手机、平板电脑、膝上型便携计算机、台式计算机、可穿戴设备、增强现实设备、虚拟现实设备等。

[0047] 该电子设备还可以是提供各种服务的服务器,在一些实施例中,服务器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(Content Delivery Network,内容分发网络)、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。

[0048] 在基于浏览器展示的、基于Threejs的三维场景中使用新增的worker线程加载运行场景,为了保证canvas可以在worker线程中使用,采用基于主线程创建离屏canvas的方式。同时,为了保证用户针对三维场景的操作,如缩放、旋转、移动视角等,可以被正常展示。采用同步场景相机参数的方法保证用户操作可以在离屏画布中被同步渲染,从而满足基于Threejs技术的大规模三维场景的流畅渲染和交互。降低对浏览器线程的压力、减少阻塞和卡顿、代码简单容易管理。本公开的场景渲染方法独立与框架之外,可以结合React(一种用于构建用户界面JavaScript库)、Vue(一种用于构建用户界面的JavaScript框架)、Angular(一种Web前端框架)等主流框架使用,提升性能。使用独立的worker线程、使用离屏canvas渲染,主线程无需关注加载、运算、更新渲染,保证了页面运行效率及流畅度,降低页面卡顿。

[0049] 下面结合附图及实施例对本示例实施方式进行详细说明。

[0050] 本公开实施例中提供了一种场景渲染方法,该方法可以由任意具备计算处理能力的电子设备执行。

[0051] 图1示出本公开一个实施例中的场景渲染方法流程图,如图1所示,本公开实施例中提供的场景渲染方法包括如下S101至S104。

[0052] S101,通过浏览器的主线程获取待渲染的场景数据。

[0053] 关于场景数据具体为何种场景下的数据,本公开的实施例不做限制。例如,场景数据可以是3D(3Three Dimensions,三维)游戏中的三维画面对应的数据,也可以是任意一个三维场景对应的数据。例如,该场景数据中包括用于描述场景对象的数据,以及用于描述每个场景对象的位置、形状、大小等信息的数据。

[0054] 电子设备获取到场景数据后,主线程可以直接通过存储地址调用该场景数据。关于电子设备如何获取该场景数据,本公开的实施例不做限制。例如,在电子设备中直接创建该场景数据,或者通过网络获取该场景数据。

[0055] S102,在主线程中创建worker线程及canvas,并将canvas转换为对象格式的离屏canvas数据。

[0056] 在一个实施例中,在主线程中创建worker线程,可以包括:通过主线程中调用new Worker()方法创建worker线程。

[0057] 在一个实施例中,将canvas转换为对象格式的离屏canvas数据,包括:通过主线程调用offCanvas=canvas.value.transferControlToOffscreen()方法,将canvas转换为对象格式的离屏canvas数据。

[0058] 其中,离屏canvas数据中记载了canvas的参数信息,例如,canvas的位置、长度、宽度等。

[0059] S103,通过主线程将离屏canvas数据和场景数据发往worker线程。

[0060] 在一个实施例中,通过主线程将离屏canvas数据和场景数据发往worker线程,可以包括:通过主线程调用worker.postMessage()方法,将离屏canvas数据和场景数据发往worker线程。

[0061] 在worker线程利用canvas数据和场景数据进行三维场景渲染前,worker线程可以调用worker.onmessage()方法接收离屏canvas数据和场景数据。

[0062] S104,通过worker线程根据离屏canvas数据,渲染场景数据对应的三维场景。

[0063] 在一个实施例中,通过worker线程根据离屏canvas数据,渲染场景数据对应的三维场景,可以包括:通过worker线程解析离屏canvas数据,得到canvas参数;在worker线程中创建canvas参数对应的离屏canvas;通过worker线程在离屏canvas中创建初始化三维场景;通过worker线程在初始化三维场景中渲染场景数据,得到三维场景。

[0064] 解析离屏canvas数据后,worker线程可以得到canvas的具体参数,包括canvas的位置、宽度和长度等。之后,可以根据canvas参数创建一个离屏canvas,以用于在该离屏canvas中进行三维场景的渲染。

[0065] 在一个实施例中,通过worker线程在离屏canvas中创建初始化三维场景,可以包括:使用worker线程中的init()方法,创建初始化三维场景。

[0066] 在一个实施例中,创建初始化三维场景可以包括:创建初始化的scene(场景)、light(灯光)、camera。

[0067] 在一个实施例中,创建初始化的camera,可以包括:设置正交相机或者透视相机;设置camera不同的参数,包括:位置、长、宽、上边界、下边界、远面、近面、视野角度、朝向等。

[0068] 在一个实施例中,创建初始化的light,可以包括:设置场景中光源的位置、颜色、设置是否有阴影、阴影分辨率。

[0069] 在一个实施例中,通过worker线程在初始化三维场景中渲染场景数据,得到三维场景,可以包括:通过worker线程调用render()函数,在初始化三维场景中渲染场景数据,得到三维场景。

[0070] 通过在主线程中创建worker线程及canvas,并将canvas的离屏canvas数据及场景数据发往worker线程,通过worker线程根据离屏canvas数据及场景数据渲染三维场景的方式,实现了将计算量大的三维渲染任务转移到worker线程,避免了在主线程中进行三维渲染任务,减少了浏览器阻塞和页面克顿,提升用户体验。

[0071] 在本公开的一个实施例中,在完成场景的渲染后,还可以在对三维场景进行缩放、旋转、移动角度等操作后,通过如图2所示的另一种场景渲染方法来更新渲染该场景。如图2所示,该方法包括如下S201至S206。

[0072] S201,在worker线程中,将第一camera的参数转换为对象格式的第一camera参数,并将第一camera参数发往主线程。

[0073] 在一个实施例中,将第一camera参数发往主线程,可以包括:通过worker线程调用worker.postMessage()方法,将第一camera参数发往主线程。

[0074] 需要说明的是,worker.postMessage()方法还可以用于触发主线程调用createCamera()方法,进而使用createCamera()方法创建第一camera参数对应的第二camera。

[0075] 其中,三维场景是在第一camera下渲染出的场景。该第一camera是初始化三维场景下创建的初始化的camera。

[0076] S202,在主线程中构建第一camera参数对应的第二camera。

[0077] 在一个实施例中,在主线程中构建第一camera参数对应的第二camera之前,还包括:在主线程中使用worker.onmessage()方法配置监听方法;在主线程中调用监听方法接收第一camera参数。

[0078] 在一个实施例中,在主线程中构建第一camera参数对应的第二camera,可以包括:在主线程中低调用createCamera()方法创建第一camera参数对应的第二camera。

[0079] S203,在主线程中构造第二camera对应的OrbitControls。

[0080] 其中,OrbitControls(轨道控制器)用于提供对三维场景进行调整的界面操作,如缩放、旋转、移动视角等,从而记录该调整操作。

[0081] 在一个实施例中,在主线程中构造第二camera对应的OrbitControls,可以包括:在主线程中调用new OrbitControls()方法,构造第二camera对应的OrbitControls。

[0082] S204,在主线程中通过OrbitControls获取对三维场景的调整操作,得到对第二camera调整后的第三camera。

[0083] 通过OrbitControls提供界面操作后,对该三维场景进行的调整操作会被主线程记录,进而根据该记录可以更新调整主线程中的第二camera,从而得到调整后的第三camera。

[0084] 在一个实施例中,调整操作可以是操作者对三维场景进行一些缩放、旋转、移动视角等操作。例如,操作者可以通过该三维场景提供的调整操作输入口输入该调整操作,该输入口可以通过鼠标,或键盘,或滑动屏幕等方式输入调整操作。

[0085] S205,在主线程中,将第三camera的参数转换为对象格式的第二camera参数,并将第二camera参数发往worker线程。

[0086] 在一个实施例中,将第二camera参数发往worker线程,可以包括:通过主线程调用worker.postMessage()方法将第二camera参数发往worker线程。

[0087] 其中,第二camera参数包含的具体内容可以参见上述图1对应的实施例中在创建初始化的camera的描述,此处不再赘述。

[0088] S206,通过worker线程根据第二camera参数更新三维场景。

[0089] 在一个实施例中,通过worker线程根据第二camera参数更新三维场景,可以包括:通过worker线程调用updateCamera()方法解析第二camera参数,并根据解析得到的数据更新第一camera,得到第四camera;通过worker线程根据第四camera,重新对场景数据进行渲染,得到更新后的三维场景。

[0090] worker线程中更新对第一camera调整后,得到第四camera,之后,在根据第四camera及其他初始化的三维场景,重新对场景数据进行渲染,进而得到重新渲染后的三维场景。

[0091] 本公开通过在worker线程完成三维场景的渲染后,通过主线程来接收对该三维场景的调整操作并记录,之后,将该调整操作通过更新后的camera参数(第二camera参数)传递给worker线程,再由worker线程根据第二camera参数对第一camera进行调整得到第四camera后,重新基于camera对场景数据进行渲染,从而得到经调整操作处理的三维场景。此种方式,避免了在主线程中重新渲染三维场景,避免了因跳调整操作而导致需要占用主线程渲染场景,提高了页面流畅度,减少了页面卡顿,提升用户体验。

[0092] 为便于理解本公开的实施例提供的场景渲染方法,下面将结合图3进行说明。如图3所示,场景渲染的过程可以包括S301至S308。其中,S301至S308的具体实现,可以参加上述图1和图2对应的实施例。

[0093] S301,在主线程中创建worker线程、创建canvas及设置监听方法。

[0094] S302,主线程向worker线程发送离屏canvas数据。

[0095] S303,worker线程根据接收到离屏canvas数据,创建初始化场景、灯光、相机,创建并渲染离屏canvas。

[0096] S304,worker线程向主线程发送相机参数。

[0097] S305,主线程根据接收到的相机参数设置主线程中相机,以及创建轨道控制器。

[0098] S306,主线程监听对三维场景的调整操作。

[0099] S307,主线程向worker线程发送调整后的相机参数。

[0100] S308,worker线程根据接收到的相机参数重新渲染三维场景。

[0101] 通过在主线程中创建worker线程及canvas,并将canvas的离屏canvas数据及场景数据发往worker线程,通过worker线程根据离屏canvas数据及场景数据渲染三维场景的方式,实现了将计算量大的三维渲染任务转移到worker线程,避免了在主线程中进行三维渲染任务,减少了浏览器阻塞和页面克顿。

[0102] 通过主线程来接收对该三维场景的调整操作并记录,之后,将该调整操作通过更新后的camera参数(第二camera参数)传递给worker线程,再由worker线程根据第二camera参数对第一camera进行调整得到第四camera后,重新基于camera对场景数据进行渲染,从而得到经调整操作处理的三维场景。此种方式,避免了在主线程中重新渲染三维场景,避免了因跳调整操作而导致需要占用主线程渲染场景,提高了页面流畅度,减少了页面卡顿,提升用户体验。

[0103] 基于同一发明构思,本公开实施例中还提供了一种场景渲染装置,如下面的实施例所述。由于该装置实施例解决问题的原理与上述方法实施例相似,因此该装置实施例的实施可以参见上述方法实施例的实施,重复之处不再赘述。

[0104] 图4示出本公开一个实施例中的场景渲染装置示意图,如图4所示,该装置包括:获取模块401,用于通过浏览器的主线程获取待渲染的场景数据;创建及转换模块402,用于在主线程中创建worker线程及画布canvas,并将canvas转换为对象格式的离屏canvas数据;发送模块403,用于worker线程管理模块,还用于通过主线程将离屏canvas数据和场景数据发往worker线程;渲染模块404,用于通过worker线程根据离屏canvas数据,渲染场景数据对应的三维场景。

[0105] 在本公开的一个实施例中,渲染模块404,用于通过worker线程解析离屏canvas数据,得到canvas参数;在worker线程中创建canvas参数对应的离屏canvas;通过worker线程在离屏canvas中创建初始化三维场景;通过worker线程在初始化三维场景中渲染场景数据,得到三维场景。

[0106] 在本公开的一个实施例中,创建及转换模块402,用于通过主线程调用offCanvas=canvas.value.transferControlToOffscreen()方法,将canvas转换为对象格式的离屏canvas数据。

[0107] 在本公开的一个实施例中,发送模块403,用于通过主线程调用worker.postMessage()方法,将离屏canvas数据和场景数据发往worker线程;装置还包括:接收模块,用于通过worker线程调用worker.onmessage()方法接收离屏canvas数据和场景数据。

[0108] 在本公开的一个实施例中,三维场景是在第一相机camera下的场景,装置还包括:

转换及发送模块405,用于在worker线程中,将第一camera的参数转换为对象格式的第一camera参数,并将第一camera参数发往主线程;构建模块406,用于在主线程中构建第一camera参数对应的第二camera;构造模块407,用于在主线程中构造第二camera对应的轨道控制器OrbitControls;记录及调整模块408,用于在主线程中通过OrbitControls获取对三维场景的调整操作,得到对第二camera调整后的第三camera;转换及发送模块402,还用于在主线程中,将第三camera的参数转换为对象格式的第二camera参数,并将第二camera参数发往worker线程;其中,渲染模块404,还用于通过worker线程根据第二camera参数更新三维场景。

[0109] 在本公开的一个实施例中,装置还包括:配置模块409,用于在主线程中使用worker.onmessage()方法配置监听方法;在主线程中调用监听方法接收第一camera参数。

[0110] 在本公开的一个实施例中,渲染模块404,用于通过worker线程调用updateCamera()方法解析第二camera参数,并根据解析得到的数据更新第一camera,得到第四camera;通过worker线程根据第四camera,重新对场景数据进行渲染,得到更新后的三维场景。

[0111] 基于同一发明构思,本公开实施例中还提供了另一种场景渲染装置,如下面的实施例所述。由于该装置实施例解决问题的原理与上述方法实施例相似,因此该装置实施例的实施可以参见上述方法实施例的实施,重复之处不再赘述。

[0112] 图5示出本公开另一个实施例中的场景渲染装置示意图,如图5所示,该装置包括:worker线程管理模块501、离屏canvas初始化模块502、交互监控模块503和离屏canvas更新渲染模块504。

[0113] worker线程管理模块501,用于负责维护worker线程,包括:新创建worker线程;在主线程中设置监听方法用于监听worker线程返回的数据;向worker线程发送数据,包括离屏canvas数据和缩放、旋转调整场景后主线程的相机参数。

[0114] 离屏canvas初始化模块502,用于初始化时,worker线程中创建离屏canvas画布;worker线程接收并解析离屏canvas数据主线程传来的场景数据,读取数据中的canvas画布的长、宽等参数;初始化创建场景、灯光、相机,开始进行画布渲染。

[0115] 交互监控模块503,用于监控用户对三维场景的操作并发送给worker线程。

[0116] 离屏canvas更新渲染模块504,用于worker线程中,根据接收到的主线程中新相机参数,调整worker线程相机的位置、朝向、缩放等参数;依据新相机参数,设置worker线程中的相机;更新相机的投影矩阵;同时重新渲染离屏画布,完成对调整操作的同步更新渲染。

[0117] 所属技术领域的技术人员能够理解,本公开的各个方面可以实现为系统、方法或程序产品。因此,本公开的各个方面可以具体实现为以下形式,即:完全的硬件实施方式、完全的软件实施方式(包括固件、微代码等),或硬件和软件方面结合的实施方式,这里可以统称为“电路”、“模块”或“系统”。

[0118] 下面参照图6来描述根据本公开的这种实施方式的电子设备600。图6显示的电子设备600仅仅是一个示例,不应对本公开实施例的功能和使用范围带来任何限制。

[0119] 如图6所示,电子设备600以通用计算设备的形式表现。电子设备600的组件可以包括但不限于:上述至少一个处理单元610、上述至少一个存储单元620、连接不同系统组件(包括存储单元620和处理单元610)的总线630。

[0120] 其中,所述存储单元存储有程序代码,所述程序代码可以被所述处理单元610执

行,使得所述处理单元610执行本说明书上述“具体实施方式”部分中描述的根据本公开各种示例性实施方式的步骤。

[0121] 存储单元620可以包括易失性存储单元形式的可读介质,例如随机存取存储单元(RAM)6201和/或高速缓存存储单元6202,还可以进一步包括只读存储单元(ROM)6203。

[0122] 存储单元620还可以包括具有一组(至少一个)程序模块6205的程序/实用工具6204,这样的程序模块6205包括但不限于:操作系统、一个或者多个应用程序、其它程序模块以及程序数据,这些示例中的每一个或某种组合中可能包括网络环境的实现。

[0123] 总线630可以为表示几类总线结构中的一种或多种,包括存储单元总线或者存储单元控制器、外围总线、图形加速端口、处理单元或者使用多种总线结构中的任意总线结构的局域总线。

[0124] 电子设备600也可以与一个或多个外部设备640(例如键盘、指向设备、蓝牙设备等)通信,还可与一个或者多个使得用户能与该电子设备600交互的设备通信,和/或与使得该电子设备600能与一个或多个其它计算设备进行通信的任何设备(例如路由器、调制解调器等等)通信。这种通信可以通过输入/输出(I/O)接口650进行。并且,电子设备600还可以通过网络适配器660与一个或者多个网络(例如局域网(LAN),广域网(WAN)和/或公共网络,例如因特网)通信。如图6所示,网络适配器660通过总线630与电子设备600的其它模块通信。应当明白,尽管图中未示出,可以结合电子设备600使用其它硬件和/或软件模块,包括但不限于:微代码、设备驱动器、冗余处理单元、外部磁盘驱动阵列、RAID系统、磁带驱动器以及数据备份存储系统等。

[0125] 通过以上的实施方式的描述,本领域的技术人员易于理解,这里描述的示例实施方式可以通过软件实现,也可以通过软件结合必要的硬件的方式来实现。因此,根据本公开实施方式的技术方案可以以软件产品的形式体现出来,该软件产品可以存储在一个非易失性存储介质(可以是CD-ROM,U盘,移动硬盘等)中或网络上,包括若干指令以使得一台计算设备(可以是个人计算机、服务器、终端装置、或者网络设备等)执行根据本公开实施方式的方法。

[0126] 在本公开的示例性实施例中,还提供了一种计算机可读存储介质,该计算机可读存储介质可以是可读信号介质或者可读存储介质。其上存储有能够实现本公开上述方法的程序产品。在一些可能的实施方式中,本公开的各个方面还可以实现为一种程序产品的形式,其包括程序代码,当所述程序产品在终端设备上运行时,所述程序代码用于使所述终端设备执行本说明书上述“具体实施方式”部分中描述的根据本公开各种示例性实施方式的步骤。

[0127] 本公开中的计算机可读存储介质的更具体的例子可以包括但不限于:具有一个或多个导线的电连接、便携式计算机磁盘、硬盘、随机访问存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、光纤、便携式紧凑磁盘只读存储器(CD-ROM)、光存储器件、磁存储器件、或者上述的任意合适的组合。

[0128] 在本公开中,计算机可读存储介质可以包括在基带中或者作为载波一部分传播的数据信号,其中承载了可读程序代码。这种传播的数据信号可以采用多种形式,包括但不限于电磁信号、光信号或上述的任意合适的组合。可读信号介质还可以是可读存储介质以外的任何可读介质,该可读介质可以发送、传播或者传输用于由指令执行系统、装置或者器件

使用或者与其结合使用的程序。

[0129] 可选地,计算机可读存储介质上包含的程序代码可以用任何适当的介质传输,包括但不限于无线、有线、光缆、RF等等,或者上述的任意合适的组合。

[0130] 在具体实施时,可以以一种或多种程序设计语言的任意组合来编写用于执行本公开操作的程序代码,所述程序设计语言包括面向对象的程序设计语言—诸如Java、C++等,还包括常规的过程式程序设计语言—诸如“C”语言或类似的程序设计语言。程序代码可以完全地在用户计算设备上执行、部分地在用户设备上执行、作为一个独立的软件包执行、部分在用户计算设备上部分在远程计算设备上执行、或者完全在远程计算设备或服务器上执行。在涉及远程计算设备的情形中,远程计算设备可以通过任意种类的网络,包括局域网(LAN)或广域网(WAN),连接到用户计算设备,或者,可以连接到外部计算设备(例如利用因特网服务提供商来通过因特网连接)。

[0131] 在本公开的示例性实施例中,还提供了一种计算机程序产品,计算机程序产品包括计算机程序或计算机指令,计算机程序或计算机指令由处理器加载并执行,以使计算机实现本说明书上述“具体实施方式”部分中描述的根据本公开各种示例性实施方式的步骤。

[0132] 应当注意,尽管在上文详细描述中提及了用于动作执行的设备的若干模块或者单元,但是这种划分并非强制性的。实际上,根据本公开的实施方式,上文描述的两个或更多模块或者单元的特征和功能可以在一个模块或者单元中具体化。反之,上文描述的一个模块或者单元的特征和功能可以进一步划分为由多个模块或者单元来具体化。

[0133] 此外,尽管在附图中以特定顺序描述了本公开中方法的各个步骤,但是,这并非要求或者暗示必须按照该特定顺序来执行这些步骤,或是必须执行全部所示的步骤才能实现期望的结果。附加的或备选的,可以省略某些步骤,将多个步骤合并为一个步骤执行,以及/或者将一个步骤分解为多个步骤执行等。

[0134] 通过以上实施方式的描述,本领域的技术人员易于理解,这里描述的示例实施方式可以通过软件实现,也可以通过软件结合必要的硬件的方式来实现。因此,根据本公开实施方式的技术方案可以以软件产品的形式体现出来,该软件产品可以存储在一个非易失性存储介质(可以是CD-ROM,U盘,移动硬盘等)中或网络上,包括若干指令以使得一台计算设备(可以是个人计算机、服务器、移动终端、或者网络设备等)执行根据本公开实施方式的方法。

[0135] 本领域技术人员在考虑说明书及实践这里公开的发明后,将容易想到本公开的其它实施方案。本公开旨在涵盖本公开的任何变型、用途或者适应性变化,这些变型、用途或者适应性变化遵循本公开的一般性原理并包括本公开未公开的本技术领域中的公知常识或惯用技术手段。说明书和实施例仅被视为示例性的,本公开的真正范围由所附的权利要求指出。

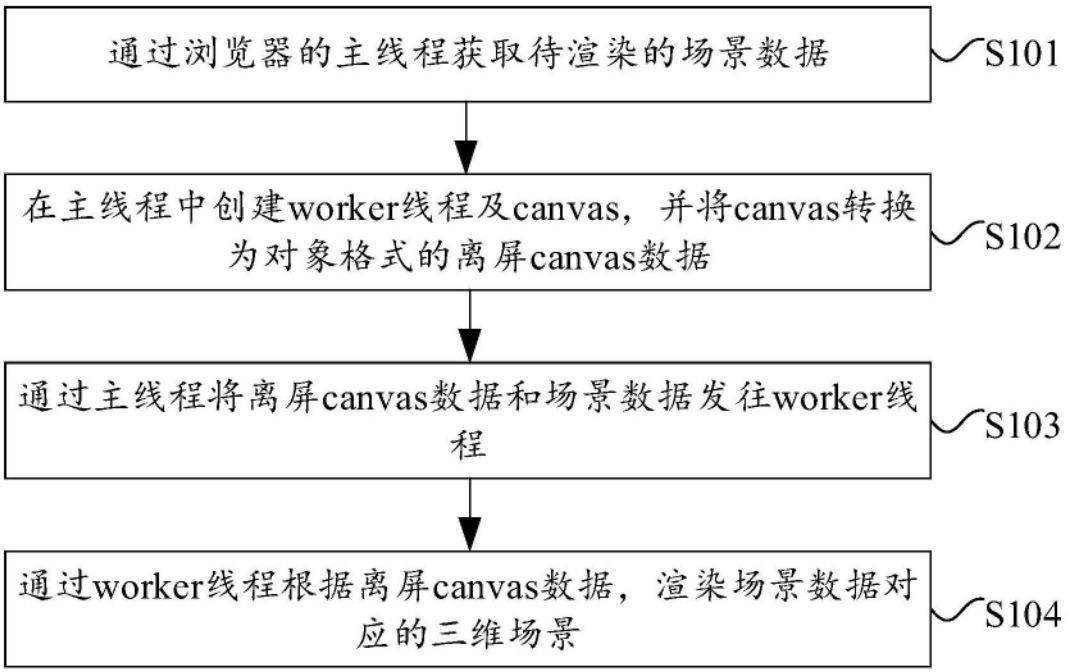

通过浏览器的主线程获取待渲染的场景数据
S101
在主线程中创建worker线程及canvas，并将canvas转换为对象格式的离屏canvas数据
S102
通过主线程将离屏canvas数据和场景数据发往worker线程
S103
通过worker线程根据离屏canvas数据，渲染场景数据对应的三维场景
S104

图1

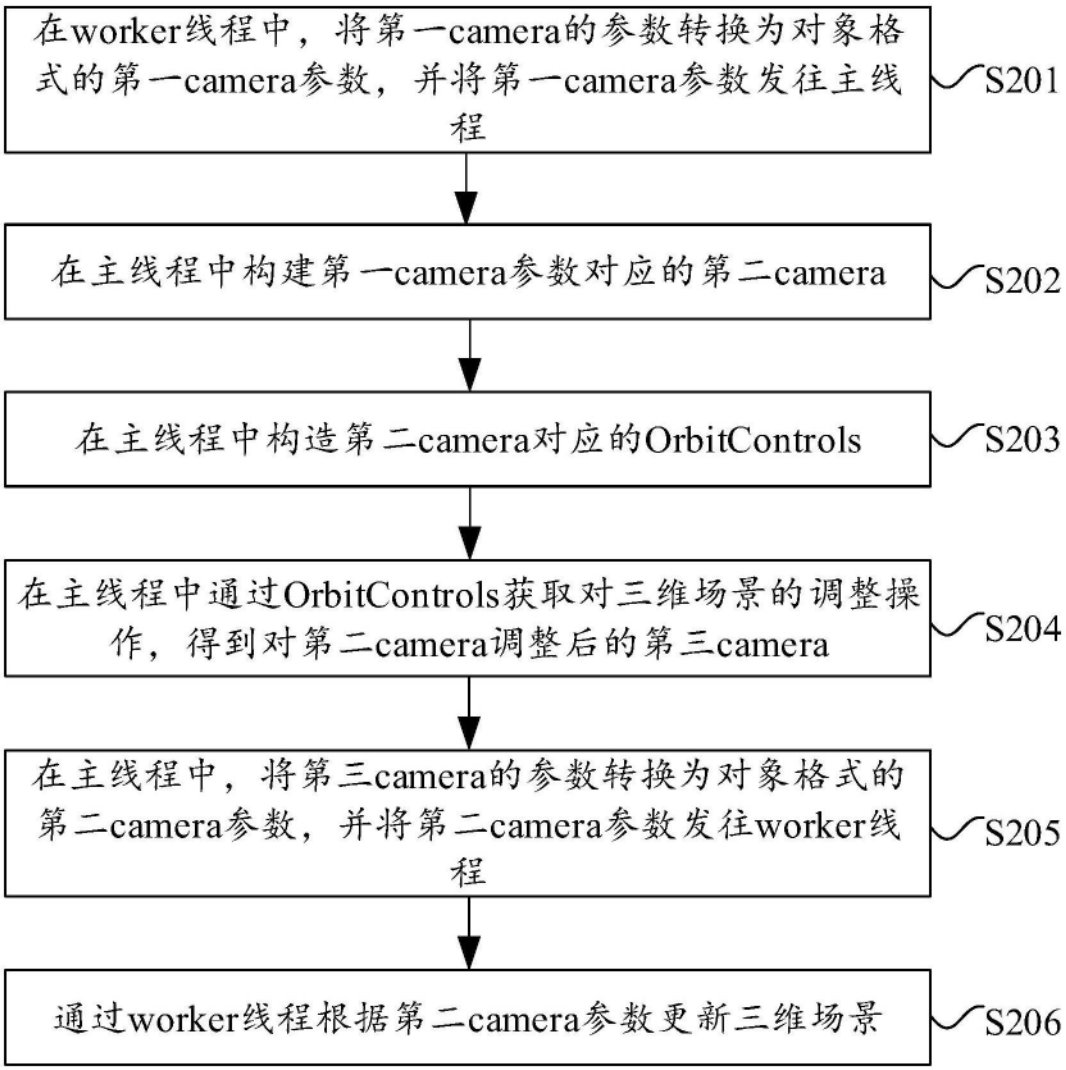

在worker线程中，将第一camera的参数转换为对象格式的第一camera参数，并将第一camera参数发往主线程
S201
在主线程中构建第一camera参数对应的第二camera
S202
在主线程中构造第二camera对应的OrbitControls
S203
在主线程中通过OrbitControls获取对三维场景的调整操作，得到对第二camera调整后的第三camera
S204
在主线程中，将第三camera的参数转换为对象格式的第二camera参数，并将第二camera参数发往worker线程
S205
通过worker线程根据第二camera参数更新三维场景
S206

图2

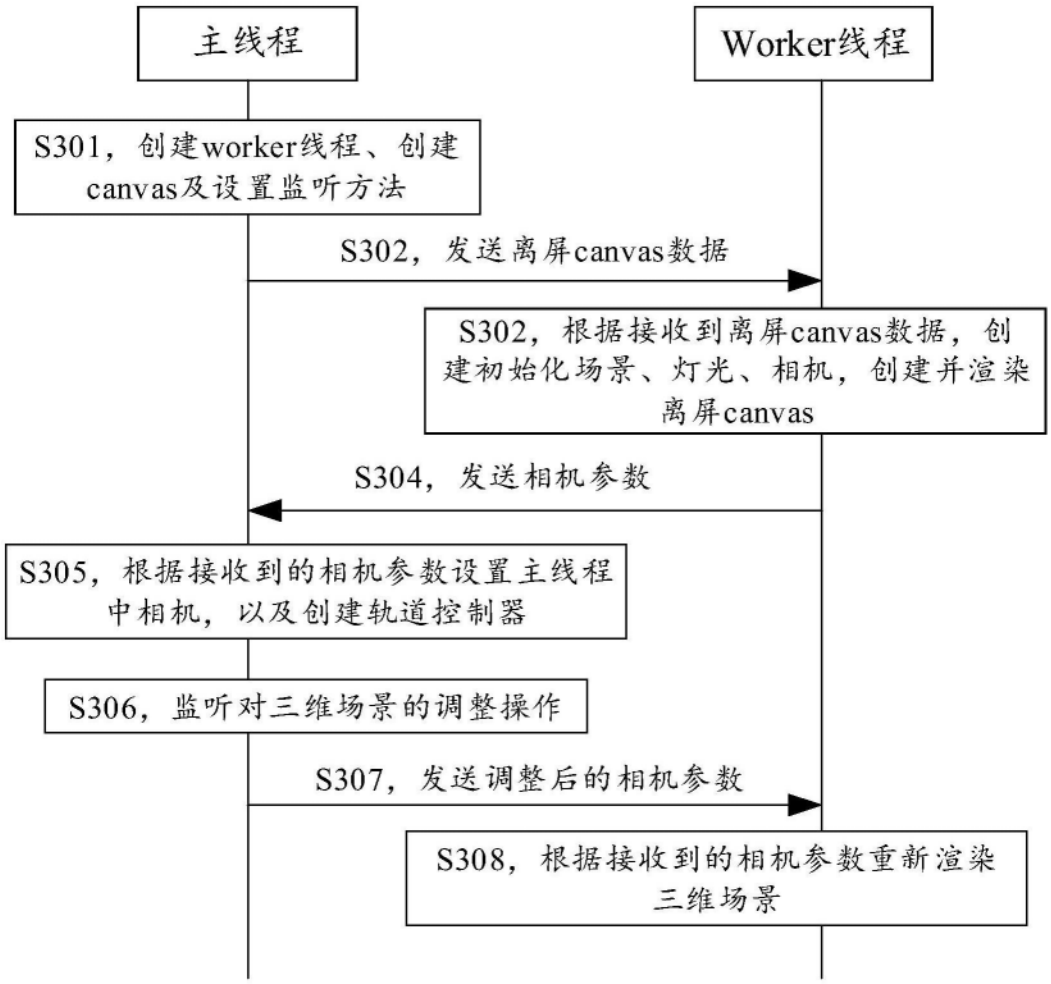

主线程
Worker线程
S301，创建worker线程、创建canvas及设置监听方法
S302，发送离屏canvas数据
S302，根据接收到离屏canvas数据，创建初始化场景、灯光、相机，创建并渲染离屏canvas
S304，发送相机参数
S305，根据接收到的相机参数设置主线程中相机，以及创建轨道控制器
S306，监听对三维场景的调整操作
S307，发送调整后的相机参数
S308，根据接收到的相机参数重新渲染三维场景

图3

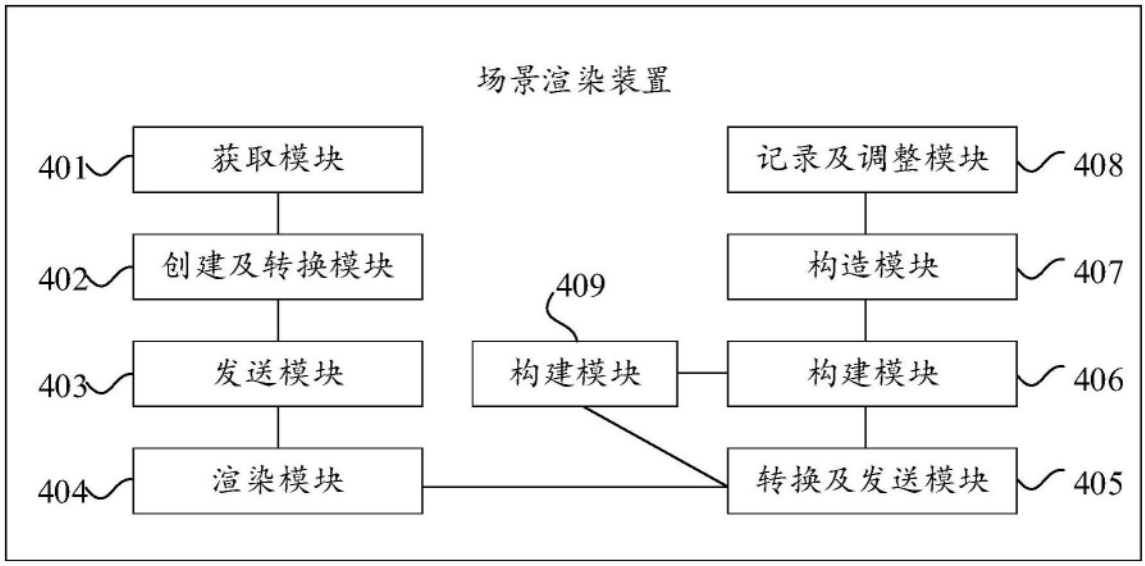

场景渲染装置
401
获取模块
402
创建及转换模块
403
发送模块
404
渲染模块
409
构建模块
记录及调整模块
408
构造模块
407
构建模块
406
转换及发送模块
405

图4

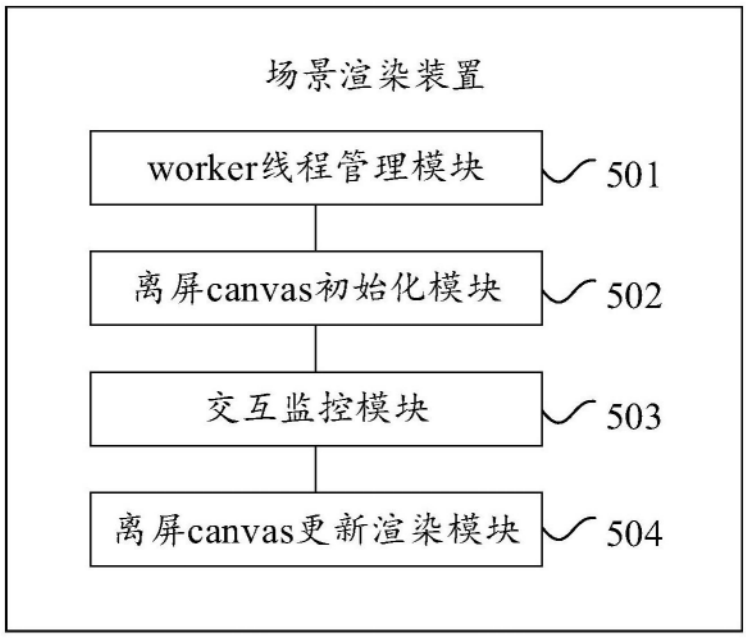

场景渲染装置
worker线程管理模块
501
离屏canvas初始化模块
502
交互监控模块
503
离屏canvas更新渲染模块
504

图5

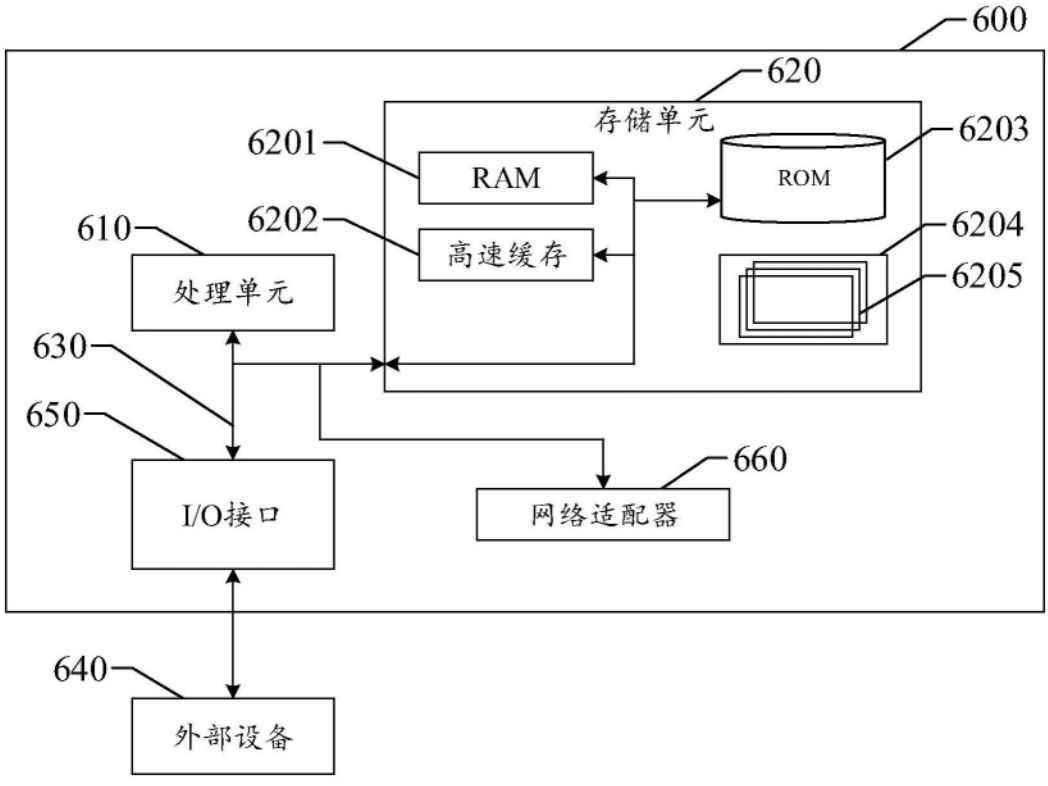

600
620
存储单元
6201
RAM
6203
ROM
6202
高速缓存
6204
6205
610
处理单元
630
650
I/O接口
660
网络适配器
640
外部设备

图6