(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 110969685 A
(43)申请公布日 2020.04.07

(21)申请号 201910920484.4

(22)申请日 2019.09.27

(30)优先权数据
62/738,866 2018.09.28 US

(71)申请人 苹果公司
地址 美国加利福尼亚

(72)发明人 C•J•怀特 R•W•拉莫洛
P-K•恩格斯塔德 I•加文科夫
M•斯托尔 周阳

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 吴信刚

(51)Int.Cl.
G06T 15/00(2011.01)

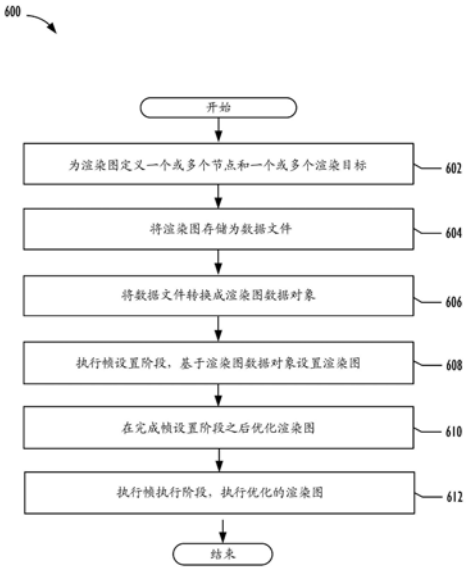
权利要求书2页 说明书11页 附图5页

(54)发明名称

使用渲染图的可定制渲染管线

(57)摘要

本公开涉及使用渲染图的可定制渲染管线。描述了用于数据驱动渲染图的系统、方法和计算机可读介质。渲染图系统定义渲染图的一个或多个节点以及与节点相关联的一个或多个渲染目标。节点包括一个或多个函数，用于定义和解析用于识别渲染目标的目标句柄。渲染图系统定义节点和渲染目标之间的一个或多个连接。节点和渲染目标之间的连接形成渲染图。渲染图系统将渲染图存储为数据文件，并利用渲染图形API将数据文件转换成渲染图数据对象。渲染图系统执行帧设置阶段，基于渲染图数据对象设置帧的渲染图。



CN 110969685 A

1.一种非暂态程序存储设备,所述非暂态程序存储设备能够由一个或多个处理器读取并且包括存储在其上的指令,以使得所述一个或多个处理器:

定义渲染图的一个或多个节点以及与所述节点相关联的一个或多个渲染目标,其中所述节点包括用于定义和解析用于识别渲染目标的目标句柄的一个或多个函数;

定义所述节点和渲染目标之间的一个或多个连接,其中所述节点和渲染目标之间的所述连接形成所述渲染图;

将所述渲染图存储为数据文件;

用渲染图形API将所述数据文件转换成渲染图数据对象;以及

执行帧设置阶段,所述帧设置阶段基于所述渲染图数据对象来设置帧的所述渲染图。

2.根据权利要求1所述的非暂态程序存储设备,其中所述指令还使得所述处理器在完成所述帧设置阶段之后优化所述渲染图。

3.根据权利要求2所述的非暂态程序存储设备,其中所述指令还使得所述处理器执行帧执行阶段,所述帧执行阶段执行经优化的渲染图。

4.根据权利要求2所述的非暂态程序存储设备,其中使所述处理器优化所述渲染图的所述指令还包括用于以下操作的指令:

分析渲染目标句柄依赖关系;以及

基于分析所述渲染目标句柄的依赖关系,从执行中去除未使用的渲染目标和未使用的节点。

5.根据权利要求2所述的非暂态程序存储设备,其中使所述处理器优化所述渲染图的所述指令还包括用于以下操作的指令:

分析在所述帧设置阶段期间声明的所述渲染目标句柄;

确定一组节点是从单个渲染目标读取还是向单个渲染目标写入;

为所述一组节点分配命令编码器。

6.根据权利要求1所述的非暂态程序存储设备,其中所述渲染图表示渲染管线。

7.根据权利要求1所述的非暂态程序存储设备,其中所述渲染图对应于用于渲染所述帧的渲染通道。

8.根据权利要求1所述的非暂态程序存储设备,其中所述节点中的一个节点具有允许编写代码的定制节点类型。

9.一种系统,包括:

存储器,所述存储器包含指令;以及

至少一个处理器,所述至少一个处理器耦接到存储器,其中所述指令在被执行时使得所述至少一个处理器:

定义渲染图的一个或多个节点以及与所述节点相关联的一个或多个渲染目标,其中所述节点包括用于定义并解析用于识别渲染目标的目标句柄的一个或多个函数;

定义所述节点和渲染目标之间的一个或多个连接,其中所述节点和渲染目标之间的所述连接形成所述渲染图;

将所述渲染图存储为数据文件;

用渲染图形API将所述数据文件转换成渲染图数据对象;以及

执行帧设置阶段,所述帧设置阶段基于所述渲染图数据对象来设置帧的所述渲染图。

10.根据权利要求9所述的系统,其中所述指令还使得所述至少一个处理器在完成所述帧设置阶段之后优化所述渲染图。

11.根据权利要求10所述的系统,其中所述指令还使得所述至少一个处理器执行帧执行阶段,所述帧执行阶段执行经优化的渲染图。

12.根据权利要求10所述的系统,其中使所述至少一个处理器优化所述渲染图的所述指令还包括使得所述至少一个处理器执行以下操作的指令:

分析渲染目标句柄依赖关系;以及

基于分析所述渲染目标句柄的依赖关系,从执行中去除未使用的渲染目标和未使用的节点。

13.根据权利要求10所述的系统,其中使所述至少一个处理器优化渲染图的所述指令还包括使得所述至少一个处理器执行以下操作的指令:

分析在所述帧设置阶段期间声明的所述渲染目标句柄;

确定一组节点是从单个渲染目标读取还是向单个渲染目标写入;

为所述一组节点分配命令编码器。

14.根据权利要求9所述的系统,其中所述渲染图表示渲染管线。

15.根据权利要求9所述的系统,其中所述渲染图对应于用于渲染所述帧的渲染通道。

16.根据权利要求9所述的系统,其中所述节点中的一个节点具有允许编写代码的定制节点类型。

17.一种计算机实现的方法,包括:

定义渲染图的一个或多个节点以及与所述节点相关联的一个或多个渲染目标,其中所述节点包括用于定义并解析用于识别渲染目标的目标句柄的一个或多个函数;

定义所述节点和渲染目标之间的一个或多个连接,其中所述节点和渲染目标之间的所述连接形成所述渲染图;

将所述渲染图存储为数据文件;

用渲染图形API将所述数据文件转换成渲染图数据对象;以及

执行帧设置阶段,所述帧设置阶段基于所述渲染图数据对象来设置帧的所述渲染图。

18.根据权利要求17所述的方法,还包括在完成所述帧设置阶段之后优化所述渲染图。

19.根据权利要求18所述的方法,还包括执行帧执行阶段,所述帧执行阶段执行经优化的渲染图。

20.根据权利要求18所述的方法,还包括:

分析渲染目标句柄依赖关系;以及

基于分析所述渲染目标句柄的依赖关系,从执行中去除未使用的渲染目标和未使用的节点。

使用渲染图的可定制渲染管线

技术领域

[0001] 本公开整体涉及图形处理领域。更具体地，但不作为限制，本公开涉及实现自动管理渲染目标的可定制渲染管线。

背景技术

[0002] 图形处理器单元(GPU)对于处理当今计算机、移动设备和其他计算系统中的数据并行图形任务变得非常重要。开发者还通过让GPU以并行方式执行非图形数据任务来利用GPU的并行功能。供应商和标准化组织已经创建了由于开发者编程交互的高层级而使执行数据并行任务更易于编程的应用编程接口(API)。例如，存在各种低级API(库和框架)，它们靠近图形硬件，通常使用更高级别API的输出。特别地，更高级别的API通常为应用程序准备程序代码，并将程序代码呈现给较低级别的API进行处理。

[0003] 今天的图形处理领域包括改善实时图形渲染。为了实现实时图形渲染，现代渲染引擎通常需要足够灵活以允许自定义编程和一定程度的可配置性以形成复杂的渲染管线。渲染通常从应用程序开始，使图形发生变化，从而导致场景发生变化。为了生成场景的帧，渲染引擎可以在将内容提交到帧缓冲器之前采用若干渲染通道。例如，效果可以顺序地应用于图形元素，诸如照明、阴影、反射、镜面照明等。在另一个示例中，可以采用多个渲染通道来创建稍后要合成的单个帧的片段或子集以形成整个帧。使用多个渲染通道可能导致延迟，该延迟根据系统的速度以及图形的复杂性和变化率而变化。例如，在游戏应用程序中，图形的范围和复杂性可是资源需求的并且不同于其他图形应用程序(例如，三维(3D)建模)。具有足够灵活的API以产生适应各种系统和/或图形应用程序的渲染管线可能有益于改善处理时间和延迟。

发明内容

[0004] 在一个实施方案中，提供了用于数据驱动渲染图的方法。示例性方法为渲染图定义一个或多个节点以及与节点相关联的一个或多个渲染目标。节点包括一个或多个函数，用于定义和解析用于识别渲染目标的目标句柄。示例性方法定义节点和渲染目标之间的一个或多个连接。节点和渲染目标之间的连接形成渲染图。示例性方法将渲染图存储为数据文件，并使用渲染图形API将数据文件转换为渲染图数据对象。示例性方法执行帧设置阶段，基于渲染图数据对象设置帧的渲染图。

[0005] 在另一个实施方案中，一种系统，所述系统包括包括指令的存储器和耦接到存储器的至少一个处理器，其中所述指令当被执行时使得所述至少一个处理器为渲染图定义一个或多个节点和与节点相关联的一个或多个渲染目标。节点包括一个或多个函数，用于定义和解析用于识别渲染目标的目标句柄。处理器定义节点和渲染目标之间的一个或多个连接。节点和渲染目标之间的连接形成渲染图。处理器将渲染图存储为数据文件，并利用渲染图形API将数据文件转换为渲染图数据对象。处理器执行帧设置阶段，基于渲染图数据对象设置帧的渲染图。

[0006] 在其他实施方案中，上述的每一个方法及其变型形式都可被实施为一系列计算机可执行指令。此类指令可以使用任何一种或多种方便的编程语言。此类指令可以被收集到引擎和/或程序中，并且可以被存储在计算机系统或其他可编程控制设备可读和可执行的任何介质中。从以下结合附图和权利要求的详细描述中，将更清楚地理解这些和其他特征。

附图说明

[0007] 为了更完整地理解本公开，现在参考以下结合附图和详细描述的简要描述，其中相同的附图标记表示相同的部分。

[0008] 图1是本公开的具体实施可在其中操作的图形处理流的示意图。

[0009] 图2示出了处理为应用程序生成的多个帧的处理器的实施方案。

[0010] 图3示出了用于基于渲染图API处理多个帧的处理器的实施方案。

[0011] 图4示出了渲染帧的图形表示，其包括使用渲染图API构建的多个渲染图。

[0012] 图5表示渲染图系统，其采用渲染图API来数据驱动和/或代码驱动渲染图。

[0013] 图6示出了示出数据驱动渲染帧的渲染图的图形处理操作的流程图。

[0014] 图7是其中可以操作本公开的实施方案的计算系统的简化框图。

具体实施方式

[0015] 首先应该理解，尽管下面提供了一个或多个实施方案的例示性具体实施，但是所公开的系统和/或方法可以使用任何数量的技术来实现，无论是当前已知的还是现有的。本公开绝不应限于以下所示的例示性具体实施、附图和技术，包括本文所示和所述的示例性设计和具体实施实施方式，而是可以在所附权利要求的范围内以及它们的等同物的全部范围内进行修改。

[0016] 本公开包括创建模块化和可定制的渲染管线的各种示例实施方案，其能够在代码中创作、由数据驱动或两者。特别地，渲染图应用程序接口(API)能够通过支持开发者在不同编程级别创建渲染管线的能力来生成数据驱动和代码驱动的渲染图两者。在顶级编程级别，渲染图API能够通过使开发者界面具有可视图形编辑器来生成具有数据驱动渲染图的渲染帧。可视图形编辑器是用于开发者创建和创作所需渲染帧和/或渲染图的可视图形表示的用户界面(UI)。可视图形编辑器还能够为开发者提供钩子，以将渲染图资产附加到渲染帧的部分。基于可视图形表示，渲染图API生成渲染图资产(例如，数据文件)，其指定后端渲染引擎如何生成渲染图。在低于顶级编程级别的下一个编程级别，渲染图API通过允许开发者使用代码创建形成渲染帧的一个或多个渲染图来提供对后端渲染引擎的部分的开发者访问。例如，渲染图API使用渲染帧程序对象公开和处理渲染图的输入和/或输出。在下一个较低的编程级别，渲染图API提供对后端渲染引擎的访问，以便开发者能够编写用于管理渲染图的节点集合的代码。每个节点由设置和执行函数组成，其声明目标使用和依赖关系(例如，设置函数)并将目标句柄解析为纹理并执行图形命令(例如，执行函数)。

[0017] 出于本公开的目的，术语“渲染图资产”指的是指定后端渲染引擎如何生成特定渲染图的数字文件。在一个或多个实施方案中，“渲染图资产”表示为JavaScript对象表示法(JSON)数据文件。“渲染图资产”类似于其他基于图形的资产(例如，网孔资产或纹理资产)，因为开发者能够为了期望的目的重用和/或修改“渲染图资产”。如本文所用，术语“渲染图”

表示为渲染管线执行渲染(例如,着色器)和/或计算操作(例如,计算内核)的节点的集合。在一个或多个实施方案中,“渲染图”表示执行渲染管线的渲染通道。附加地或另选地,“渲染图”表示将不同场景对象分离成单独图像的渲染层。

[0018] 如本文所用,术语“渲染目标”指的是用于存储与在诸如GPU的图形处理器上执行图形命令有关的数据的存储空间的分配。例如,术语“渲染目标”指的是处理器和/或图形处理器在创建和执行图形命令时访问和/或修改的任何存储空间。例如,术语“渲染目标”包括图形API资源(例如,Direct3D®资源),诸如缓冲器和纹理。缓冲器表示可包含数据(诸如顶点、着色器和计算状态数据)的无格式存储器的分配。纹理表示用于存储格式化图像数据的存储器分配。在一个或多个实施方案中,“渲染目标”表示用于执行一个或多个图形处理任务的临时缓冲器。出于本公开的目的,术语“目标句柄”指的是对“渲染目标”的抽象引用。

[0019] 图1是本公开的具体实施可在其中操作的图形处理流100的示意图。图1示出了应用程序产生图形102可以发出对高级图形框架104分析和处理的场景中的帧的图形请求。应用程序产生图形102的示例包括游戏应用程序、3D建模应用程序、网络浏览器应用程序和文档查看器应用程序(例如,便携式文档格式(pdf)查看器)。

[0020] 高级图形框架104与低级图形框架106交互以管理帧之间的变化(例如,屏幕上的图形的移动)。低级图形框架将图形请求传递到硬件驱动器108,之后硬件(例如,图形处理器)可以处理数据并填充帧缓冲器。虽然图1明确地说明了这一点,存在许多到显示设备的软件路径,包括图1中未示出的层和其他框架,但是通用软件架构选项在本领域中是众所周知的。

[0021] 在一个或多个实施方案中,低级图形框架106可以与提供GPU操作的粒度控制的库相关联。特别地,一些实施方案具有低级图形框架106,其具有以下能力或特征中的一者或多者:GPU状态向量的直接控制;便于直接确定/选择被提交给硬件的命令缓冲器(编码和提交);能够延迟提供动作(例如,能够延迟提交或并行提交命令缓冲器);提供标准库;和/或提供GPU的粒度控制(例如,组织、处理以及图形和计算命令的提交的控制以及这些命令的相关联数据和资源的管理)。在一些实施方案中,低级图形框架可以为或可以包括标准或已发布的API或库。低级图形框架106的示例包括Mantle或Direct3D®。

[0022] 通过公布的低级图形框架106紧密地控制GPU的能力提供了可以促进更有序的渲染路径同时允许应用程序102使用高级图形框架104来与系统的图形能力交接的优点。在一个或多个实施方案中,高级图形框架104表示渲染图API,其基于数据驱动和/或代码驱动的操作生成渲染图资产。对于场景的每个帧,渲染图API能够将渲染分解为互连渲染图的集合。每个渲染图包括节点集合,其中每个节点由与渲染或计算机操作相关联的设置和执行函数组成。在一些实施方案中,渲染图是数据驱动的,其中高级图形框架104(例如,渲染图API)分析数据并生成以低级GPU界面语言表达的图形命令,诸如由低级图形框架106促进的图形命令(例如,Direct3D®)。其他实施方案可以包括代码驱动的渲染图和/或渲染图内的节点以生成图形命令。

[0023] 图2示出了处理器200的实施方案,处理器200处理为应用程序生成的多个帧204。处理器200表示能够处理来自一个或多个数据源(诸如存储器)的数据的可编程硬件设备。在图2中,处理器200是通用处理器(例如,中央处理单元(CPU)或微控制器),其不被定制成

执行特定操作(例如,过程、计算、功能或任务),而是被构建为执行一般计算操作。虽然图2示出了处理器200是通用处理器,但是其他类型的处理器能够包括被定制成执行特定操作(例如,过程、计算、功能或任务)的专用处理器。专用处理器的示例包括GPU、浮点处理单元(FPU)、DSP、FPGA、专用集成电路(ASIC)、和嵌入式处理器(例如通用串行总线(USB)控制器)。

[0024] 处理器200编码并提交将帧204A-204D渲染到图形处理器(图2中未示出)的图形命令。图形处理器是用于执行图形处理操作的专用处理器。“图形处理器”的示例包括GPU、DSP、FPGA和/或模拟GPU的CPU。在一个或多个实施方案中,图形处理器还能够执行通用处理器能够执行的非专用操作。这些一般计算操作的示例是与计算内核相关联的计算命令。计算内核是指用于图形处理器(例如,GPU、DSP或FPGA)的程序。在图形处理操作的上下文中,用于图形处理器的程序被分类为计算内核或着色器。计算内核是指执行一般计算操作(例如,计算图形命令)的图形处理器的程序,术语着色器是指执行图形操作(例如,渲染图形命令)的图形处理器的程序。出于本公开的目的,术语计算内核与术语内核或操作系统内核不同并且不应与它们混淆。

[0025] 每个帧204A-204D是图形处理器渲染的单个场景的表示。在图2中,场景内容能够从帧204(例如,帧204A)改变为帧204(帧204B)。在一个或多个实施方案中,为了渲染每个帧204(例如,帧204B)的改变,图形处理器通过执行多个渲染通道来划分帧204。通过利用多个渲染通道,图形处理器将每个帧204解构为多个分量图像,这些分量图像能够在重新组合之前被独立地改变。例如,渲染通道能够将帧的场景的不同特征(例如,阴影、高光或反射)分离成单独的分量图像。

[0026] 在图2中,基于不同的相机渲染层206A和206B划分帧204B,其中每个相机渲染层206对应于场景的特定相机视角。相机视角可以基于各种参数,诸如位置、旋转、比例、视野和剪裁。具有两个相机渲染层206A和206B不会使图形处理器两次绘制整个场景,而是根据不同的相机视角绘制场景。换句话讲,每个相机渲染层206输出对于其特定相机视角可见的场景对象。由于不同的相机视角,相机渲染层206A能够产生未在相机渲染层206B中渲染的一个或多个场景对象,反之亦然。未渲染的场景对象表示根据相机渲染层206B的相机视角不可视的场景对象。

[0027] 每个相机渲染层206A和206B能够进一步划分为不同的图形操作(例如,渲染操作)。图2示出了相机渲染层206A和206B都被分解为渲染不透明网孔操作208A和渲染天空盒操作210A。渲染不透明网孔操作208A包括用于绘制网孔的一个或多个渲染命令。渲染天空盒操作210还包括用于绘制天空盒的一个或多个渲染命令。例如,天空盒操作210能够在场景周围渲染天空盒,以根据特定的相机视角在地平线上创建复杂的场景。本领域普通技术人员知道,相机渲染层206A和206B能够包括与渲染特定相机视角的场景相关的其他图形操作。

[0028] 图3示出了用于基于渲染图API处理多个帧204的处理器200的实施方案。如图3所示,渲染图API将框204B细分为不同的API程序对象。与图2相比较,图3示出了渲染图API表示帧204B,其具有渲染帧程序对象302和分别具有渲染图程序对象304A和304B的相机渲染层206A和206B。相机渲染层206A和206B内的不同图形操作被表示为渲染图节点306A-306D。参照图2,渲染图API分别将渲染不透明网孔操作208A和208B表示为渲染图节点306A和

306C。图2中的渲染天空盒操作210A和210B分别表示为渲染图节点306B和306D。

[0029] 图4示出了渲染帧400的图形表示，其包括使用渲染图API构建的多个渲染图402A-402C。每个渲染图402表示渲染通道或渲染层能够执行的渲染管线。在图4中，渲染帧400包括顺序连接在一起的三个不同的渲染图402A-402C。阴影渲染图402A表示用于阴影通道的渲染管线；相机渲染图402B表示用于相机渲染层的渲染管线；并且后处理渲染图402C表示用于后处理通道的渲染管线。为了生成渲染帧400，渲染图API将阴影渲染图402A顺序地连接到相机渲染图402B，然后相机渲染图402B连接到后处理渲染图402C。具体地，阴影渲染图402A输出阴影缓冲器410A，然后阴影缓冲器410A输入到相机渲染图402B。相机渲染图402B使用阴影缓冲器410A输出OutColor缓冲器410B，然后OutColor缓冲器410B成为后处理渲染图402C的输入。后处理渲染图402C使用OutColor缓冲器410B输出到颜色缓冲器410C。

[0030] 在每个渲染图402A-402C内包括一个或多个节点404A-404I。每个节点404表示为给定的渲染管线执行一个或多个图形命令的图形操作。使用图4作为示例，阴影渲染图402A包括阴影节点404A；相机渲染图402B包括不透明节点404B、调试节点404C、天空盒节点404D、透明节点404E和文本节点404F；并且后处理渲染图402C包括浓郁度下采样节点404G、亮度计算节点404H和后处理组合节点404I。阴影节点404A、不透明节点404B和透明节点404E具有指示生成和编码绘制网孔图形命令的图形操作的网孔节点类型。调试节点404C、天空盒节点404D和测试节点404F具有自定义节点类型，其执行生成图形命令的开发者定制操作。例如，开发者可能已经编写或提供了代码，用于执行自定义图形操作，生成用于渲染天空盒的图形命令。浓郁度下采样节点404G、亮度计算节点404H和后处理组合节点404I表示全屏类型节点，其表示用于生成渲染到整个屏幕的图形命令的图形操作。

[0031] 图4还示出了用于生成渲染帧400的处理资源被示为阴影框。渲染图402A-402C包括节点404A-404I可以使用的渲染目标406A-406G。在图4中，渲染目标406A-406G表示在将图形数据(例如，像素数据)输出到渲染图输出缓冲器(诸如，阴影缓冲器410A、OutColor缓冲器410B和颜色缓冲器410C)之前用于存储图形数据的中间存储器缓冲器。对于图4，阴影节点404A生成用于阴影标测图406A的数据；相机渲染图402B内的节点404B-404F将图像信息(例如，像素数据)输出到输出颜色缓冲器406B。深度模板406C可以表示深度缓冲器和/或模板缓冲器，用于跟踪屏幕上的像素深度以及哪些片段应该绘制哪些片段不应该绘制。其他渲染目标406E、406F和406G对应于后处理缓冲器。阴影缓冲器410A、OutColor缓冲器410B和颜色缓冲器410C表示另一个渲染图402能够利用的处理资源。

[0032] 在一个或多个实施方案中，开发者使用可视图形编辑器来创建渲染帧400。换句话讲，开发者能够使用某种类型的人类可读表示(例如，可视图形表示)来定义不同的渲染管线以渲染帧，而不是编写代码以生成渲染图402A-402C。可视图形编辑器还可提供用于将渲染图资产附接到渲染帧400的钩子。在开发者生成渲染帧400的人类可读表示之后，渲染图API能够将人类可读表示转换为一个或多个数据文件，其指定数据如何流过后端渲染引擎。例如，渲染图API能够为每个渲染图402创建数据文件。具体地，渲染图API为阴影渲染图402A创建数据文件，为相机渲染图402B创建另一数据文件，并为后处理渲染图402C创建第三数据文件。每个数据文件表示能够重复使用和/或修改以渲染其他帧的渲染图资产。

[0033] 渲染图资产能够为渲染图定义节点集合、渲染目标以及节点和渲染目标之间的连接。在数据驱动的操作中，渲染图API编译渲染图资产以生成渲染图数据对象，该对象被馈

送到后端渲染引擎。基于渲染图数据对象，后端渲染引擎构建管理节点集合的渲染图。每个节点都包含设置函数和执行函数，其能够使用lambda函数实现为回调操作。设置函数声明渲染目标使用和渲染图节点与渲染目标之间的依赖关系，并且执行函数将目标句柄解析为图形资源(例如，纹理)并实现某些图形命令(例如，使用给定的一组材料绘制调用收集的网孔)。在一个或多个具体实施中，后端渲染引擎能够管理渲染目标创建和/或用于渲染目标的存储器别名。

[0034] 图4示出了渲染帧400的特定具体实施，本公开不限于图4中所示的特定具体实施。本领域普通技术人员知道可以创建各种其他渲染通道和/或渲染层以生成渲染帧400。图1的使用和讨论仅仅是便于描述和解释的示例。

[0035] 图5表示渲染图系统500，其使用渲染图API来数据驱动和/或代码驱动渲染图。图5示出了渲染图系统500在逻辑上被划分为多个编程级别，其向开发者提供对后端渲染引擎501的不同访问级别。在顶级编程级别530，开发者利用可视图形编辑器来创建用于渲染帧和/或渲染图的人类可读表示(例如，可视图形表示)。在下一个较低编程级别532，渲染图API允许开发者通过暴露节点的输入和/或输出以及渲染图来在代码中写入渲染图和/或渲染帧。在编程级别534，开发者能够编写后端渲染引擎501如何从头到尾将像素写入屏幕的代码。后端渲染引擎501管理渲染目标创建、别名和/或存储器管理操作。渲染图API还允许开发者访问低级图形API(例如，Direct3D®)以生成用于渲染帧的图形命令。

[0036] 顶级编程级别530允许开发者为帧生成各种渲染图资产。在图5中，可视图形编辑器允许开发者为相机部件502和光部件504生成渲染图。开发者能够通过利用渲染图资产管线506为相机部件502和/或光部件504创建渲染图。渲染图资产管线506能够通过让开发者创建渲染图的人类可读表示和/或获得先前创建和保存的渲染图来创建渲染图。例如，开发者可以利用可视图形编辑器来修改已创建的渲染图资产。然后，渲染图资产管线506为每个渲染图生成渲染图文件508。在一个或多个实施方案中，渲染图文件508是JSON文件，其定义节点的集合、渲染目标以及节点和渲染目标之间的连接。渲染图API编译渲染图文件508以生成渲染图数据对象510。通过生成渲染图数据对象510，可视图形编辑器能够提供用于将不同渲染图资源附接到渲染图提供器和/或其他程序对象的钩子。

[0037] 对于数据驱动渲染图，渲染图节点程序对象512能够通过内省将节点公开给可视图形编辑器。渲染图文件508能够定义渲染图中的节点的输入、输出和设置。可视图形编辑器还向开发者公开钩子以将渲染图程序对象524应用于具有渲染图提供器接口的渲染图提供器程序对象516。如图5所示，通过利用渲染图提供器接口，渲染图提供器程序对象516可以提供用于返回渲染图数据对象510的类型串和函数。诸如相机部件502的部件可以利用渲染图提供器接口来实现渲染图提供器程序对象516并且具有渲染图资产句柄(例如，标识符)以渲染图形文件508。通过这样做，在运行时，当渲染相机时，相机部件502能够提供其自己的渲染图而不是默认的渲染图。

[0038] 在编程级别532，渲染图API允许开发者访问后端渲染引擎501的部分以在代码中写入渲染图。为了在代码中写入渲染图，开发者能够使用渲染图节点程序对象512、渲染帧程序对象514和渲染图管理器程序对象518。渲染图节点程序对象512用作为渲染图中的节点公开一个或多个参数的接口。例如，渲染图节点程序对象512将节点的输入和/或输出公开为成员，并且也是用于实现任何类型的渲染图节点的基类。例如，渲染图节点程序对象

512对应于网孔节点，该网孔节点对具有特定材料的收集网孔发出绘制调用。

[0039] 渲染帧程序对象514处理渲染图之间的输入和输出。使用阴影操作示例，开发者能够具有有许多阴影投射灯的渲染图，以及不同相机的渲染图以查看不同的阴影。每个渲染图可以具有相同的节点集合以产生场景的图像输出。渲染帧程序对象415允许开发者将渲染场景分解成渲染图的集合并互连渲染图。在图5中，开发者能够使用渲染帧程序对象514来创建渲染图构建器程序对象522，并跨渲染图的设置和编译阶段管理渲染图构建器程序对象522。

[0040] 渲染帧程序对象514还能够使用渲染图提供器程序对象516。如图5所示，渲染图管理器程序对象518可以管理何时提供渲染图以处理帧渲染的部分。为此，渲染图提供器程序对象516向渲染图管理器程序对象518注册。渲染图管理器程序对象518能够映射键以识别渲染图提供器程序对象516。渲染图提供器程序对象516能够选择为渲染帧的任意块提供渲染图。例如，相机能够实现图形提供器程序对象516并产生渲染帧的渲染图。回想一下，具有渲染图提供器程序对象516允许可视图形编辑器提供钩子以可选地数据驱动渲染图程序对象524的产生。

[0041] 在顶部编程级别530，渲染图API允许开发者访问后端渲染引擎501的部分以编写后端渲染引擎501如何从头到尾将像素写入屏幕的代码。渲染图节点接口程序对象520表示用于实现编程对象的渲染图系统500的基本单元。渲染图节点接口程序对象520中的每个包括设置和执行函数。设置函数允许开发者声明渲染图目标526和依赖关系。例如，设置函数定义渲染图目标526以创建、读取和写入。对于读取操作，开发者使用目标句柄来指定要从中访问的渲染图目标526。对于写操作，开发者使用目标句柄来指定渲染图目标526。执行函数将目标句柄解析为渲染图目标526并执行图形命令。可将图形命令添加到最终提交给图形处理器的命令缓冲器。如

[0042] 图5所示，渲染图目标526对应于图形API纹理资源528。

[0043] 渲染帧程序对象514为想要在顶级编程级别530工作的开发者提供代码驱动的接口。具体地，渲染帧程序对象514允许开发者构建和管理节点集合。渲染帧程序对象514提供被称为回调渲染图节点程序对象的内置节点实现类型。回调渲染图节点程序对象将设置和执行函数作为lambdas，使得开发者能够创建节点，而无需使用自己的渲染图节点接口实现类型来包装节点。回调渲染图节点程序对象为开发者提供了从头到尾将像素写入屏幕的功能。后端渲染引擎501管理创建渲染图目标526和别名，但是开发者管理渲染的内容。例如，如果开发者需要阴影标测图，开发者会将节点添加到渲染图中，该渲染图生成一个节点并使用一个节点。如图5所示，为了使用渲染图程序对象524，开发者可以利用渲染图构建器程序对象522，其提供用于声明渲染图目标526的接口。

[0044] 图6描绘了示出图形处理操作600的流程图，该图形处理操作600数据驱动渲染帧的渲染图。操作600利用渲染图API来生成数据驱动和代码驱动的渲染图。渲染图API能够支持用于与后端渲染引擎交接的各种编程级别。图5的使用和讨论仅仅是有利于解释的示例，并不旨在将本公开限制于这个具体示例。例如，虽然图6示出了操作600内的块以连续顺序实现，操作600不限于该顺序。

[0045] 操作600可以在框602处开始并且为渲染图定义一个或多个节点和一个或多个渲染目标。操作600能够使用数据驱动方法和/或代码驱动方法来定义渲染图。渲染图表示能

够作为渲染通道或渲染层执行的渲染管线。然后，操作600可以移动到框604并将渲染图存储为数据文件(例如，JSON数据文件)。作为数据文件，渲染图成为允许开发者重用或修改渲染图的资产。然后，操作600能够进行到框606并将数据文件覆盖到渲染图数据对象中。在框608处，操作600执行帧设置阶段，基于渲染图数据对象设置渲染图。在设置阶段期间，操作600可以声明目标使用以及节点和渲染目标之间的依赖关系。

[0046] 在完成设置阶段之后，操作600移动到框610并优化渲染图以进行处理。在一个或多个实施方案中，操作600分析目标和/或缓冲器句柄依赖图以确定渲染图是否包括未使用的渲染目标和/或节点。操作600可以基于其输出最终是否连接到帧缓冲器或来自渲染图的其他输出来对未使用的渲染目标和/或节点进行分类。如果渲染目标和/或节点不影响渲染图的输出，则操作600将渲染目标和/或节点分类为未使用。附加地或另选地，操作600还可以分析在设置期间声明的渲染目标使用，以管理用于在命令缓冲器中编码图形命令的命令编码器的寿命。具体地讲，操作600评估读取和/或写入的节点是否来自同一渲染目标。如果是，然后操作600将节点组合在一起以改善命令编码操作。在优化渲染图之后，操作600移动到框612并执行帧执行阶段，该帧执行阶段执行优化的渲染图。

[0047] 图7是包括渲染图API的计算系统700的简化框图，该渲染图API可以对应于或可以是计算机和/或任何其他计算设备的一部分，诸如工作站、服务器、大型机、超级计算机和/或便携式计算设备。在一个或多个实施方案中，计算系统700表示使人能够感测和/或与各种CGR环境交互的不同类型的电子系统。示例包括头戴式系统、基于投影的系统、平视显示器(HUD)、集成有显示能力的车辆挡风玻璃、集成有显示能力的窗户、被形成为被设计用于放置在人眼睛上的透镜的显示器(例如，类似于隐形眼镜)、耳机/听筒、扬声器阵列、输入系统(例如，具有或没有触觉反馈的可穿戴或手持控制器)、智能电话、平板电脑、和台式/膝上型计算机。头戴式系统可以具有一个或多个扬声器和集成的不透明显示器。另选地，头戴式系统可以被配置成接受外部不透明显示器(例如，智能电话)。头戴式系统可以结合用于捕获物理环境的图像或视频的一个或多个成像传感器、和/或用于捕获物理环境的音频的一个或多个麦克风。头戴式系统可以具有透明或半透明显示器，而不是不透明显示器。透明或半透明显示器可以具有媒介，代表图像的光通过该媒介被引导到人的眼睛。显示器可以利用数字光投影、有机发光二极管(OLED)、LED、uLED、硅基液晶、激光扫描光源或这些技术的任意组合。媒介可以是光学波导、全息图媒介、光学组合器、光学反射器、或它们的任意组合。在一个实施方案中，透明或半透明显示器可被配置成选择性地变得不透明。基于投影的系统可以采用将图形图像投影到人的视网膜上的视网膜投影技术。投影系统也可以被配置成将虚拟对象投影到物理环境中，例如作为全息图或在物理表面上。

[0048] 图7示出了计算系统700包括处理器702，其也可以被称为CPU。处理器702可以(例如，经由系统总线770)将指令传达和/或提供给计算系统700内的其他部件，诸如输入接口704、输出接口706和/或图形处理器712。在一个实施方案中，处理器702可以包括一个或多个多核处理器和/或用作数据的缓冲器和/或存储器的存储介质(例如，高速缓冲存储器)。另外，处理器702可以是一个或多个其他处理部件的一部分，诸如专用集成电路(ASIC)、现场可编程门阵列(FPGA)和/或数字信号处理器(DSP)。虽然图7示出了处理器702可以是单个处理器，处理器702不限于此，而是可以表示多个处理器。处理器702可以被配置为实现本文描述的任何操作，其包括如图6中所描述的操作600。

[0049] 图7示出了存储器708可以操作地耦接到处理器702。存储器708可以是被配置为存储各种类型的数据的非暂时性介质。例如，存储器708可以包括一个或多个存储器设备，其包括辅助存储器、只读存储器(ROM)和/或随机存取存储器(RAM)。辅助存储器通常包括一个或多个磁盘驱动器、光盘驱动器、固态驱动器(SSD)和/或磁带驱动器，并用于数据的非易失性存储。在某些情况下，如果分配的RAM不足以容纳所有工作数据，则辅助存储器可用于存储溢出数据。辅助存储器还可以用于存储当选择执行此类程序时加载到RAM中的程序。ROM用于存储指令以及可能在程序执行期间读取的数据。ROM是非易失性存储器设备，其相对于辅助存储器的较大存储容量通常具有较小的存储容量。RAM用于存储易失性数据，也许用于存储指令。

[0050] 如图7所示，存储器708可用于容纳用于执行本文描述的各种实施方案的指令。在实施方案中，存储器708可以包括可以由处理器702访问和实现的渲染引擎710。附加地或另选地，可以在嵌入在处理器702(例如，高速缓冲存储器)中的存储器内存储和访问渲染引擎710。渲染引擎710可以被配置为提供用于生成数据驱动渲染图的计算机可执行指令。在一个实施方案中，渲染引擎710可使用如图5所示的渲染图系统500和/或如图6中所述的操作600来实现。在一个实施方案中，存储器708可以与系统总线770(例如，计算机总线)交接，以便在执行软件程序(诸如包括程序代码的软件应用程序，和/或包含本文描述的功能的计算机可执行处理步骤)期间，将存储器708中存储的信息传达和/或传输到处理器702和/或图形处理器712。

[0051] 本领域普通技术人员知道，软件程序可以用各种计算语言针对各种软件平台和/或操作系统开发、编码和编译，并随后由处理器702加载和执行。在一个实施方案中，软件程序的编译过程可以将用编程语言编写的程序代码转换成另一种计算机语言，使得处理器702能够执行编程代码。例如，软件程序的编译过程可以生成可执行程序，该可执行程序为处理器702提供编码指令(例如，机器代码指令)以完成特定的、非通用的特定计算功能，诸如数据驱动渲染图。

[0052] 在编译过程之后，渲染引擎710可以作为计算机可执行指令或处理步骤从存储器(例如，存储器708、存储介质/媒介、可移除媒介驱动器和/或其他存储设备)加载到处理器702和/或嵌入在处理器702内。处理器702可执行存储的指令或处理步骤，以便执行指令或处理步骤(例如，渲染引擎710)以将计算系统700变换为非通用的、特定的、专门编程的机器或装置。存储数据(例如，存储设备存储的数据)能够在执行计算机可执行指令或处理步骤期间由处理器702访问，以指示计算系统700内的一个或多个部件。

[0053] 图7还示出了处理器702可以可操作地耦接到被配置为接收图像数据的输入接口704，以及被配置为输出和/或显示帧的输出接口706以及用于渲染帧的图形处理器712。输入接口704可以被配置为经由电缆、连接器、无线连接和/或其他通信协议获得图像数据和/或其他基于传感器的信息。在一个实施方案中，输入接口704可以是网络接口，其包括被配置为经由网络接收和/或发送数据的多个端口。具体地，网络接口可以经由有线链路、无线链路和/或逻辑链路来发送图像数据。输入接口704的其他示例可以是通用串行总线(USB)接口、CD-ROM、DVD-ROM和/或到一个或多个传感器的连接。输出接口706可包括到用于图形显示器(例如，监视器)、产生所生成结果的硬拷贝的打印设备和/或通过电缆、连接器、无线连接和/或其他通信协议传输数据的多个端口的一个或多个连接。

[0054] 在本公开中提到“一个实施方案”或“实施方案”意指包括在本公开的至少一个实施方案中的结合该实施方案所述的特定特征、结构或特性，并且多次提到“一个实施方案”或“实施方案”不应被理解为必然地全部参考相同的实施方案。除非明确限定，否则术语“一个”、“一种”和“该”并非旨在指代单数实体，而是包括其特定示例可以被用于举例说明的一般性类别。因此，术语“一个”或“一种”的使用可以意指至少一个的任意数目，包括“一个”、“一个或多个”、“至少一个”和“一个或不止一个”。术语“或”意指可选项中的任意者以及可选项的任何组合，包括所有可选项，除非可选项被明确指示是相互排斥的。短语“中的至少一者”在与项目列表组合时是指列表中的单个项目或列表中项目的任何组合。所述短语并不要求所列项目的全部，除非明确如此限定。

[0055] 出于本公开的目的，术语“物理环境”指的是人们能够在没有电子系统的帮助的情况下感知和/或交互的物理世界。物理环境诸如物理公园包括物理物品，诸如物理树木、物理建筑物和物理人。人们能够诸如通过视觉、触觉、听觉、味觉和嗅觉来直接感测物理环境和/或与物理环境交互。

[0056] 相反，术语“计算机生成现实(CGR)环境”是指人们经由电子系统感知和/或交互的完全或部分模拟的环境。在CGR中，跟踪人的物理运动的一个子集或其表示，并且作为响应，以符合至少一个物理定律的方式调节在CGR环境中模拟的一个或多个虚拟对象的一个或多个特征。例如，CGR系统可以检测人的头部转动，并且作为响应，以与此类视图和声音在物理环境中变化的方式类似的方式调节呈现给人的图形内容和声场。在一些情况下(例如，出于可达性原因)，对CGR环境中虚拟对象的特征的调节可以响应于物理运动的表示(例如，声音命令)来进行。

[0057] 人可以利用其感官中的任一者来感测CGR对象和/或与CGR对象交互，包括视觉、听觉、触觉、味觉和嗅觉。例如，人可以感测音频对象和/或与音频对象交互，该音频对象创建3D或空间音频环境，该3D或空间音频环境提供3D空间中点音频源的感知。又如，音频对象可以使能音频透明度，该音频透明度在有或者没有计算机生成的音频的情况下选择性地引入来自物理环境的环境声音。在某些CGR环境中，人可以感测和/或只与音频对象交互。CGR的示例包括虚拟现实和混合现实。

[0058] 如本文所使用的，术语“虚拟现实(VR)环境”指的是被设计为完全基于计算机生成的一个或多个感官的感觉输入的模拟环境。VR环境包括人可以感测和/或交互的多个虚拟对象。例如，树木、建筑物和代表人的化身的计算机生成的图像是虚拟对象的示例。人可以通过在计算机生成的环境内人的存在的模拟、和/或通过在计算机生成的环境内人的物理运动的一个子组的模拟来感测和/或与VR环境中的虚拟对象交互。

[0059] 与被设计成完全基于计算机生成的感官输入的VR环境相比，术语“混合现实(MR)环境”是指被设计成除了包括计算机生成的感官输入(例如，虚拟对象)之外还引入来自物理环境的感官输入或其表示的模拟环境。在虚拟连续体上，混合现实环境是完全物理环境作为一端和虚拟现实环境作为另一端之间的任何状况，但不包括这两端。

[0060] 在一些MR环境中，计算机生成的感官输入可以对来自物理环境的感官输入的变化进行响应。另外，用于呈现MR环境的一些电子系统可以跟踪相对于物理环境的位置和/或取向，以使虚拟对象能够与真实对象(即，来自物理环境的物理物品或其表示)交互。例如，系统可以导致运动使得虚拟树木相对于物理地面看起来是静止的。混合现实的示例包括增强

现实和增强虚拟。

[0061] 在本公开内容中，术语“增强现实(AR)环境”指的是模拟环境，其中一个或多个虚拟对象叠加在物理环境或其表示上。例如，用于呈现AR环境的电子系统可具有透明或半透明显示器，人可以透过该显示器直接查看物理环境。该系统可以被配置成在透明或半透明显示器上呈现虚拟对象，使得人利用该系统感知叠加在物理环境之上的虚拟对象。另选地，系统可以具有不透明显示器和一个或多个成像传感器，成像传感器捕获物理环境的图像或视频，这些图像或视频是物理环境的表示。系统将图像或视频与虚拟对象组合，并在不透明显示器上呈现组合物。人利用系统经由物理环境的图像或视频而间接地查看物理环境，并且感知叠加在物理环境之上的虚拟对象。如本文所用，在不透明显示器上显示的物理环境的视频被称为“透传视频”，意味着系统使用一个或多个图像传感器捕获物理环境的图像，并且在不透明显示器上呈现AR环境时使用那些图像。进一步另选地，系统可以具有投影系统，该投影系统将虚拟对象投射到物理环境中，例如作为全息图或者在物理表面上，使得人利用该系统感知叠加在物理环境之上的虚拟对象。

[0062] 增强现实环境也是指其中物理环境的表示被计算机生成的感官信息进行转换的模拟环境。例如，在提供透传视频中，系统可以对一个或多个传感器图像进行转换以施加与成像传感器所捕获的视角不同的选择视角(例如，视点)。又如，物理环境的表示可以通过图形地修改(例如，放大)其部分而进行转换，使得修改后的部分可以是原始捕获图像的代表性的但不是真实的版本。再如，物理环境的表示可以通过以图形方式消除或模糊其部分而进行转换。

[0063] 出于本公开的目的，“增强虚拟(AV)环境”是指模拟环境，其中虚拟或计算机生成的环境包含来自物理环境的一个或多个传感输入。感官输入可以是物理环境的一个或多个特征的表示。例如，AV公园可以具有虚拟树木和虚拟建筑物，但人的脸部是从对物理人拍摄的图像逼真再现的。又如，虚拟对象可以采用一个或多个成像传感器所成像的物理物品的形状或颜色。再如，虚拟对象可以采用符合太阳在物理环境中的位置的阴影。

[0064] 公开了至少一个实施方案，并且本领域的普通技术人员所做的所述具体实施和/或所述具体实施的特征的变型、组合和/或修改在本公开的范围内。由组合、集成和/或省略具体实施的特征而得到的另选具体实施也在本公开的范围内。在明确说明数值范围或限制的情况下，此类表述范围或限制可以被理解为包括落入明确指出的范围或限制内的类似量值的迭代范围或限制(例如，从约1至约10包括2、3、4等；大于0.10包括0.11、0.12、0.13等)。除非另外指明，否则术语“约”的使用是指后面数的±10％。

[0065] 在仔细研究以上描述时，许多其他具体实施对于本领域的技术人员而言将是显而易见的。因此，应当参考所附权利要求以及赋予此类权利要求的等同形式的完整范围来确定本发明的范围。在所附权利要求书中，术语“包括(including)”和“其中(in which)”被用作相应术语“包括(comprising)”和“其中(wherein)”的通俗英语等同形式。

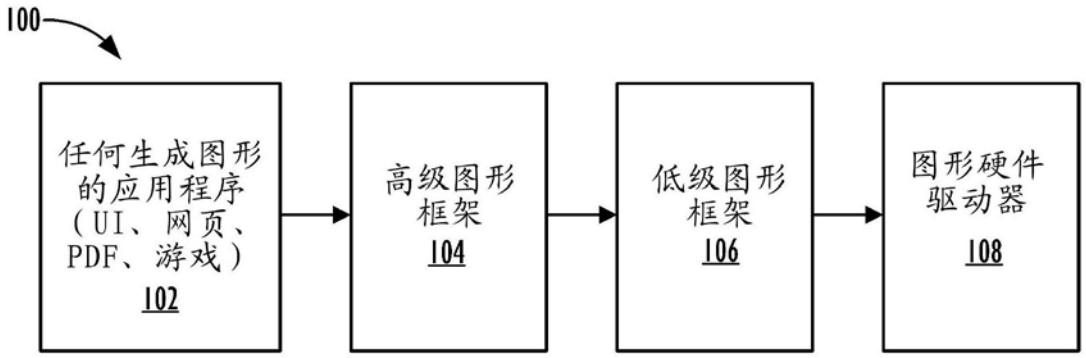

100
任何生成图形的应用程序（UI、网页、PDF、游戏）
102
高级图形框架
104
低级图形框架
106
图形硬件驱动器
108

图1

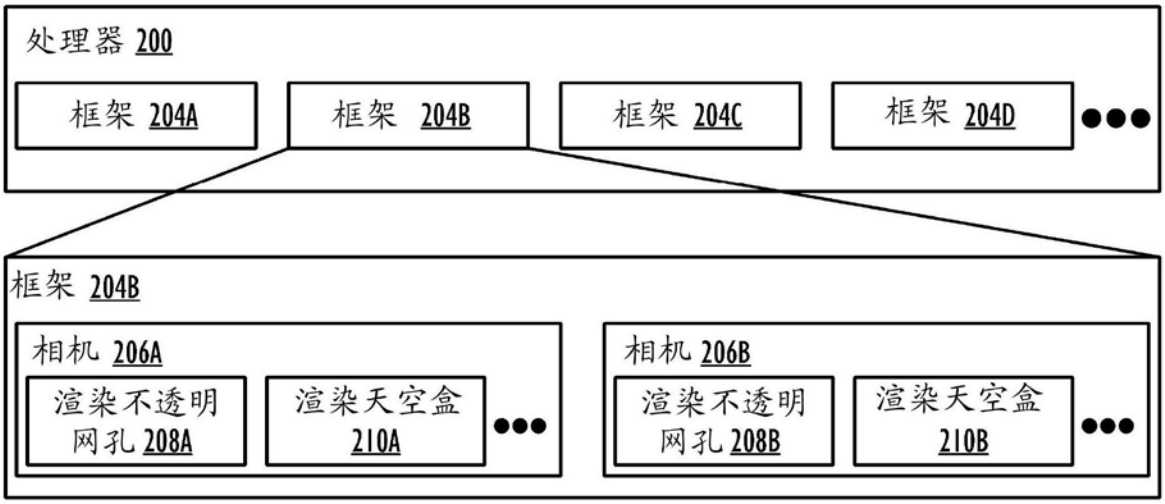

处理器 200
框架 204A
框架 204B
框架 204C
框架 204D
框架 204B
相机 206A
渲染不透明网孔 208A
渲染天空盒 210A
相机 206B
渲染不透明网孔 208B
渲染天空盒 210B

图2

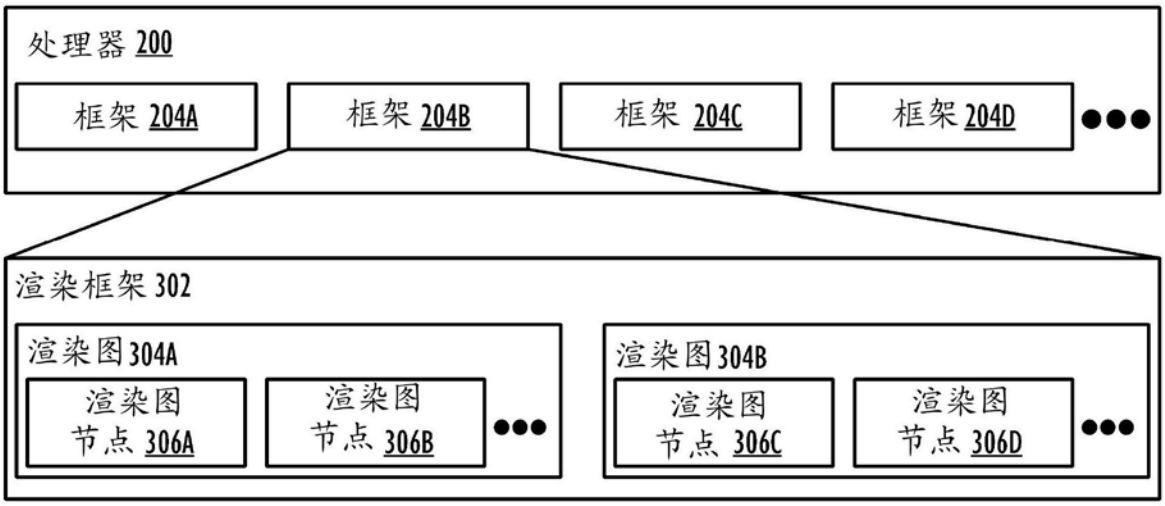

处理器 200
框架 204A
框架 204B
框架 204C
框架 204D
渲染框架 302
渲染图 304A
渲染图节点 306A
渲染图节点 306B
渲染图 304B
渲染图节点 306C
渲染图节点 306D

图3

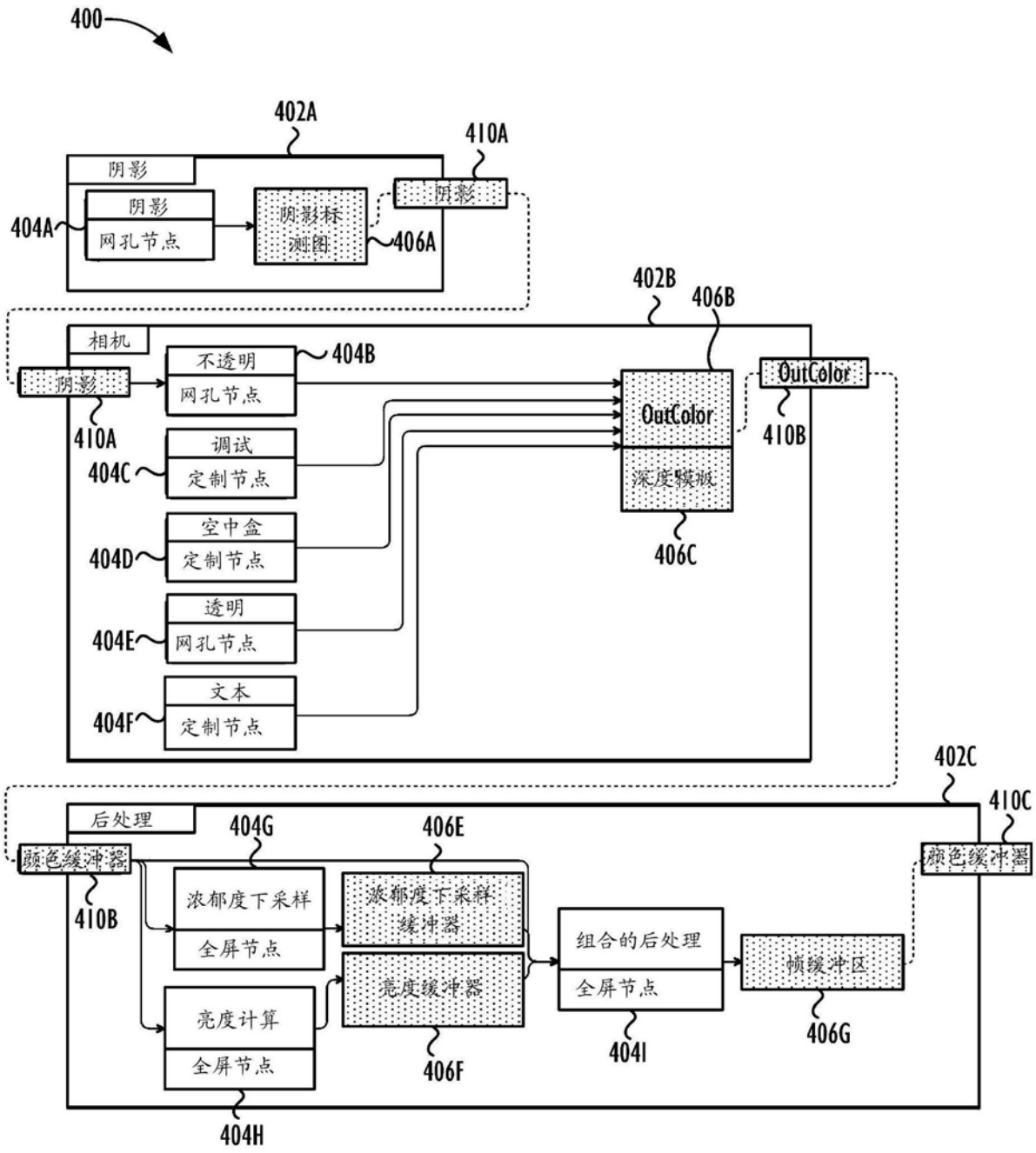

400
402A
410A
阴影
阴影
网孔节点
404A
阴影标
测图
406A
阴影
402B
406B
相机
不透明
网孔节点
404B
阴影
410A
OutColor
OutColor
410B
调试
定制节点
404C
深度模板
406C
空中盒
定制节点
404D
透明
网孔节点
404E
文本
定制节点
404F
402C
410C
后处理
404G
406E
颜色缓冲器
颜色缓冲器
410B
浓郁度下采样
全屏节点
浓郁度下采样
缓冲器
组合的后处理
全屏节点
帧缓冲区
406G
亮度计算
全屏节点
亮度缓冲器
406F
404I
404H

图4

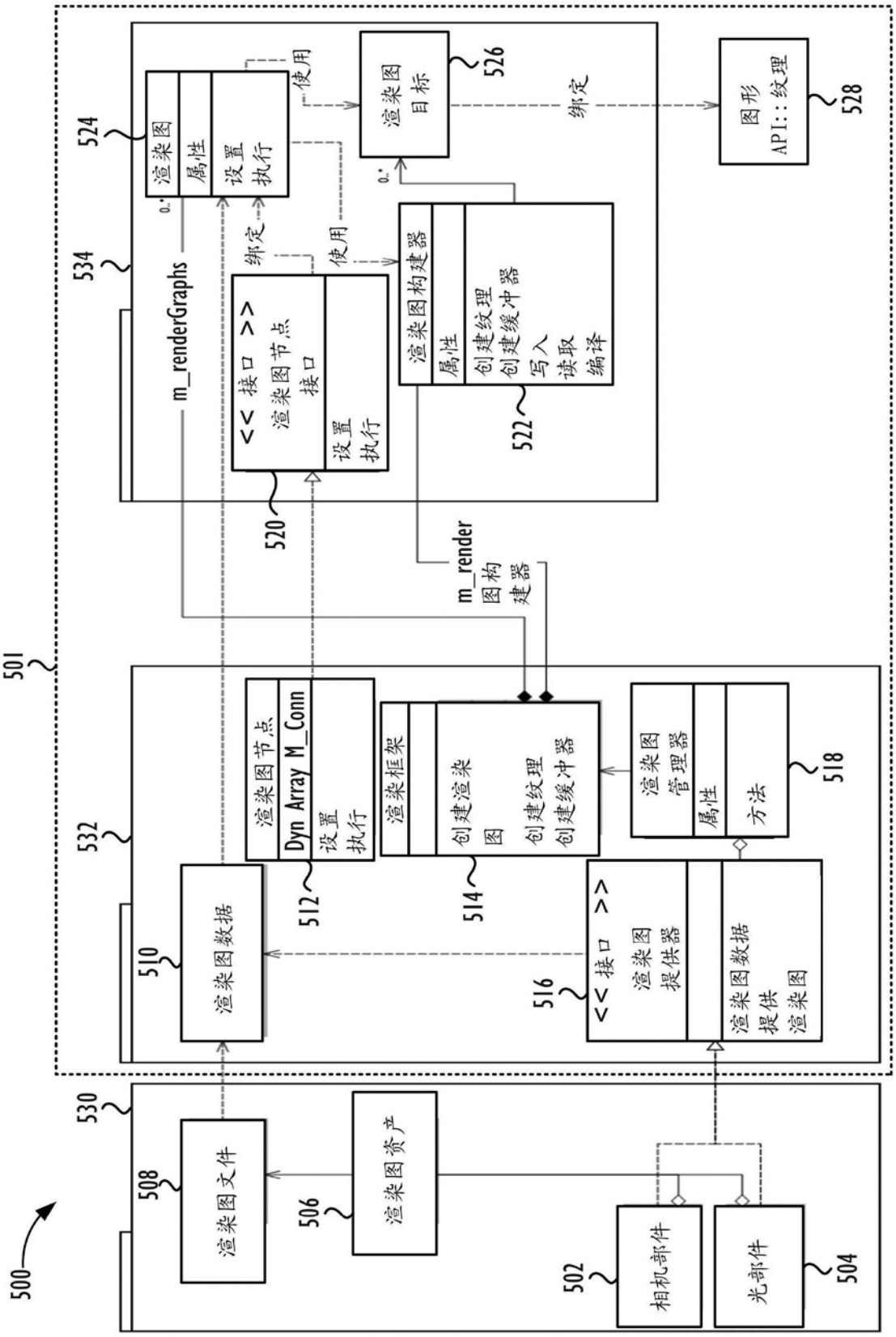

500
501
502
相机部件
504
光部件
506
渲染图资产
508
渲染图文件
510
渲染图数据
512
渲染图节点
Dyn Array M_Conn
设置
执行
514
渲染框架
创建渲染
图
创建纹理
创建缓冲器
516
<< 接口 >>
渲染图
提供器
渲染图数据
提供
渲染图
518
渲染图
管理器
属性
方法
520
<< 接口 >>
渲染图节点
接口
设置
执行
522
渲染图构建器
属性
创建纹理
创建缓冲器
写入
读取
编译
524
渲染图
属性
设置
执行
526
渲染图
目标
528
图形
API::纹理
530
532
534
m_renderGraphs
m_render
图构
建器
使用
绑定
使用
绑定
0..*
0..*

图5

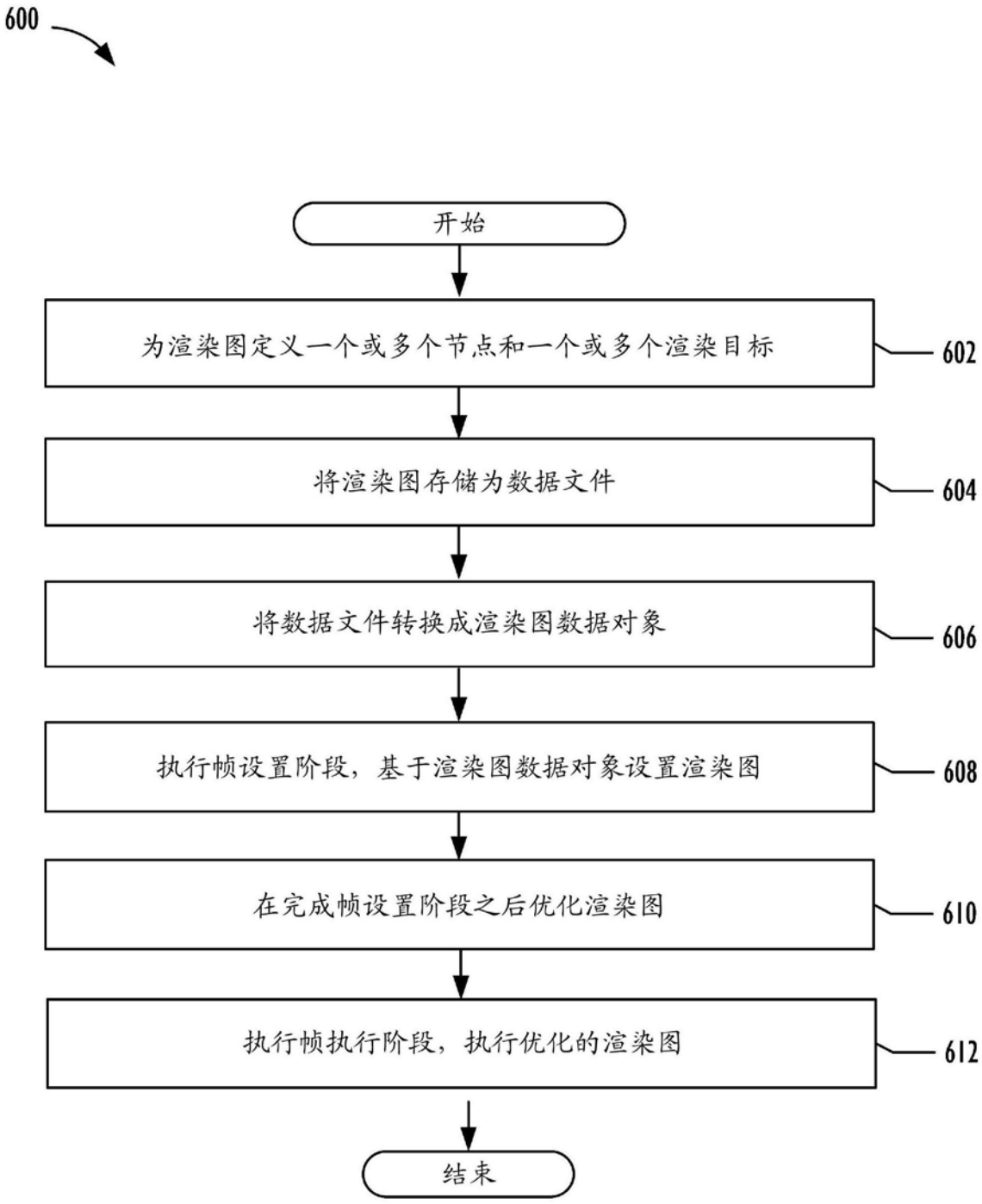

600
开始
为渲染图定义一个或多个节点和一个或多个渲染目标
602
将渲染图存储为数据文件
604
将数据文件转换成渲染图数据对象
606
执行帧设置阶段，基于渲染图数据对象设置渲染图
608
在完成帧设置阶段之后优化渲染图
610
执行帧执行阶段，执行优化的渲染图
612
结束

图6

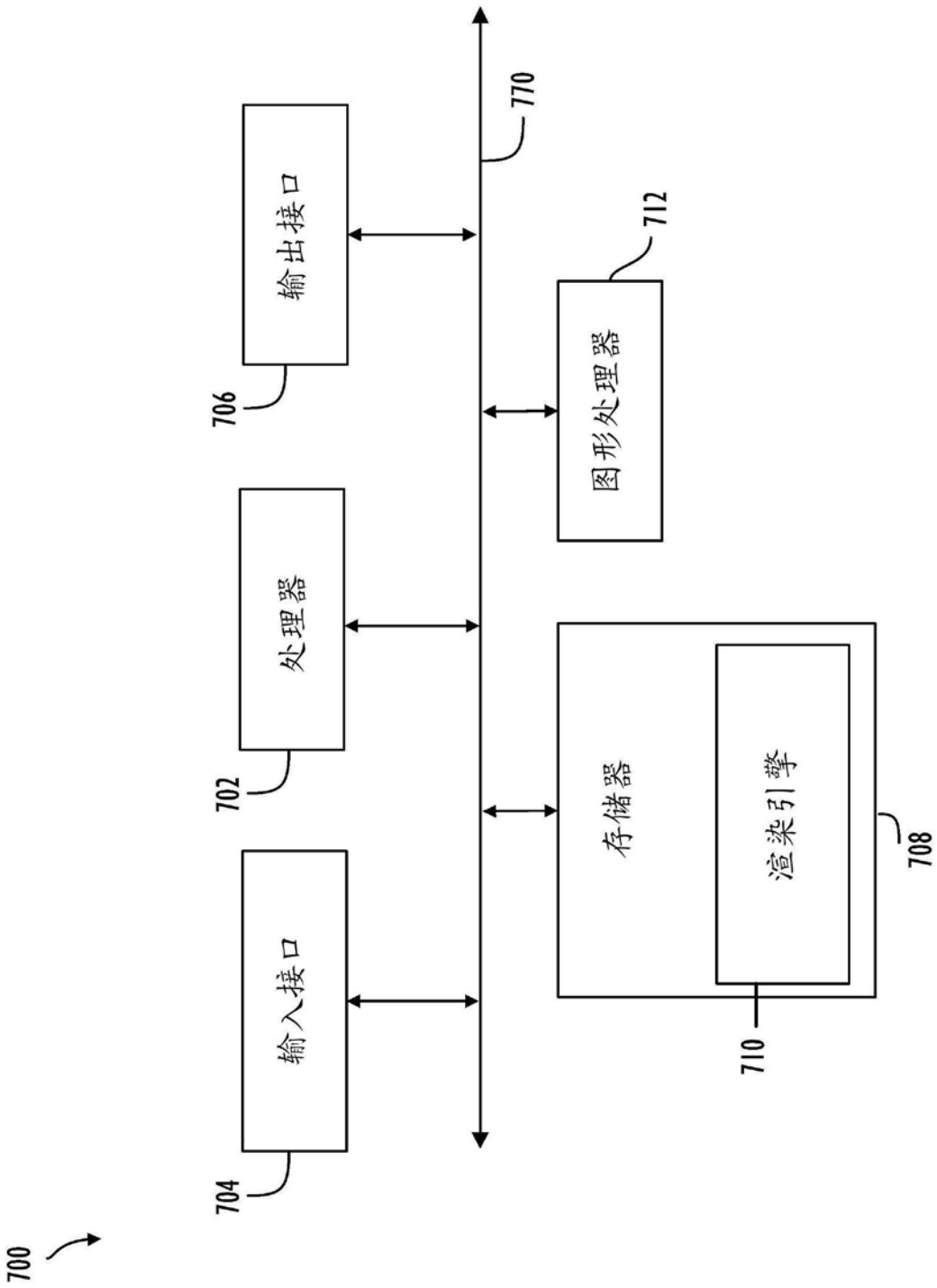


图7