(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2022 年 12 月 29 日 (29.12.2022)

WIPO | PCT

(10) 国际公布号
WO 2022/266821 A1

(51) 国际专利分类号:
G06T 15/00 (2011.01)

(21) 国际申请号: PCT/CN2021/101386

(22) 国际申请日: 2021 年 6 月 22 日 (22.06.2021)

(25) 申请语言: 中文

(26) 公布语言: 中文

(71) 申请人: 华 为 技 术 有 限 公 司 (**HUAWEI TECHNOLOGIES CO., LTD.**) [CN/CN]; 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。

(72) 发明人: **杨喜乐 (YANG, Xile)**; 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。 **赖斯奚 (LAI, Siyan)**; 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。 **吴任初 (WU, Renchu)**; 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。

(74) 代理人: 北京龙双利达知识产权代理有限公司 (**LONGSUN LEAD IP LTD.**); 中国北京市海淀区北清路 81 号院二区 3 号楼 8 层 801-1室, Beijing 100094 (CN)。

(81) 指定国 (除另有指明，要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW。

(84) 指定国 (除另有指明，要求每一种可提供的地区保护): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG,

(54) **Title:** GRAPHICS RENDERING METHOD AND APPARATUS

(54) **发明名称:** 图形渲染的方法及装置

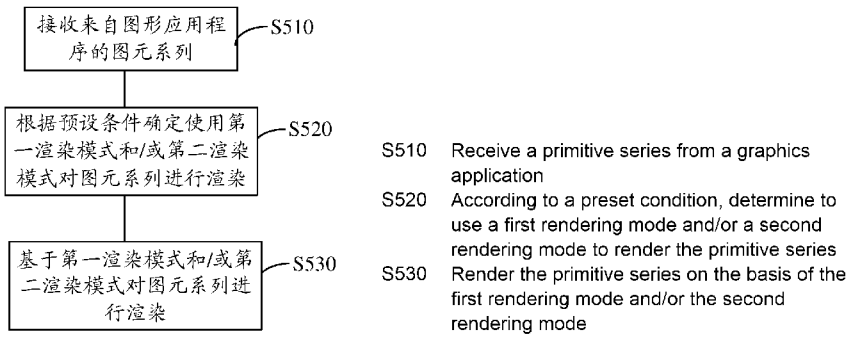

图 5

(57) **Abstract:** The present application provides a graphics rendering method and apparatus, the method being applied to a GPU tiled rendering system comprising a first rendering mode and a second rendering mode; the first rendering mode indicates that primitive vertex data is generated and stored in a geometric processing phase, and the primitive vertex data is read from memory in a raster processing phase for rendering to generate an output image; the second rendering mode indicates that primitive vertex data generated in the geometric processing phase is not stored in the GPU memory, and vertex shading processing is performed on primitives during the raster processing phase to generate primitive vertex data. Specifically, the method comprises: receiving a primitive series from a graphics application program; according to a preset condition, determining to use the first rendering mode and/or the second rendering mode to render the primitive series; rendering the primitive series on the basis of the first rendering mode and/or the second rendering mode. Selecting a rendering mode according to the preset condition prevents rendering on the basis of a fixed mode from affecting the rendering performance of a GPU.

WO 2022/266821 A1

[见续页]

CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:
— 包括国际检索报告(条约第21条(3))。

(57) 摘要: 本申请提供了一种图形渲染的方法和装置，该方法应用于包括第一渲染模式和第二渲染模式GPU分块渲染系统中，第一渲染模式指示在几何处理阶段生成并存储图元顶点数据，光栅处理阶段从内存中读取图元顶点数据进行渲染产生输出图像，第二渲染模式指示在几何处理阶段生成的图元顶点数据未存储在GPU内存中，光栅处理阶段对图元进行顶点着色处理生成图元顶点数据具体地，该方法包括：接收来自图形应用程序的图元系列；根据预设条件确定使用第一渲染模式和/或第二渲染模式对图元系列进行渲染；基于第一渲染模式和/或第二渲染模式对图元系列进行渲染。通过根据预设条件选择渲染模式进行渲染，避免基于固定模式进行渲染影响GPU的渲染性能。

图形渲染的方法及装置

技术领域

本申请涉及信息技术领域，并且更具体地，涉及一种图形渲染的方法及装置。

背景技术

计算机图形处理器（graphics processing unit，GPU）分块渲染系统将待渲染处理的画面分割成许多个片块，渲染过程包括：在几何处理阶段对图形应用程序输入的图元系列做包括顶点着色在内的几何处理之后，并根据图元对屏幕的覆盖关系将图元分类到渲染画面的各个片块中；然后在光栅处理阶段分别对各个片块进行渲染处理，多个片块的渲染可以在GPU内并行处理。在GPU分块渲染系统中，几何处理阶段之后生成的图元顶点数据不能直接用于光栅处理阶段的渲染处理。

目前提供的一种图形渲染的方法：将几何处理阶段生成的图元顶点数据存储到GPU内存中，光栅处理阶段从GPU内存中读取并使用图元顶点数据；还提供另一种图形渲染的方法：几何处理阶段生成的图元顶点数据不进行存储，光栅处理阶段渲染处理过程中重新对图元做几何处理，由顶点着色器生成图元顶点数据。

上述的图形渲染的方法基于固定模式进行图形渲染，可能会增加GPU的内存需求或增加GPU的运算量，影响GPU的渲染性能。因此如何在图形渲染的过程中保证GPU的渲染性能成为亟待解决的问题。

发明内容

本申请提供一种图形渲染的方法，能够提高图形渲染的性能。

第一方面，提供了一种图形渲染的方法，应用于计算机图形处理器GPU分块渲染系统中，该GPU分块渲染系统支持的渲染模式包括第一渲染模式和第二渲染模式，

该方法包括：

接收来自图形应用程序的图元系列；根据预设条件确定使用该第一渲染模式和/或该第二渲染模式对该图元系列进行渲染；基于该第一渲染模式和/或该第二渲染模式对该图元系列进行渲染；其中，该第一渲染模式指示将在几何处理阶段进行顶点着色处理生成的图元顶点数据存储在该GPU内存中，光栅处理阶段从该GPU内存中读取该图元顶点数据进行渲染处理，该第二渲染模式指示未将在几何处理阶段进行顶点着色处理生成的图元顶点数据存储在该GPU内存中，该光栅处理阶段进行顶点着色处理生成该图元顶点数据并对该图元顶点数据进行渲染处理。

本申请实施例提供的图形渲染的方法，GPU分块渲染系统能够根据预设条件选择使用哪种模式进行图形渲染，能够避免基于固定模式进行图形渲染而影响GPU的渲染性能。

结合第一方面，在第一方面的某些实现方式中，该预设条件包括以下至少一种：比例常数大于第一阈值、顶点着色器程序复杂程度的加权系数大于第二阈值、图元加权系数大

于第三阈值、存储资源加权系数大于第四阈值或综合加权系数大于第五阈值，其中，该比例常数为该第一渲染模式执行次数和该第二渲染模式执行次数的比值，该顶点着色器程序复杂程度加权系数为顶点着色器程序的指令数、复杂指令数、内存读取指令数和循环执行指令数的加权值，该图元加权系数为图元覆盖和图元顶点数据字节数的加权值，该存储资源加权系数为该 GPU 内存的存储资源利用率，该综合加权系数为该比例常数、该顶点着色器程序复杂程度的加权系数、该图元加权系数和该存储资源加权系数的加权值。

进一步地，该预设条件还包括：渲染场景为预设的渲染场景。

上述预设条件的具体形式可以是多种，也就是说 GPU 分块渲染系统可以通过多种不同的方式确定使用哪种模式进行图形渲染，能够增加方案的灵活性。

结合第一方面，在第一方面的某些实现方式中，该方法还包括：根据该图元系列确定预设条件包括该顶点着色器程序复杂程度的加权系数大于第二阈值、该图元加权系数大于第三阈值和该存储资源加权系数大于第四阈值中的至少一项，其中，该图元系列所绑定的顶点着色器程序用于确定该顶点着色器程序复杂程度的加权系数，该图元系列中图元覆盖和图元顶点数据字节数用于确定该图元加权系数，该 GPU 内存的存储资源利用率用于确定该存储资源加权系数。

或者，

该方法还包括：

接收来自该图形应用程序的顶点着色器程序；根据该顶点着色器程序确定预设条件包括顶点着色器程序复杂程度的加权系数大于第二阈值。

本申请实施例提供的图形渲染的方法，GPU 分块渲染系统不仅能够根据输入的图元数据确定上述的预设条件，还可以基于输入的顶点着色器程序确定上述的预设条件，增加方案的灵活性。

结合第一方面，在第一方面的某些实现方式中，在该第一渲染模式下，该方法还包括：在该几何处理阶段基于第一顶点着色器生成图元顶点位置数据，该图元顶点位置数据用于剔除屏幕上不可见的图元；以及基于第二顶点着色器生成该屏幕上可见的图元对应的图元顶点属性数据。

结合第一方面，在第一方面的某些实现方式中，在该第二渲染阶段下，该方法还包括：在该光栅处理阶段基于第三顶点着色器生成图元顶点位置数据，该图元顶点位置数据用于剔除屏幕上不可见的图元；以及基于第四顶点着色器生成该屏幕上可见的图元对应的图元顶点属性数据。

本申请实施例提供的图形渲染的方法，可以将顶点着色器编译成两个独立的程序，一个专门用来产生图元顶点位置数据的顶点着色器程序和另外一个用来产生图元顶点属性数据的顶点着色器程序。其中，用来产生图元顶点位置数据的顶点着色器程序中的指令数量以及执行程序所需的输入输出数据量通常会有所减少。因此在需要图元顶点位置数据的阶段，顶点着色器程序会减少渲染过程的计算量，从而提高 GPU 系统的性能。

第二方面，提供一种图形渲染装置，该装置包括用于执行第一方面中所描述的方法/操作/步骤/动作所对应的模块。

上述装置可以是服务器，也可以是服务器中的用于执行图形渲染的装置（例如，芯片，或者是能够和服务器匹配使用的装置）。

上述图形渲染装置包括的模块可以是硬件电路，也可是软件，也可以是硬件电路结合软件实现。

第三方面，提供了一种图形渲染装置，该装置包括处理器，该处理器用于调用存储器存储的程序代码以执行上述第一方面中的任意一种方式中的部分或全部操作。

上述装置中，存储程序代码的存储器既可以位于图形渲染装置内部（图形渲染装置除了包括处理器之外，还可以包括存储器），也可以位于图形渲染装置外部（可以是其他设备的存储器）。

可选地，上述存储器为非易失性存储器。

当图形渲染装置包括处理器和存储器时，该处理器和存储器可以耦合在一起。

第四方面，提供了一种计算机可读存储介质，计算机可读存储介质存储了程序代码，其中，程序代码包括用于执行上述任一方面所描述的方法中的部分或全部操作的指令。

可选地，上述计算机可读存储介质位于电子设备内，该电子设备可以是能够进行图形渲染的装置。

第五方面，本申请实施例提供一种计算机程序产品，当计算机程序产品在通信装置上运行时，使得通信装置执行上述任一方面所描述的方法中的部分或全部操作。

第六方面，提供了一种芯片，所述芯片包括处理器，所述处理器用于执行上述任一方面所描述的方法中的部分或全部操作。

第七方面，提供了一种系统，包括前述的图形应用程序和图形渲染装置。

附图说明

图 1 是本申请实施例提供的图形渲染的方法适用的一种系统。

图 2 是本申请提供的一种电子设备的示意图。

图 3 是一种 GPU 分块渲染示意性流程图。

图 4 是另一种 GPU 分块渲染示意性流程图。

图 5 是本申请实施例提供的一种图形渲染的方法示意性流程图。

图 6 是本申请实施例提供一种确定渲染模式的示意性流程图。

图 7 是本申请实施例提供另一种确定渲染模式的示意性流程图。

图 8 是本申请实施例提供的一种 GPU 系统示意性框图。

图 9 是本申请实施例的图形渲染装置 900 的示意性框图。

图 10 示出了本申请一个实施例提供的服务器 1000 的示意性框图。

具体实施方式

下面将结合附图，对本申请中的技术方案进行描述。

本申请实施例的图形渲染的方法可以应用于图 1 所示系统架构中。图 1 是本申请实施例提供的图形渲染的方法适用的一种系统。

如图 1 所示，系统架构中可以包括图形应用程序 110 和 GPU120，GPU120 以计算机图形应用程序 110（例如，计算机游戏）中的三维模型作为输入数据，经过对三维模型中图元（例如，三角形、线或点）以及顶点数据（例如，顶点坐标、表面法向量、RGBA 颜色、辅助颜色、颜色索引、雾坐标、纹理坐标、多边形的边界标志）进行渲染处理，最后

产生相应的输出图像。

示例性地，GPU 的渲染过程主要分成几何处理及光栅处理两个阶段。

其中，几何处理阶段主要模块包括:

1）顶点着色器：顶点着色器是 GPU 渲染流程中的第一个阶段，也是一个可编程处理器。GPU 读取描述三维图形外观的顶点数据，通过执行图形应用程序所提供的顶点着色器程序来确定三维图形的形状及位置关系，建立起三维图形的骨架。

2）投影及屏幕映射：GPU 把经过顶点着色器产生的三维图形从三维空间映射到屏幕的二维平面上。

3）图元剔除：GPU 把三维图形从三维空间映射到屏幕的二维平面上以后，有一些图元可能因为某种原因不会出现在屏幕的二维平面上。将这些图元从 GPU 的渲染过程中剔除，不会影响渲染的输出图像，而且还会提高 GPU 系统的性能。

例如，三维物体背面的图元，或者被其他物体遮盖的图元都不会显示在屏幕的二维平面上；还例如，有一些小三角形可能不覆盖任何像素采样点，因此也不会显示在屏幕的二维平面上。

4）分块器：在 GPU 分块渲染系统中，分块器将几何处理后的图元按照其投影在二维屏幕上的坐标划分到屏幕上相应的片块中，以便在渲染过程做分块渲染。

光栅处理阶段主要模块包括:

1）光栅生成器：在 GPU 分块渲染系统中光栅处理阶段是在屏幕的片块中分别进行的。光栅生成器根据几何处理阶段产生的分块信息来读取渲染本片块中图元的图元顶点数据。经过几何处理阶段产生的屏幕二维平面上图元，由光栅生成器产生代表相应图元的像素。在 GPU 流程光栅处理阶段会对这些屏幕二维平面上的像素做进一步处理。

2）深度测试：深度测试阶段对光栅生成器产生的图元像素对比背景深度做深度测试，从而去除在屏幕上被遮挡在后面的图元像素。经过深度测试可以减少 GPU 对在屏幕上不可见像素的进一步处理。

3）像素着色器：像素着色器是 GPU 流程中除了顶点着色器以外另外一个可编程处理器。像素着色器通过执行图形应用程序所提供的像素着色器指令来确定代表三维图形的每一个像素的颜色。

4）像素操作：代表三维图形的每一个像素的颜色由像素着色器产生后，GPU 流程中可能对这些像素的颜色做进一步的操作，例如对像素的合并，混合，透明度测试等等。GPU 在像素操作完成后输出最终渲染图像。

本申请实施例中用于图形渲染方法可以由电子设备来执行。该电子设备可以是移动终端（例如，智能手机），电脑，个人数字助理，可穿戴设备，车载设备，物联网设备或者其他能够进行图形渲染处理的设备。该电子设备可以是运行安卓系统、IOS 系统、windows 系统以及其他系统的设备。

下面结合图 2 对电子设备的具体结构进行详细的介绍。图 2 是本申请提供的一种电子设备的示意图。图 2 所示的电子设备可以是包括图 1 所示的系统架构的电子设备 200。

在一个实施例中，如图 2 所示，电子设备 200 可以包括：中央处理器（CPU）201、图形处理器（GPU）202、显示设备 203 和存储器 204。可选地，该电子设备 200 还可以包括至少一个通信总线 210（图 2 中未示出），用于实现各个组件之间的连接通信。

应当理解，电子设备 200 中的各个组件还可以通过其他连接器相耦合，其他连接器可包括各类接口、传输线或总线等。电子设备 200 中的各个组件还可以是以处理器 201 为中心的放射性连接方式。在本申请的各个实施例中，耦合是指通过相互电连接或连通，包括直接相连或通过其他设备间接相连。

中央处理器 201 和图形处理器 202 的连接方式也有多种，不局限于图 2 所示的方式。电子设备 200 中的中央处理器 201 和图形处理器 202 可以位于同一个芯片上，也可以分别为独立的芯片。

下面对中央处理器 201、图形处理器 202、显示设备 203 的作用进行简单的介绍。

中央处理器 201：用于运行操作系统 205 和应用程序 207。应用程序 207 可以为图形类应用程序，比如游戏、视频播放器等等。操作系统 205 提供了系统图形库接口，应用程序 207 通过该系统图形库接口，应用程序 207 可以接收服务器发送的用于渲染图形或图像帧的指令流（例如，渲染指令）。操作系统 205 提供的驱动程序，比如图形库用户态驱动和/或图形库内核态驱动，生成图形处理器 202 中渲染管线能够识别的指令流以及所需的相关渲染数据。其中，系统图形库包括但不限于：嵌入式开放图形库（open graphics library for embedded system，OpenGL ES）、柯罗诺斯平台图形界面（the khronos platform graphics interface）或 Vulkan（一个跨平台的绘图应用程序接口）等系统图形库。指令流包含一些列的指令，这些指令通常为对系统图形库接口的调用指令。

可选地，中央处理器 201 可以包括以下至少一种类型的处理器：应用处理器、一个或多个微处理器、数字信号处理器（digital signal processor, DSP）、微控制器（microcontroller unit，MCU）或人工智能处理器等。

中央处理器 201 还可进一步包括必要的硬件加速器，如专用集成电路（application specific integrated circuit, ASIC）、现场可编程门阵列（field programmable gate array, FPGA）、或者用于实现逻辑运算的集成电路。处理器 202 可以被耦合到一个或多个数据总线，用于在电子设备 200 的各个组件之间传输数据和指令。

图形处理器 202：用于接收处理器 202 发送的图形指令流，通过渲染管线（pipeline）生成渲染目标，并通过操作系统的图层合成显示模块将渲染目标显示到显示设备 203。

可选地，图形处理器 202 可以包括执行软件的通用图形处理器，如 GPU 或其他类型的专用图形处理单元等。

显示设备 203：用于显示由电子设备 200 生成的各种图像，该图像可以为操作系统的图形用户界面（graphical user interface，GUI）或由图形处理器 2002 处理的图像数据（包括静止图像和视频数据）。

可选地，显示设备 203 可以包括任何合适类型的显示屏。例如液晶显示器（liquid crystal display，LCD）或等离子显示器或有机发光二极管（organic light-emitting diode，OLED）显示器等。

渲染管线是图形处理器 202 在渲染图形或图像帧的过程中顺序执行的一系列操作，典型的操作包括：顶点处理（vertex processing）、图元处理（primitive processing）、光栅化（rasterization）、片段处理（fragment processing）等等。

上文中介绍了本申请实施例适用的系统架构以及电子设备内部执行图形渲染的流程。

由上述的图 1 所示的 GPU 分块渲染系统架构可知几何处理阶段之后生成的图元顶点

数据不可能直接用于第二阶段的渲染处理。一种解决方案是在几何处理阶段将由顶点着色器生成的图元顶点数据经过几何处理之后存储到 GPU 内存中，用于第二阶段渲染处理过程中读取。如图 3 所示，图 3 是一种 GPU 分块渲染示意性流程图。

从图 3 中可以看出几何处理阶段通过对图形应用程序输入图元的顶点着色器以及投影屏幕映射处理，从而产生图元在屏幕上的坐标。屏幕上不可见的图元在图元剔除阶段会被剔除。剩余的图元经过分块器将其分类到屏幕上的各个片块中，并将图元分块数据存储在 GPU 内存里面。同时图元剔除阶段后剩余图元的顶点数据也会被存储在 GPU 内存中。

在渲染阶段光栅生成器将属于本片块的图元分块数据从 GPU 内存中读取出来，同时将图元的顶点数据从内存中读取出来。这些属于该片块的图元顶点数据将被用在 GPU 渲染阶段，进而产生该片块的输出图像。

图 3 所示的方案虽然可以避免在渲染阶段重复执行顶点着色器来产生图元顶点数据，但是存储顶点着色器产生的图元顶点数据会增加 GPU 的内存需求，并且图元顶点数据的读写通常会增加 GPU 的带宽。特别是在 GPU 的内存以及带宽资源有限的情况下，存储大量的几何处理后的图元顶点数据可能会影响 GPU 的总体性能。

另一种解决方案是在第二阶段渲染处理过程中重新对图元做几何处理，由顶点着色器生成图元顶点数据。如图 4 所示，图 4 是另一种 GPU 分块渲染示意性流程图。

从图 4 中可以看出在几何处理阶段中由图形应用程序输入的图元经过顶点着色器以及投影屏幕映射处理从而产生图元在屏幕上的坐标。屏幕上不可见的图元在图元剔除阶段会被剔除。剩余的图元将被送到第二阶段做渲染处理。图元剔除阶段后图元的顶点数据没有被保存在 GPU 内存里面。

在渲染阶段经过几何处理的图元再一次经过顶点着色器而重新产生图元顶点在屏幕上的坐标。这些重新生成的图元顶点数据将被用在光栅生成器中做光栅化处理，进而用在 GPU 的整个渲染阶段以产生输出图像。

图 4 所示的方案在 GPU 渲染阶段需要再一次经过顶点着色器处理来重新生成的图元顶点数据。在渲染阶段重复执行顶点着色器会增加 GPU 的运算量，进而影响 GPU 的渲染性能。在几何处理阶段可以执行简化的顶点着色器产生几何处理阶段所需的顶点位置数据，而在渲染阶段再执行完整的顶点着色器来产生所有的图元顶点数据。即使采用了这个优化措施在复杂的顶点着色计算场景下重复执行顶点着色器对 GPU 运算量以及渲染性能的影响也是不可忽略的。

为了解决目前的 GPU 分块渲染所存在的问题，本申请提出一种图形渲染的方法，通过选择不同的渲染模式，减少对 GPU 内存和带宽的需求和顶点着色器的计算量，以提高 GPU 分块渲染系统的总体性能。

为便于理解本申请实施例，首先对本申请实施例中涉及的几个基本概念做简单说明。应理解，下文中所介绍的基本概念是以目前的相关技术中记载为例进行简单说明，本申请中对于具体名称并不限定。

1、渲染（Render）。

渲染是最终使图像符合三维（three dimensions，3D）场景的阶段。渲染有多种软件，如：各计算机绘图(computer graphics，CG)软件自带渲染引擎，还有诸如 RenderMan 等。建筑设计、动画制作等利用 3DS MAX、MAYA 等软件制作好模型、动画帧之后，将所设

计内容利用软件本身或者辅助软件（lightscape、vray 等）制作成最终效果图或者动画的过程。

2、分块渲染。

在分块渲染架构中，光栅化是在每一个块上执行的，这是与传统的立即渲染结构的主要区别，为了实现这一目的，所有待绘制的图元经过几何变换和裁剪之后都需要按其所在的块暂存在系统存储器里，一旦一帧的图元都存储完毕，就可以按照块的顺序逐块进行渲染，而传统的针对颜色缓冲区、深度缓冲区和模板缓冲区的读改写操作都可以在一个小的片上存储器上进行而无需重复的访问系统存储器，当一块内的所有图元渲染完成之后，可以将该块的片上存储器的内容一次性的写入存储器（一般只需要写回颜色缓冲区的内容），这种架构可以大大减小访存次数，在提高绘制效率的同时也降低了功耗。

3、顶点着色。

顶点着色是一种渲染方法。在顶点着色中，可以访问到顶点的三维位置、颜色、方向等信息。可以通过修改这些值、或者将其传递到片元着色阶段，实现特定的渲染效果。

4、图元。

图元是基本图形元素。基本图形元素是图形元素构造复杂的几何图形和图幅的基本图形实体。不同的图形系统有不同的图形元素。基本图形元素是指点、线、三角形和面片等。在光栅扫描显示器上显示的图形，都是具有一种或多种颜色的像素的集合。确定最佳逼近图形的像素集合，并用指定属性写像素的过程称图形的扫描转换或生成 。

5、图元系列。

本申请涉及的图元系列包括至少一个图元。

下面结合附图 5 对本申请实施例中的图形渲染的方法进行详细的介绍。

图 5 是本申请实施例提供的一种图形渲染的方法示意性流程图。该图形渲染的方法应用于 GPU 分块渲染系统（如图 1 所示的系统）中，该 GPU 分块渲染系统支持的渲染模式包括第一渲染模式和第二渲染模式，可以理解本申请实施例中 GPU 分块渲染系统支持的渲染模式并不是固定的某一种渲染模式，能够支持两种不同的渲染模式进行图形渲染处理。

其中，第一渲染模式中图形应用程序输入的图元系列在几何处理阶段生成的图元顶点数据存储在该 GPU 内存中，光栅处理阶段从内存中读取该图元顶点数据用于进行渲染产生输出图像（如上述图 3 所示的渲染方式）；第二渲染模式中图形应用程序输入的图元系列在几何处理阶段生成的图元顶点数据未存储在该 GPU 内存中，光栅处理阶段对几何处理阶段处理之后的图元进行顶点着色处理生成图元顶点数据，该图元顶点数据用于进行渲染产生输出图像（如上述图 4 所示的渲染方式），

示例性地，由于第一渲染模式下图元系列在几何处理阶段经过顶点着色器处理即可，在光栅处理阶段无需再次经过顶点着色器处理；而第二渲染模式下图元系列在几何处理阶段经过顶点着色器处理之后，在光栅处理阶段还需再次经过顶点着色器处理。从顶点着色器的执行模式的维度来说，第一渲染模式可以称为单一执行模式，第二渲染模式可以称为重复执行模式。

应理解，上述的“第一渲染模式”和“第二渲染模式”，或者，“单一执行模式”和“重复执行模式”，只是为了对本申请实施例中 GPU 分块渲染系统支持的不同渲染模式进行区分，

对本申请的保护范围不构成任何的限定，还可以通过其他的名称进行区分，这里不再一一举例说明。如，还可以用“存储模式”和“非存储模式”进行区分，其中，对几何处理阶段生成的图元顶点数据进行存储的模式称为存储模式；对几何处理阶段生成的图元顶点数据不进行存储的模式称为非存储模式。

该方法包括以下步骤：

S510，接收来自图形应用程序的图元系列。

本申请实施例中对于 GPU 分块渲染系统如何接收到图形应用程序输入的图元系列不做限制，可以参考目前相关技术中的描述。

示例性地，本申请实施例中涉及的图形应用程序可以是图 1 中所示的图形应用程序 110，图形应用程序输入的图元系列可以是图形应用程序 110 中的待渲染的图形中的部分图元。

作为一种可能的实现方式，图形应用程序可以将待渲染的图形中的图元分批次输入到 GPU 分块渲染系统中，由 GPU 分块渲染系统进行处理，其中，图形应用程序每次输入到 GPU 分块渲染系统中的图元称为图元系列，下文中图元系列也可以称为图元数据，或者称为图元输入系列，或者称为图元序列等。

应理解，上述图元系列、图元输入系列，或者称为图元序列等名称只是举例，对本申请的保护范围不构成任何的限定。本申请实施例中对于图形应用程序输入至 GPU 分块渲染系统中待处理的图元的具体名称不做限定，可以参考目前相关技术中的描述。

进一步地，GPU 分块渲染系统接收到图形应用程序输入的图元系列之后，需要对图元系列进行渲染处理，最后产生相应的输出图像。

本申请实施例中，GPU 分块渲染系统对图元系列进行渲染处理之前需要确定使用怎样的模式进行渲染，图 5 所示的方法流程还包括：

S520，根据预设条件确定使用第一渲染模式和/或第二渲染模式对所述图元系列进行渲染。

本申请实施例中 GPU 分块渲染系统进行渲染时能够根据预设的条件确定使用更合适的渲染模式对所述图元系列进行渲染，避免使用固定的渲染模式影响 GPU 的渲染性能（如固定使用上述图 3 所示的模式可能会增加 GPU 的内存需求；还如，固定使用上述图 4 所示的模式可能会增加 GPU 的运算量）。

作为一种可能的实现方式，可以根据预设的条件确定使用第一渲染模式对所述图元系列进行渲染；

作为另一种可能的实现方式，可以根据预设的条件确定使用第二渲染模式对所述图元系列进行渲染；

作为又一种可能的实现方式，可以根据预设的条件确定使用第一渲染模式和第二渲染模式对所述图元系列进行渲染，如，输入的图元系列中一部分图元使用第一渲染模式进行渲染，另一部分图元使用第二渲染模式进行渲染。

由上述可知，本申请实施例提供的图形渲染的方法，在渲染过程中从 GPU 分块渲染系统支持的多种渲染模式中选择适合当前图元系列的渲染模式，将会提高 GPU 分块渲染系统的总体性能。

具体地，根据预设条件确定使用的渲染模式之后，GPU 分块渲染系统能够基于已确

定的渲染模式进行图元渲染，图 5 所示的方法流程还包括:

S530，基于所述第一渲染模式和/或所述第二渲染模式对所述图元系列进行渲染。

应理解，本申请实施例中对于 GPU 分块渲染系统进行渲染的具体流程不做限定，例如，当确定采用第一渲染模式进行渲染时，具体的渲染流程可以参考上述图 3 所示的渲染流程；还例如，当确定采用第二渲染模式进行渲染时，具体的渲染流程可以参考上述图 4 所示的渲染流程；又例如，当确定采用第一渲染模式和第二渲染模式进行渲染时，采用第一渲染模式的图元具体的渲染流程可以参考上述图 3 所示的渲染流程，采用第二渲染模式的图元具体的渲染流程可以参考上述图 4 所示的渲染流程。

示例性地，本申请实施例中的预设条件包括以下至少一种：比例常数大于第一阈值、顶点着色器程序复杂程度的加权系数大于第二阈值、图元加权系数大于第三阈值、存储资源加权系数大于第四阈值或综合加权系数大于第五阈值。

应理解，上述的预设条件只是举例说明预设条件可能的形式，对本申请的保护范围不构成任何的限定，还可以设定其他的预设条件用于选择第一渲染模式和/或第二渲染模式。如，随机选择，还如，根据历史渲染记录等，这里不再一一举例说明。

为了便于理解，下面分别介绍上述的几种预设条件:

1）比例常数大于第一阈值。

其中，比例常数为第一渲染模式执行次数和所述第二渲染模式执行次数的比值，记为 Rc:

Rc = single_mode_count / repeat_mode_count。single_mode_count 表示第一渲染模式的执行次数；repeat_mode_count 表示第二渲染模式的执行次数。

第一渲染模式类似于图 3 所示的渲染模式可以称为单一执行模式；第二渲染模式类似于图 4 所示的渲染模式可以称为重复执行模式。

比例常数大于第一阈值表示第一渲染模式的执行次数和第二渲染模式的执行次数之间的比值小于预设的某个阈值，例如，第一阈值小于 1 表示第一渲染模式的执行次数少于第二渲染模式的执行次数相等。

示例性地，当满足比例常数大于第一阈值时，确定使用第一渲染模式；当不满足比例常数大于第一阈值时，确定使用第二渲染模式，或者，使用第一渲染模式和第二渲染模式。

示例性地，当满足比例常数大于第一阈值时，确定使用第二渲染模式；当不满足比例常数大于第一阈值时，确定使用第一渲染模式，或者，使用第一渲染模式和第二渲染模式。

2）顶点着色器程序复杂程度的加权系数大于第二阈值。

其中，顶点着色器程序复杂程度加权系数为顶点着色器程序的指令数、复杂指令数、内存读取指令数和循环执行指令数的加权值。

第二渲染模式的主要弱点是增加了顶点着色器的执行次数，从而影响 GPU 渲染过程的性能。不同的加权系数可以用来描述顶点着色器程序的复杂程度，从而用来决定是否选择第二渲染模式。

以下因素可以考虑作为顶点着色器程序复杂程度的加权系数：顶点着色器程序的指令数，复杂指令（例如 log，square root 等）数，内存读取指令（例如纹理读取）数，以及循环执行指令数。例如顶点着色器程序复杂程度的加权系数 Rs 可以定义为:

Rs = k1 * inst_count + k2 * complex_count + k3 * texute_count + k4 * loop_count

其中，inst_count 代表顶点着色器程序的总指令数；complex_count 代表顶点着色器程序的复杂指令数；texute_count 代表顶点着色器程序的内存读取指令数；loop_count 代表顶点着色器程序的循环指令数；k1，k2，k3，k4 则是各项指令数所对应的加权系数。

作为一种可能的实现方式，各项指令数所对应的加权系数可以根据实验数据来确定并调整，以达到最佳效果。

作为另一种可能的实现方式，各项指令数所对应的加权系数也可以根据 GPU 系统的设计来调整，如 k4 = 0 则代表在加权系数中不考虑循环指令对顶点着色器程序复杂程度的影响。

示例性地，当满足顶点着色器程序复杂程度的加权系数大于第二阈值时，确定使用第一渲染模式；当不满足顶点着色器程序复杂程度的加权系数大于第二阈值时，确定使用第二渲染模式，或者，使用第一渲染模式和第二渲染模式。

示例性地，当满足顶点着色器程序复杂程度的加权系数大于第二阈值时，确定使用第二渲染模式；当不满足顶点着色器程序复杂程度的加权系数大于第二阈值时，确定使用第一渲染模式，或者，使用第一渲染模式和第二渲染模式。

如上所述的用来描述顶点着色器程序复杂程度的加权系数可以在顶点着色器的编译过程中产生，从而实现依据顶点着色器程序特性来选择渲染模式。

在一般情况下对于复杂程度高 Rs 系数大的顶点着色器程序应该采用第一渲染模式，即将几何处理阶段产生的图元顶点数据存储在 GPU 内存里面，光栅处理阶段读取内存中的图元顶点数据，从而减少了对复杂的顶点着色器程序的重复执行的工作量。

对于复杂程度低 Rs 系数小的顶点着色器程序可以采用第二渲染模式，即在光栅处理阶段重复执行简单的顶点着色器程序来重新产生图元顶点数据。在几何处理阶段产生的图元顶点数据不需要存储在 GPU 内存里面，从而减少了对 GPU 内存空间以及带宽的需求。

进一步地，关于顶点着色器程序复杂度对渲染模式的影响，本申请提出了另外一个参考因素，即顶点着色器相比像素着色器计算量的比例。因为在屏幕上图元一般由多个像素组成，光栅处理阶段以像素为粒度的像素着色器的计算量会比几何处理阶段以图元顶点为粒度的顶点着色器的计算量大很多。

在某种特殊情况下，GPU 分块渲染系统中几何处理阶段顶点着色器的计算量有可能大于光栅处理阶段像素着色器的计算量。

例如，像素着色器的程序可能非常简单，而顶点着色器的程序可能相对比较复杂。在这种特殊情况下，应该避免选择第一渲染模式，进而避免加重顶点着色器的工作量。在几何处理阶段执行简化的顶点着色器程序来产生图元顶点的位置数据，用于对图元的几何处理和分块处理，而在光栅处理阶段重新执行顶点着色器程序来产生图元顶点数据。

3）图元加权系数大于第三阈值。

其中，图元加权系数为图元覆盖和图元顶点数据字节数的加权值。

在 GPU 分块渲染系统中，经过几何处理阶段后的图元由分块器分配到屏幕上。

每个该图元所覆盖的片块里在光栅处理阶段渲染过程将会在屏幕上的各个片块进行分别处理。如果顶点着色器的执行方式是第二渲染模式，在各个片块的渲染过程将对图元再一次执行顶点着色器程序来产生图元的顶点数据。对于在屏幕上覆盖多个片块的图元，顶点着色器程序将有可能需要在每个图元所覆盖的片块里重复执行。这种顶点着色器程序

的多次重复执行会进一步增加顶点着色器的工作量，从而影响 GPU 分块渲染系统的性能。

另外一个可能影响到渲染模式选择的图元特性是：图元顶点数据的大小。在渲染过程图元顶点数据的字节数可能在 16 到 512 的范围内。对于同样数量的图元顶点数据，用第一渲染模式，存储图元顶点数据字节数多的图元系列将会需要更多的 GPU 内存空间及带宽。因此图元顶点数据字节数多的图元系列适合应用第二渲染模式。

本申请提出的图元几何加权法利用图元几何特性来帮助选择更加优化的渲染模式。例如对于一个特定的图元系列，其图元加权系数 Rg 可以定义为：

$$Rg = k1 * (tiles / primitives) + k2 * (vertices * (max\_size - vertex\_size))$$

其中，tiles 代表该图元系列图元在屏幕上覆盖的片块总数；primitives 代表该图元系列中的图元总数；vertices 代表该图元系列图元顶点的总数；vertex_size 代表该图元系列图元顶点的字节数；max_size 代表图元顶点数据字节数的最大值，例如 512 字节；而 k1，k2 则是图元覆盖以及图元顶点数据字节数所对应的加权系数。

作为一种可能的实现方式，加权系数 k1，k2 可以根据实验数据来确定并调整，以达到最佳效果。

作为另一种可能的实现方式，加权系数 k1，k2 也可以根据 GPU 系统的设计来调整，例如 k2 = 0 则代表在加权系数中不考虑图元顶点数据字节数对渲染模式的影响。

示例性地，当满足图元加权系数大于第三阈值时，确定使用第一渲染模式；当不满足图元加权系数大于第三阈值时，确定使用第二渲染模式，或者，使用第一渲染模式和第二渲染模式。

示例性地，当满足图元加权系数大于第三阈值，确定使用第二渲染模式；当不满足图元加权系数大于第三阈值时，确定使用第一渲染模式，或者，使用第一渲染模式和第二渲染模式。

如上所述的图元系列的图元加权系数 Rg 可以用来选择该图元系列的渲染模式。对于图元加权系数 Rg 大的图元系列应该采用第一渲染模式，即将几何处理阶段产生的图元顶点数据存储在 GPU 内存里面，光栅处理阶段读取内存中的顶点数据。采用第一渲染模式减少了对片块覆盖率高的图元做顶点着色器程序重复执行的工作量，同时存储图元顶点数据字节数不大的顶点数据不会造成对 GPU 内存空间及带宽很大的额外需求。

对于图元加权系数 Rg 小的图元系列可以采用第二渲染模式，即在光栅处理阶段重复执行顶点着色器程序来重新产生图元顶点数据。在几何处理阶段产生的图元顶点数据不需要存储在 GPU 内存里面，从而避免了存储图元顶点数据字节数大的顶点数据对 GPU 内存空间以及带宽造成的额外需求。对片块覆盖率不高的图元做顶点着色器程序的重复执行，其增加的顶点着色器执行工作量也会控制在可以接受的范围内。

4）存储资源加权系数大于第四阈值。

其中，存储资源加权系数为所述 GPU 内存的存储资源利用率。

在 GPU 分块渲染系统中的第一渲染模式中，经过几何处理阶段由顶点着色器产生的顶点数据会被存储在 GPU 的内存里以供渲染过程光栅处理阶段中读取。顶点着色器的第一渲染模式可以避免在光栅处理阶段重复执行顶点着色器程序来产生渲染过程所需的图元顶点数据，但是该模式需要占用 GPU 内存的存储空间。通常情况下 GPU 内存的存储资源是有限的，当 GPU 内存的利用率达到极限时渲染模式就不应该选择第一渲染模式。

本申请提出的存储资源加权法利用 GPU 内存的存储资源利用率来帮助选择更加优化的渲染模式。例如存储资源加权系数 Rm 可以定义为:

Rm = (free_space / total_space)

其中，free_space 代表 GPU 内存中的可用存储空间；而 total_space 代表 GPU 内存的总存储空间。

在存储资源加权系数 Rm 大的情况下 GPU 内存中有充分的可用存储空间，渲染模式可以选择为第一渲染模式。在存储资源加权系数 Rm 小的情况下 GPU 内存中的可用存储空间有限，渲染模式就应该选择为第二渲染模式。

示例性地，当满足存储资源加权系数大于第四阈值时，确定使用第一渲染模式；当不满足存储资源加权系数大于第四阈值时，确定使用第二渲染模式，或者，使用第一渲染模式和第二渲染模式。

5）综合加权系数大于第五阈值。

其中，综合加权系数为所述比例常数、所述顶点着色器程序复杂程度的加权系数、所述图元加权系数和所述存储资源加权系数的加权值。

本申请提出可以将以上所述的顶点着色器多种执行模式的选择方法，顶点着色器加权法，图元几何加权法以及存储资源加权法结合起来，组成一个综合加权法以用来更有效的选择渲染模式。例如综合加权系数 R 可以定义为:

R = k1 * Rs + k2 * Rg + k3 * Rm

其中，Rs 代表顶点着色器加权法加权系数；Rg 代表图元几何加权法加权系数； Rm 代表存储资源加权法加权系数；k1，k2，k3 则是综合加权系数 R 中各项加权法所对应的加权系数。

作为一种可能的实现方式，各项加权法所对应的加权系数 k1，k2，k3 可以根据实验数据来确定并调整，以达到最佳效果。

作为另一种可能的实现方式，各项加权法所对应的加权系数也可以根据 GPU 系统的设计来调整，例如 k3 = 0 则代表在综合加权系数 R 中不考虑存储资源对渲染模式的影响。

通常在综合加权系数 R 大的情况下应该采用顶点着色器的第一渲染模式，即将几何处理阶段产生的图元顶点数据存储在 GPU 内存里面，光栅处理阶段读取内存中的顶点数据。第一渲染模式可以减少对复杂的顶点着色器程序重复执行的工作量，也可以减少对屏幕片块覆盖率高的图元做顶点着色器程序重复执行的工作量。同时综合加权系数 R 大的情况下也意味着 GPU 内存有充分的可用存储空间，因此不会影响到图元顶点数据的存储。

在综合加权系数 R 小的情况下可以采用顶点着色器的第二渲染模式，即在光栅处理阶段重复执行顶点着色器程序来重新产生图元顶点数据。第二渲染模式在几何处理阶段产生的图元顶点数据不需要存储在 GPU 内存里面，从而减少了对 GPU 内存空间以及带宽的需求。同时综合加权系数 R 小的情况下也意味着顶点着色器程序复杂程度低，图元片块覆盖率低，因此重复执行顶点着色器程序所带来的额外工作量对 GPU 分块渲染系统性能的影响不会太大。

示例性地，当满足综合加权系数大于第五阈值时，确定使用第一渲染模式；当不满足综合加权系数大于第五阈值时，确定使用第二渲染模式，或者，使用第一渲染模式和第二渲染模式。

6）渲染场景为预设的渲染场景。

本申请所提出的选择渲染模式的渲染场景控制法，可以以图形应用程序的渲染场景为粒度来控制对渲染模式的选择。用渲染场景控制法在一个渲染场景中做一次对渲染模式的选择即可，简化了控制模式，从而将会大大减少加权系数（如上述的顶点着色器程序复杂程度的加权系数、图元加权系数、存储资源加权系数等）的计算次数，以及渲染模式的变换频率。

示例性地，当渲染场景为简单渲染场景（如，屏幕后处理场景）时，确定使用第一渲染模式；当渲染场景为复杂渲染场景时，确定使用第二渲染模式。

用渲染场景控制法来控制对渲染模式选择的另外一种方案是利用图形应用程序在前一个渲染场景中的特性来预测当前渲染场景中渲染模式的选择。在相邻两个渲染场景的特性相差不大的情况下，利用前一个渲染场景中的特性来预测当前渲染场景中渲染模式会是一个简单有效的方案。

上述的 1）至 6）简单介绍了本申请涉及的预设条件，进一步地可以通过以下几种方式确定上述的预设条件:

方式一:

根据图形应用程序输入的图元系列确定预设条件包括所述顶点着色器程序复杂程度的加权系数大于第二阈值、所述图元加权系数大于第三阈值和所述存储资源加权系数大于第四阈值中的至少一项，

其中，所述图元系列所绑定的顶点着色器程序用于确定所述顶点着色器程序复杂程度的加权系数，所述图元系列中图元覆盖和图元顶点数据字节数用于确定所述图元加权系数，所述 GPU 内存的存储资源利用率用于确定所述存储资源加权系数。

进一步地，能够根据是否满足预设条件确定渲染模式。

当一个特定的图元系列由图形应用程序输入到 GPU 时，可以根据该图元系列所绑定的顶点着色器程序来计算选择渲染模式的顶点着色器加权系数。也可以根据该图元系列中图元及顶点数据的特性来计算选择渲染模式的图元几何加权系数。

另外针对当前 GPU 内存的利用率可以计算选择渲染模式的存储资源加权系数。这些加权系数可以分别用来选择顶点着色器在该图元系列的执行模式，或者综合各个加权系数用来选择顶点着色器在该图元系列的执行模式。

对一个特定的图元系列其对应的顶点着色器程序，图元几何特性以及 GPU 内存可用空间都是可知的。基于图元系列做渲染模式的选择方式简单明确。

为了便于理解，结合图 6 简单介绍方式一所示的根据图形应用程序输入的图元系列确定渲染模式的方法，图 6 是本申请实施例提供一种确定渲染模式的示意性流程图。

从图 6 中可以看出，该确定渲染模式的方式包括以下步骤:

S610，接收图元系列。

S621，根据图元系列计算顶点着色器加权系数 Rs。

具体地，当一个特定的图元系列由图形应用程序输入到 GPU 时，可以根据该图元系列所绑定的顶点着色器程序来计算顶点着色器加权系数 Rs。

S622，根据图元系列计算图元几何加权系数 Rg。

具体地，可以根据该图元系列中图元及顶点数据的特性来计算图元几何加权系数 Rg。

另外，GPU 内存的可用存储空间在渲染处理每一个图元系列时是确定的，所以可以将针对每一个图元系列的当前 GPU 内存存储资源加权系数考虑到对渲染模式的选择中。图 6 所示的方法流程还包括:

S623，计算存储资源加权系数 Rm。

针对当前 GPU 内存的利用率可以计算存储资源加权系数 Rm。

进一步地，根据预设条件（上述计算得到的 Rs，Rg 和 Rm）确定渲染模式包括以下步骤:

S631，判断顶点着色器程序复杂程度的加权系数是否大于第二阈值（Ss）。

如果顶点着色器加权系数 Rs 小于特定的常数 Ss 则选择第二渲染模式，否则执行步骤 S632。

S632，判断图元加权系数是否大于第三阈值（Sg）。

如果图元几何加权系数 Rg 小于特定的常数 Sg 则选择第二渲染模式，否则执行步骤 S633。

S633，判断存储资源加权系数是否大于第四阈值（Sm）。

如果存储资源加权系数 Rm 小于特定的常数 Sm 则选择第二渲染模式，否则将选择第一渲染模式。

需要说明的是，同一个顶点着色器程序可能会被绑定在不同的图元系列上，因此可以将顶点着色器程序编译过程中生成的顶点着色器加权系数 Rs 存储起来，以避免在处理与该顶点着色器程序绑定的其他图元系列时对顶点着色器加权系数做重复计算。

图 6 所示的渲染模式的选择方式: 将顶点着色器程序特性，图元系列的几何特性以及 GPU 内存存储空间特性综合考虑在内，只是举例说明可以通过该方式进行渲染模式选择，对本申请的保护范围不构成任何的限定，还可以通过其他方式选择渲染模式。如，不考虑 GPU 内存存储空间特性和/或图元系列的几何特性，如果顶点着色器加权系数 Rs 小于特定的常数 Ss 则选择第二渲染模式，否则将选择第一渲染模式。这里不一一举例说明。

方式二:

根据图形应用程序输入的顶点着色器程序确定预设条件包括顶点着色器程序复杂程度的加权系数大于第二阈值。

本申请所提出的选择渲染模式的顶点着色器控制法，以图形应用程序所定义的顶点着色器程序为粒度来控制对渲染模式的选择。当一个特定的顶点着色器程序由图形应用程序输入到 GPU 时，可以根据对该顶点着色器程序的编译来计算选择渲染模式的顶点着色器加权系数。

对一个特定的顶点着色器程序其对应的程序指令数量和特性都是可知的。

由于同一个顶点着色器程序可以绑定到多个图元系列上，因此用顶点着色器控制法做渲染模式的选择方式不但可以充分利用顶点着色器程序的特性来选择适应的渲染模式，而且相对上述的方式一减少了加权系数的计算（如上述的图元几何加权系数 Rg 和存储资源加权系数 Rm），以及减少了可能导致的渲染模式的变换（如上述的图元几何加权系数 Rg 和存储资源加权系数 Rm 不满足条件时导致的渲染模式的变换）。

为了便于理解，结合图 7 简单介绍方式二所示的根据图形应用程序输入的图元系列确定渲染模式的方法，图 7 是本申请实施例提供另一种确定渲染模式的示意性流程图。

从图 7 中可以看出，该确定渲染模式的方式包括以下步骤：

S710，接收顶点着色器程序。

S720，确定程序指令。

进一步地，可以根据程序指令确定程序指令数。

具体地，对于某个特定的顶点着色器程序，在顶点着色器程序的编译过程中可以得到代表该顶点着色器程序特性的参数，例如顶点着色器程序的指令数，复杂指令数，内存读取指令数以及循环指令数。

进一步地，根据指令数计算顶点着色器加权系数 Rs。图 7 所示的方法流程还包括：

S730，计算顶点着色器加权系数 Rs。

具体地，根据上述的顶点着色器加权法以及顶点着色器程序的特性参数可以确定顶点着色器程序对渲染模式选择的加权系数 Rs。

进一步地，根据预设条件（上述计算得到的 Rs）确定渲染模式包括以下步骤：

S740，判断顶点着色器程序复杂程度的加权系数是否大于第二阈值（Ss）。

如果顶点着色器加权系数 Rs 小于特定的常数 Ss 则选择第二渲染模式，否则将选择第一渲染模式。

在对于某个特定的顶点着色器程序选择了其相对应的渲染模式之后，该执行模式将会应用在所有与该顶点着色器程序绑定的图元系列的渲染过程中。

图 7 所示的实施例可以有效地利用顶点着色器程序的特性来选择渲染模式，从而提高顶点着色器的执行效率，以及提高 GPU 内存的有效利用率。

进一步地，本申请实施例中顶点着色器可以采用分段输出的方式：可以将顶点着色器编译成两个独立的程序，一个专门用来产生图元顶点位置数据的顶点着色器程序和另外一个用来产生图元顶点属性数据的顶点着色器程序。

应理解，GPU 中几何处理阶段，顶点着色器通过执行图形应用程序所提供的顶点着色器程序来产生图元顶点数据，包括顶点位置数据以及属性数据，进而确定三维图形的形状及位置关系。如图 3 或图 4 所示的 GPU 分块渲染流程中的几何处理阶段，经过顶点着色器处理以及投影及屏幕映射的图元还会通过图元剔除阶段来去除在屏幕上不可见的图元。因此几何处理阶段最后输出的图元总数会少于顶点着色器处理的图元数量。

其中，位置数据是固定的，而顶点属性数据的数量是不固定的，通常顶点属性数据的数量会远大于顶点位置数据。而图元剔除阶段需要图元顶点的位置数据，以及极少数与图元剔除有关的特殊图元顶点属性数据。根据此特性可以将图形应用程序所提供的顶点着色器程序编译成两个独立的顶点着色器程序：一个用来产生图元顶点位置数据的顶点着色器程序；另外一个用来产生图元顶点属性数据的顶点着色器程序。

需要说明的是，用来产生图元顶点位置数据的顶点着色器程序中的指令数量以及执行程序所需的输入输出数据量通常会有所减少，程序较为简单。因此在需要图元顶点位置数据的阶段，采用此种简化的顶点着色器程序会减少渲染过程的计算量，从而提高 GPU 系统的性能。

为了便于理解，下面结合图 8 介绍本申请实施例中涉及的顶点着色器的分段输出的方式，图 8 是本申请实施例提供的一种 GPU 系统示意性框图。

示例性地，第一渲染模式下，在所述几何处理阶段基于第一顶点着色器生成图元顶点

位置数据，所述图元顶点位置数据用于剔除屏幕上不可见的图元；以及基于第二顶点着色器生成所述屏幕上可见的图元对应的图元顶点属性数据。

例如，如图 8 所示在几何处理阶段顶点着色器#1 执行产生图元顶点位置数据的顶点着色器程序。经过顶点着色器#1 产生的图元顶点位置数据将用于后序流程中，用来剔除屏幕上不可见的图元。没有被剔除的图元再通过顶点着色器#2 来执行产生图元顶点属性数据的顶点着色器程序，从而产生图元顶点属性数据。

如此顶点着色器的分段输出模式可用于第一渲染模式，对剔除后的图元顶点执行计算量大的产生图元顶点属性数据的顶点着色器程序，从而进一步减少顶点着色器的计算量，提高 GPU 系统的性能。

具体地，在第一渲染模式下几何处理阶段图元经由图 8 中所示的顶点着色器#1 和顶点着色器#2 来产生图元顶点数据。经过屏幕投影及图元剔除处理之后，图元顶点数据被用于分块器中将图元分类到屏幕上不同的片块里。同时图元剔除之后的图元顶点数据将被存储到 GPU 内存中。在光栅处理阶段图元渲染所需的图元顶点数据从 GPU 内存读取出来，从而不需要经过顶点着色器#3 和顶点着色器#4（在第一模式下顶点着色器#3 和顶点着色器#4 可以没有）。在此第一渲染模式下顶点着色器程序可以编译为产生顶点位置数据的程序，和产生顶点属性数据的程序。如图 8 所示，顶点着色器#1 执行简化的顶点着色器程序来产生顶点的位置数据以减少计算量。经过图元剔除之后的图元再经过顶点着色器#2 来产生顶点的属性数据。由于产生顶点属性数据的顶点着色器程序需对剔除后的图元进行处理，所以进一步的减少了顶点着色器计算量，提高了 GPU 系统的性能。

示例性地，第二渲染阶段下，在所述光栅处理阶段基于第三顶点着色器生成图元顶点位置数据，所述图元顶点位置数据用于剔除屏幕上不可见的图元；以及基于第四顶点着色器生成所述屏幕上可见的图元对应的图元顶点属性数据。

类似于几何处理阶段顶点着色器的配置，对于第二渲染模式，在图 8 中的光栅处理阶段顶点着色器#3 执行计算量少的产生图元顶点位置数据的顶点着色器程序。经过顶点着色器#3 产生的图元顶点位置数据将用于后序流程中对图元做光栅化，以及深度测试来剔除屏幕上不可见的图元像素。深度测试阶段一些被遮挡的图元会被剔除，没有被剔除的图元再通过顶点着色器#4 来执行产生图元顶点属性数据的顶点着色器程序，从而产生后序流程中像素着色器所需的图元顶点属性数据。

如此顶点着色器的分段输出模式，对深度测试后的图元顶点执行计算量大的产生图元顶点属性数据的顶点着色器程序，从而进一步减少顶点着色器重复执行的计算量，提高 GPU 系统的性能。

具体地，在第二渲染模式下几何处理阶段图元经由执行图 8 中顶点着色器#1 来产生图元顶点数据。经过屏幕投影及图元剔除处理之后，被用于分块器中将图元分类到屏幕上不同的片块里。顶点着色器#1 产生的图元顶点数据将不会被存储到 GPU 内存中。由于在几何处理阶段需要图元顶点的位置数据，因此在第二渲染模式下顶点着色器#1 执行简化的顶点着色器程序来产生顶点的位置数据即可，不需要经过顶点着色器#2（在第一模式下顶点着色器#2 可以没有）。在光栅处理阶段图元经过顶点着色器#3 的重复执行来产生图元渲染所需的图元顶点数据，在第二渲染模式中顶点着色器产生顶点属性数据的程序可以推迟到光栅处理阶段中的深度测试之后，在顶点着色器#4 中执行，而顶点着色器#3 执行

简化的顶点着色器程序来产生顶点的位置数据，用在光栅生成器和深度测试之中。

需要说明的是，本申请实施例中主要以 GPU 分块渲染系统支持的渲染模式包括第一渲染模式和第二渲染模式为例进行说明的，对本申请的保护范围不构成任何的限定，GPU 分块渲染系统支持的渲染模式还可以为其他的渲染模式。例如，本申请提出的另外一种渲染模式：将顶点着色器产生的顶点位置数据存储在 GPU 内存中。类似于第一渲染模式，在图 8 所示的分块渲染系统中顶点着色器#1 所产生的顶点位置数据经过屏幕投影及图元剔除之后，被用于分块器中将图元分类到屏幕上不同的片块里。同时图元剔除之后的顶点位置数据将被存储到 GPU 内存里。在此模式中顶点属性数据将不会被存储到 GPU 内存里，因此不需要顶点着色器#2。在光栅处理阶段存储在 GPU 内存里的顶点位置数据会被读取出来，用于对图元的光栅生成和深度测试处理，不再需要经过顶点着色器#3。经过深度测试后的图元将通过顶点着色器#4 来产生渲染所需的顶点属性数据。此渲染模式相对第一渲染模式减少了顶点数据存储的内存空间及带宽，同时第二渲染模式减少了顶点着色器重复执行工作量。另外由于存储顶点位置数据所需空间是固定的，因此更便于对 GPU 内存存储空间的管理。

在本申请的实施例中，通过根据预设条件选择渲染模式进行渲染，避免基于固定模式进行渲染影响 GPU 的渲染性能。

应理解，上述举例说明是为了帮助本领域技术人员理解本申请实施例，而非要将本申请实施例限于所例示的具体数值或具体场景。本领域技术人员根据所给出的上述举例说明，显然可以进行各种等价的修改或变化，这样的修改或变化也落入本申请实施例的范围内。

上文结合图 5 至图 8，详细描述了本申请实施例提供的图形渲染的方法，下面将结合图 9 和图 10，详细描述本申请的装置实施例。应理解，本申请实施例中的用于图形渲染装置可以执行前述本申请实施例的各种方法，即以下各种产品的具体工作过程，可以参考前述方法实施例中的对应过程。

图 9 是本申请实施例的图形渲染装置 900 的示意性框图。应理解，装置 900 能够执行图 5 至图 8 的方法中的各个步骤，为了避免重复，此处不再详述。该图形渲染装置 900 可以是电子设备，或者，图形渲染装置 900 可以是配置于电子设备中的芯片。图形渲染装置 900 包括：

接收单元 910、处理单元 920 和渲染单元 930。

接收单元 910，用于接收来自图形应用程序的图元系列；

处理单元 920，用于根据预设条件确定使用该第一渲染模式和/或该第二渲染模式进行图形渲染对该图元系列进行渲染；

渲染单元 930，用于基于该第一渲染模式和/或该第二渲染模式对该图元系列进行渲染；

其中，该第一渲染模式指示将在几何处理阶段对该图元系列进行顶点着色处理生成的图元顶点数据存储在该 GPU 内存中，光栅处理阶段从该 GPU 内存中读取该图元顶点数据进行渲染处理，

该第二渲染模式指示未将在几何处理阶段对该图元系列进行顶点着色处理生成的图元顶点数据存储在该 GPU 内存中，该光栅处理阶段对该图元系列进行顶点着色处理生成

该图元顶点数据并对该图元顶点数据进行渲染处理。

应理解，这里的图形渲染装置900以功能单元的形式体现。这里的术语“单元”可以通过软件和/或硬件形式实现，对此不作具体限定。例如，“单元”可以是实现上述功能的软件程序、硬件电路或二者结合。所述硬件电路可能包括应用特有集成电路（application specific integrated circuit，ASIC）、电子电路、用于执行一个或多个软件或固件程序的处理器（例如共享处理器、专有处理器或组处理器等）和存储器、合并逻辑电路和/或其它支持所描述的功能的合适组件。

因此，在本申请的实施例中描述的各示例的单元，能够以电子硬件、或者计算机软件和电子硬件的结合来实现。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本申请的范围。

图10示出了本申请一个实施例提供的服务器1000的示意性框图。该服务器1000包括处理器1020、存储器1030、通信接口1040和总线1050。其中，处理器1020、存储器1030、通信接口1040通过总线1050进行通信，也可以通过无线传输等其他手段实现通信。该存储器1030用于存储指令，该处理器1020用于执行该存储器1030存储的指令。该存储器1030存储程序代码1011，且处理器1020可以调用存储器1030中存储的程序代码1011执行图5至图8所示的图形渲染的方法。

该存储器1030可以包括只读存储器和随机存取存储器，并向处理器1020提供指令和数据。存储器1030还可以包括非易失性随机存取存储器。该存储器1030可以是易失性存储器或非易失性存储器，或可包括易失性和非易失性存储器两者。其中，非易失性存储器可以是只读存储器（read-only memory，ROM）、可编程只读存储器（programmable ROM，PROM）、可擦除可编程只读存储器（erasable PROM，EPROM）、电可擦除可编程只读存储器（electrically EPROM，EEPROM）或闪存。易失性存储器可以是随机存取存储器（random access memory，RAM），其用作外部高速缓存。通过示例性但不是限制性说明，许多形式的RAM可用，例如静态随机存取存储器（static RAM，SRAM）、动态随机存取存储器（DRAM）、同步动态随机存取存储器（synchronous DRAM，SDRAM）、双倍数据速率同步动态随机存取存储器（double data date SDRAM，DDR SDRAM）、增强型同步动态随机存取存储器（enhanced SDRAM，ESDRAM）、同步连接动态随机存取存储器（synchlink DRAM，SLDRAM）和直接内存总线随机存取存储器（direct rambus RAM，DR RAM）。

该总线1050除包括数据总线之外，还可以包括电源总线、控制总线和状态信号总线等。但是为了清楚说明起见，在图10中将各种总线都标为总线1050。

如图10所示的服务器1000，其中，处理器1020可以执行图9所示的处理单元920对应的步骤/功能，通信接口1040可以执行图9所示的接收单元910对应的步骤/功能。

应理解，图10所示的服务器1000能够实现图5至图8所示方法实施例中设备执行的各个过程。服务器1000中的各个模块的操作和/或功能，分别为了实现上述方法实施例中设备的相应流程。具体可参见上述方法实施例中的描述，为避免重复，此处适当省略详细描述。

在本说明书中使用的术语“部件”、“模块”、“系统”等用于表示计算机相关的实体、硬

件、固件、硬件和软件的组合、软件、或执行中的软件。例如，部件可以是但不限于，在处理器上运行的进程、处理器、对象、可执行文件、执行线程、程序和/或计算机。通过图示，在计算设备上运行的应用和计算设备都可以是部件。一个或多个部件可驻留在进程和/或执行线程中，部件可位于一个计算机上和/或分布在 2 个或更多个计算机之间。此外，这些部件可从在上面存储有各种数据结构的各种计算机可读介质执行。部件可例如根据具有一个或多个数据分组（例如来自与本地系统、分布式系统和/或网络间的另一部件交互的二个部件的数据，例如通过信号与其它系统交互的互联网）的信号通过本地和/或远程进程来通信。

本领域普通技术人员可以意识到，结合本文中所公开的实施例描述的各示例的单元及算法步骤，能够以电子硬件、或者计算机软件和电子硬件的结合来实现。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本申请的范围。

所属领域的技术人员可以清楚地了解到，为描述的方便和简洁，上述描述的系统、装置和单元的具体工作过程，可以参考前述方法实施例中的对应过程，在此不再赘述。

在本申请所提供的几个实施例中，应该理解到，所揭露的系统、装置和方法，可以通过其它的方式实现。例如，以上所描述的装置实施例仅仅是示意性的，例如，所述单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个系统，或一些特征可以忽略，或不执行。另一点，所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口，装置或单元的间接耦合或通信连接，可以是电性，机械或其它的形式。

所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

另外，在本申请各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。

所述功能如果以软件功能单元的形式实现并作为独立的产品销售或使用时，可以存储在一个计算机可读取存储介质中。基于这样的理解，本申请的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得一台计算机设备（可以是个人计算机，服务器，或者网络设备等）执行本申请各个实施例所述方法的全部或部分步骤。而前述的存储介质包括：U 盘、移动硬盘、只读存储器（Read-Only Memory，ROM）、随机存取存储器（Random Access Memory，RAM）、磁碟或者光盘等各种可以存储程序代码的介质。

以上所述，仅为本申请的具体实施方式，但本申请的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本申请揭露的技术范围内，可轻易想到变化或替换，都应涵盖在本申请的保护范围之内。因此，本申请的保护范围应以所述权利要求的保护范围为准。

权 利 要 求 书

1. 一种图形渲染的方法，其特征在于，应用于计算机图形处理器 GPU 分块渲染系统中，所述 GPU 分块渲染系统支持的渲染模式包括第一渲染模式和第二渲染模式，

所述方法包括:

接收来自图形应用程序的图元系列;

根据预设条件确定使用所述第一渲染模式和/或所述第二渲染模式对所述图元系列进行渲染;

基于所述第一渲染模式和/或所述第二渲染模式对所述图元系列进行渲染;

其中，所述第一渲染模式指示将在几何处理阶段进行顶点着色处理生成的图元顶点数据存储在所述 GPU 内存中，光栅处理阶段从所述 GPU 内存中读取所述图元顶点数据进行渲染处理，

所述第二渲染模式指示未将在几何处理阶段进行顶点着色处理生成的图元顶点数据存储在所述 GPU 内存中，所述光栅处理阶段进行顶点着色处理生成所述图元顶点数据并对所述图元顶点数据进行渲染处理。

2. 如权利要求 1 所述的方法，其特征在于，所述预设条件包括以下至少一种:

比例常数大于第一阈值、顶点着色器程序复杂程度的加权系数大于第二阈值、图元加权系数大于第三阈值、存储资源加权系数大于第四阈值或综合加权系数大于第五阈值，

其中，所述比例常数为所述第一渲染模式执行次数和所述第二渲染模式执行次数的比值，所述顶点着色器程序复杂程度加权系数为顶点着色器程序的指令数、复杂指令数、内存读取指令数和循环执行指令数的加权值，所述图元加权系数为图元覆盖和图元顶点数据字节数的加权值，所述存储资源加权系数为所述 GPU 内存的存储资源利用率，所述综合加权系数为所述比例常数、所述顶点着色器程序复杂程度的加权系数、所述图元加权系数和所述存储资源加权系数的加权值。

3. 如权利要求 2 所述的方法，其特征在于，所述方法还包括:

根据所述图元系列确定预设条件包括所述顶点着色器程序复杂程度的加权系数大于第二阈值、所述图元加权系数大于第三阈值和所述存储资源加权系数大于第四阈值中的至少一项，

其中，所述图元系列所绑定的顶点着色器程序用于确定所述顶点着色器程序复杂程度的加权系数，所述图元系列中图元覆盖和图元顶点数据字节数用于确定所述图元加权系数，所述 GPU 内存的存储资源利用率用于确定所述存储资源加权系数。

4. 如权利要求 2 所述的方法，其特征在于，所述方法还包括:

接收来自所述图形应用程序的顶点着色器程序;

根据所述顶点着色器程序确定预设条件包括顶点着色器程序复杂程度的加权系数大于第二阈值。

5. 如权利要求 1 所述的方法，其特征在于，所述预设条件还包括: 渲染场景为预设的渲染场景。

6. 如权利要求 1 至 5 中任一项所述的方法，其特征在于，在所述第一渲染模式下，

所述方法还包括:

在所述几何处理阶段基于第一顶点着色器生成图元顶点位置数据，所述图元顶点位置数据用于剔除屏幕上不可见的图元；以及

基于第二顶点着色器生成所述屏幕上可见的图元对应的图元顶点属性数据。

7. 如权利要求 1 至 6 中任一项所述的方法，其特征在于，在所述第二渲染阶段下，所述方法还包括:

在所述光栅处理阶段基于第三顶点着色器生成图元顶点位置数据，所述图元顶点位置数据用于剔除屏幕上不可见的图元；以及

基于第四顶点着色器生成所述屏幕上可见的图元对应的图元顶点属性数据。

8. 一种图形渲染的装置，其特征在于，应用于计算机图形处理器 GPU 分块渲染系统中，所述 GPU 分块渲染系统支持的渲染模式包括第一渲染模式和第二渲染模式，

所述装置包括:

接收单元，用于接收来自图形应用程序的图元系列;

处理单元，用于根据预设条件确定使用所述第一渲染模式和/或所述第二渲染模式进行图形渲染对所述图元系列进行渲染;

渲染单元，用于基于所述第一渲染模式和/或所述第二渲染模式对所述图元系列进行渲染;

其中，所述第一渲染模式指示将在几何处理阶段进行顶点着色处理生成的图元顶点数据存储在所述 GPU 内存中，光栅处理阶段从所述 GPU 内存中读取所述图元顶点数据进行渲染处理，

所述第二渲染模式指示未将在几何处理阶段进行顶点着色处理生成的图元顶点数据存储在所述 GPU 内存中，所述光栅处理阶段进行顶点着色处理生成所述图元顶点数据并对所述图元顶点数据进行渲染处理。

9. 如权利要求 8 所述的装置，其特征在于，所述预设条件包括以下至少一种:

比例常数大于第一阈值、顶点着色器程序复杂程度的加权系数大于第二阈值、图元加权系数大于第三阈值、存储资源加权系数大于第四阈值或综合加权系数大于第五阈值，

其中，所述比例常数为所述第一渲染模式执行次数和所述第二渲染模式执行次数的比值，所述顶点着色器程序复杂程度加权系数为顶点着色器程序的指令数、复杂指令数、内存读取指令数和循环执行指令数的加权值，所述图元加权系数为图元覆盖和图元顶点数据字节数的加权值，所述存储资源加权系数为所述 GPU 内存的存储资源利用率，所述综合加权系数为所述比例常数、所述顶点着色器程序复杂程度的加权系数、所述图元加权系数和所述存储资源加权系数的加权值。

10. 如权利要求 9 所述的装置，其特征在于，所述处理单元还用于:

根据所述图元系列确定预设条件包括所述顶点着色器程序复杂程度的加权系数大于第二阈值、所述图元加权系数大于第三阈值和所述存储资源加权系数大于第四阈值中的至少一项，

其中，所述图元系列所绑定的顶点着色器程序用于确定所述顶点着色器程序复杂程度的加权系数，所述图元系列中图元覆盖和图元顶点数据字节数用于确定所述图元加权系数，所述 GPU 内存的存储资源利用率用于确定所述存储资源加权系数。

11. 如权利要求 9 所述的装置，其特征在于，

所述接收单元还用于接收来自所述图形应用程序的顶点着色器程序；

所述处理单元还用于根据所述顶点着色器程序确定预设条件包括顶点着色器程序复杂程度的加权系数大于第二阈值。

12. 如权利要求 8 所述的装置，其特征在于，所述预设条件还包括：渲染场景为预设的渲染场景。

13. 如权利要求 8 至 12 中任一项所述的装置，其特征在于，所述 GPU 分块渲染系统包括：

第一顶点着色器、图元剔除模块和第二顶点着色器；

所述第一顶点着色器，用于在所述几何处理阶段生成图元顶点位置数据；

所述图元剔除模块，用于根据所述图元顶点位置数据剔除屏幕上不可见的图元；

所述第二顶点着色器，用于生成所述屏幕上可见的图元对应的图元顶点属性数据。

14. 如权利要求 8 至 13 中任一项所述的装置，其特征在于，所述 GPU 分块渲染系统包括：

第三顶点着色器、光栅生成器、深度测试单元和第四顶点着色器；

所述第三顶点着色器，用于在所述光栅处理阶段生成图元顶点位置数据；

所述光栅生成器，用于根据所述图元顶点位置数据对图元进行光栅化；

所述深度测试单元，用于根据所述图元顶点位置数据剔除屏幕上不可见的图元；

所述第四顶点着色器，用于生成所述屏幕上可见的图元对应的图元顶点属性数据。

15. 一种计算机可读存储介质，其特征在于，所述计算机可读存储介质上存储有计算机程序，当所述计算机程序在计算机上运行时，使得计算机执行如权利要求 1 至 7 中任意一项所述的方法。

16. 一种包含指令的计算机程序产品，其特征在于，所述包含指令的计算机程序产品在计算机上运行时，使得计算机执行如权利要求 1 至 7 中任意一项所述的方法。

17.一种芯片系统，其特征在于，包括：处理器，用于从存储器中调用并运行计算机程序，使得安装有所述芯片系统的装置执行如权利要求 1 至 7 中任意一项所述的方法。

18.一种图形渲染的装置，其特征在于，包括：

存储器，用于存储计算机程序；

处理器，用于执行所述存储器中存储的计算机程序，以使得所述装置执行权利要求 1 至 7 中任一项所述的方法。

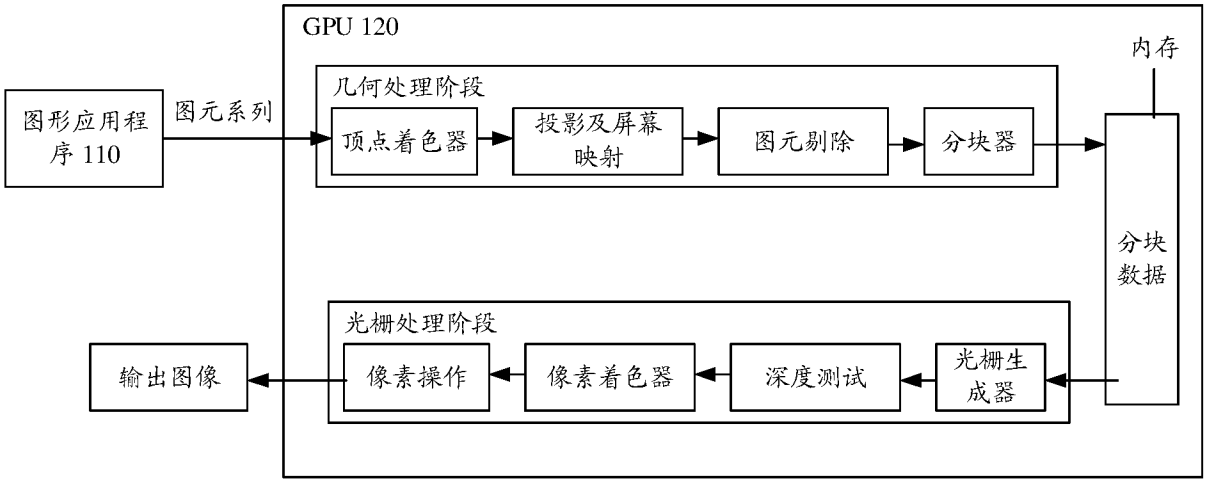
GPU 120
内存
图形应用程序 110
图元系列
几何处理阶段
顶点着色器
投影及屏幕映射
图元剔除
分块器
分块数据
光栅处理阶段
输出图像
像素操作
像素着色器
深度测试
光栅生成器

图 1

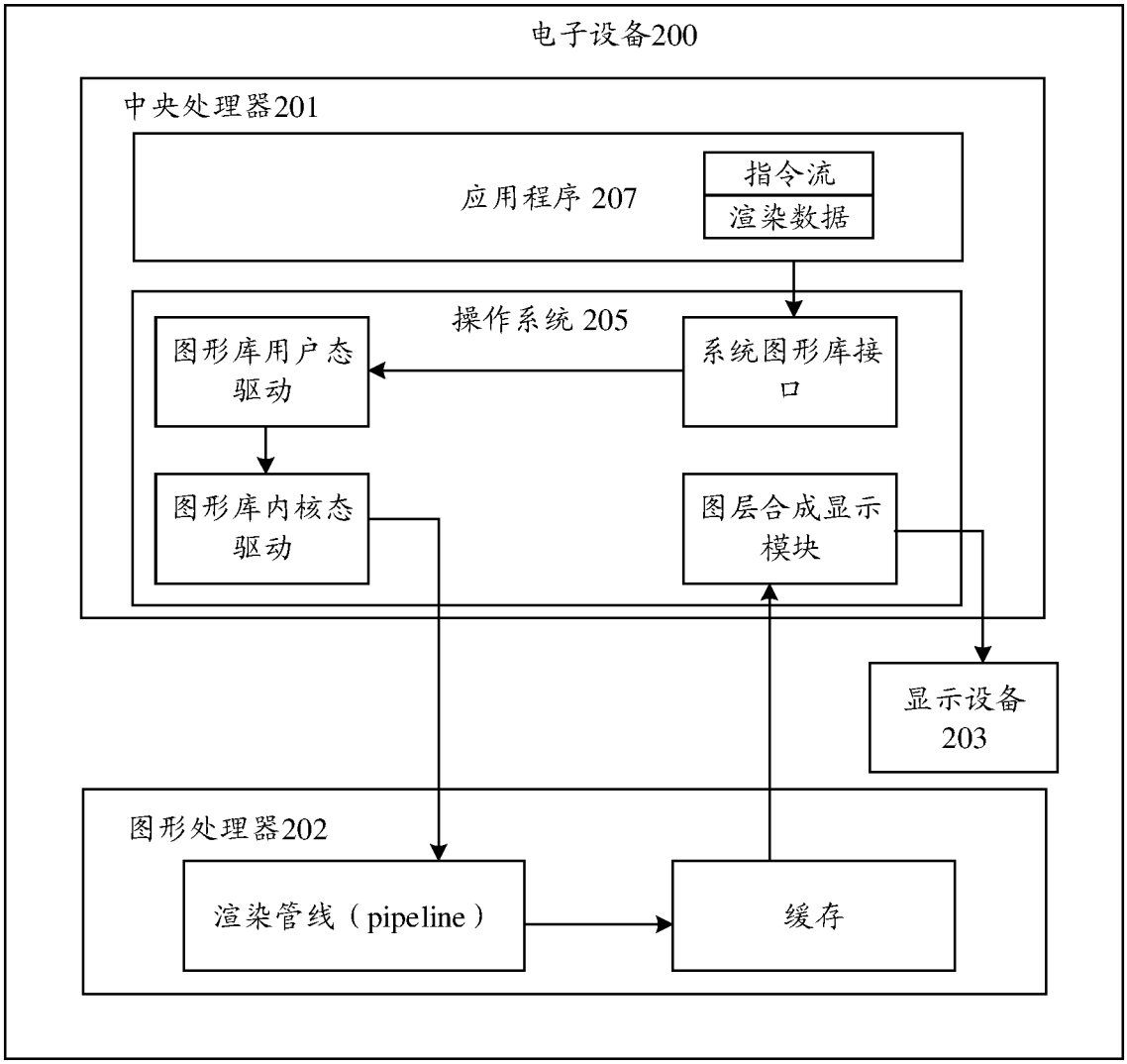
电子设备200
中央处理器201
应用程序 207
指令流
渲染数据
操作系统 205
图形库用户态驱动
系统图形库接口
图形库内核态驱动
图层合成显示模块
显示设备203
图形处理器202
渲染管线（pipeline）
缓存

图 2

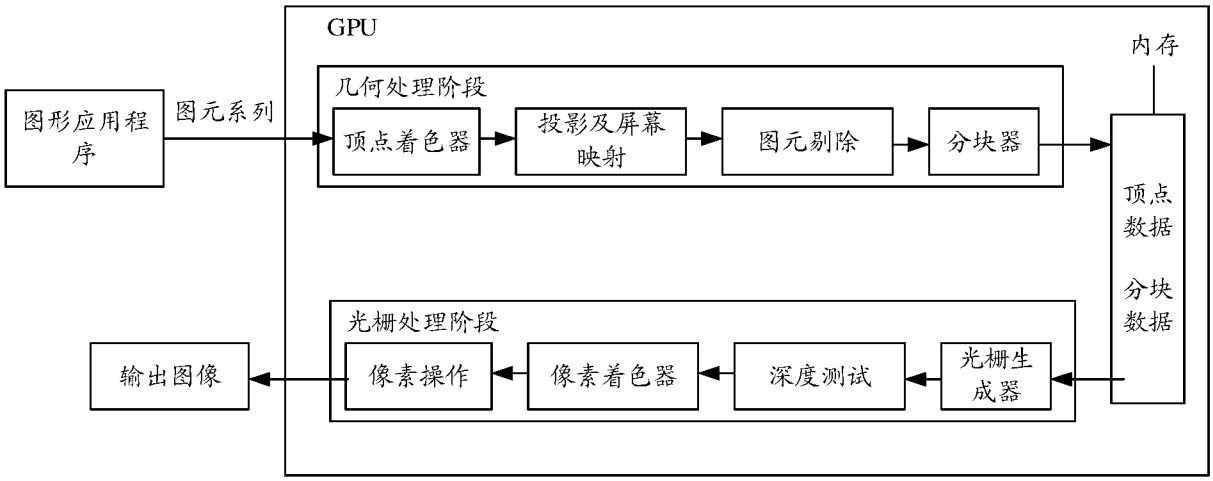
GPU
内存
几何处理阶段
图形应用程序
图元系列
顶点着色器
投影及屏幕映射
图元剔除
分块器
顶点数据
分块数据
光栅处理阶段
输出图像
像素操作
像素着色器
深度测试
光栅生成器

图 3

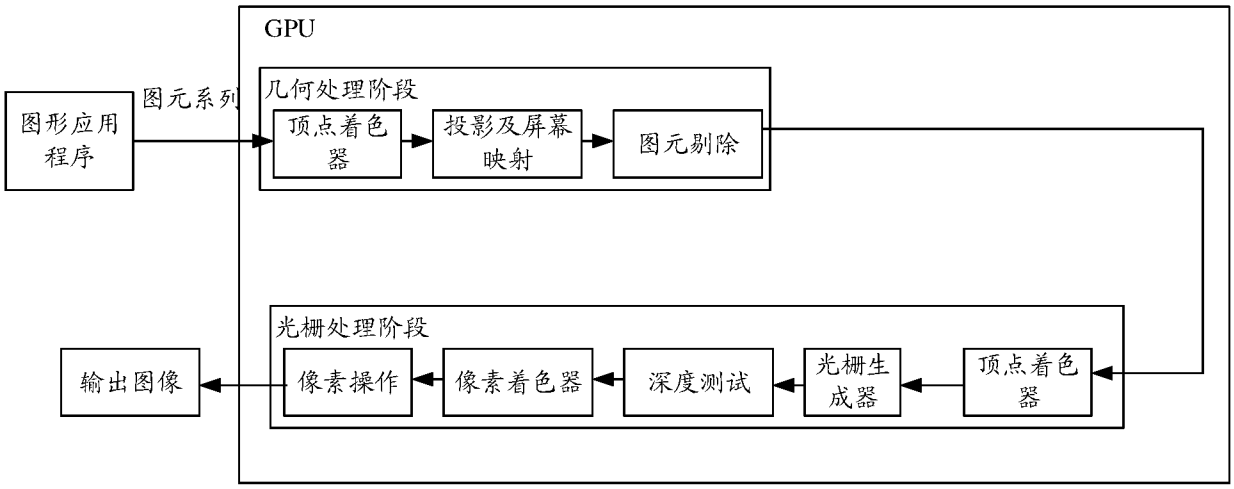
GPU
图形应用程序
图元系列
几何处理阶段
顶点着色器
投影及屏幕映射
图元剔除
光栅处理阶段
输出图像
像素操作
像素着色器
深度测试
光栅生成器
顶点着色器

图 4

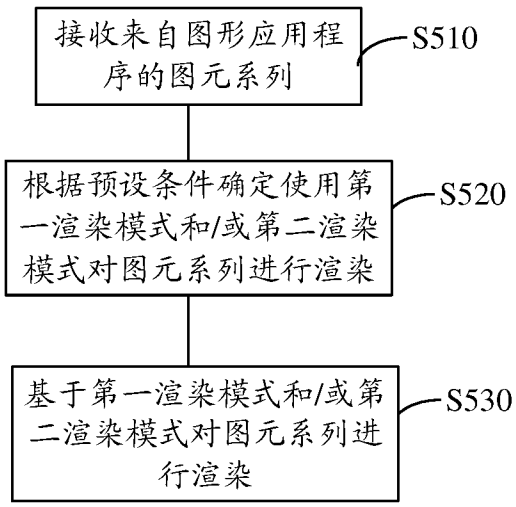
接收来自图形应用程序的图元系列
S510
根据预设条件确定使用第一渲染模式和/或第二渲染模式对图元系列进行渲染
S520
基于第一渲染模式和/或第二渲染模式对图元系列进行渲染
S530

图 5

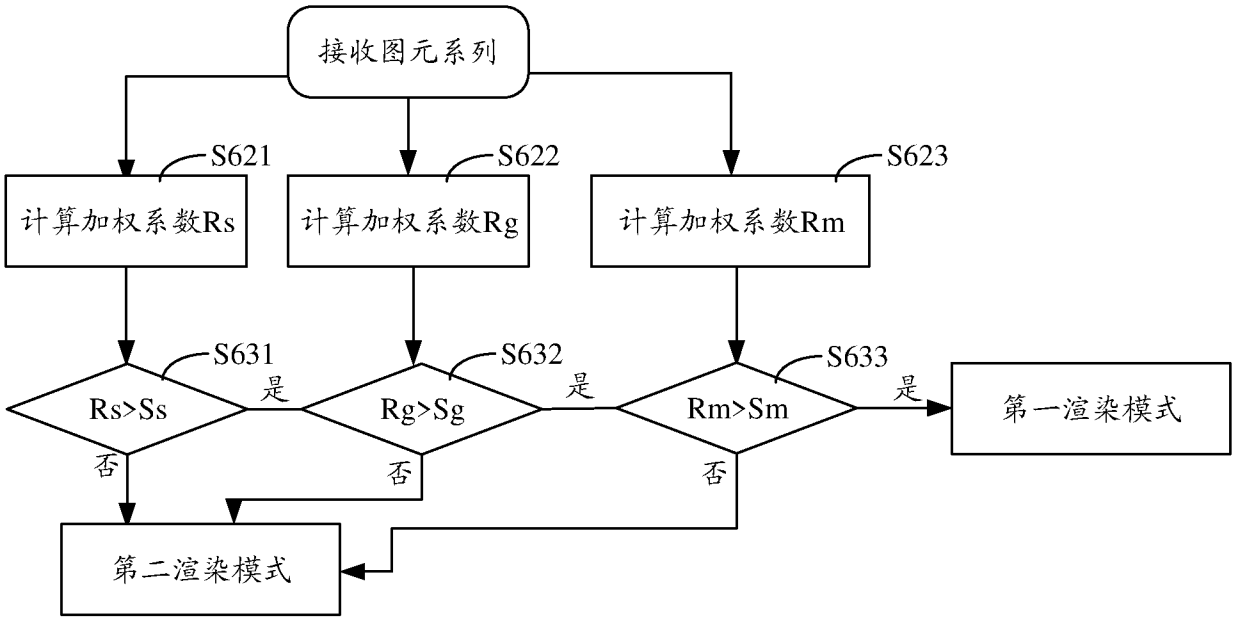
接收图元系列
S621
S622
S623
计算加权系数Rs
计算加权系数Rg
计算加权系数Rm
S631
S632
S633
Rs>Ss
Rg>Sg
Rm>Sm
是
是
是
否
否
否
第一渲染模式
第二渲染模式

图 6

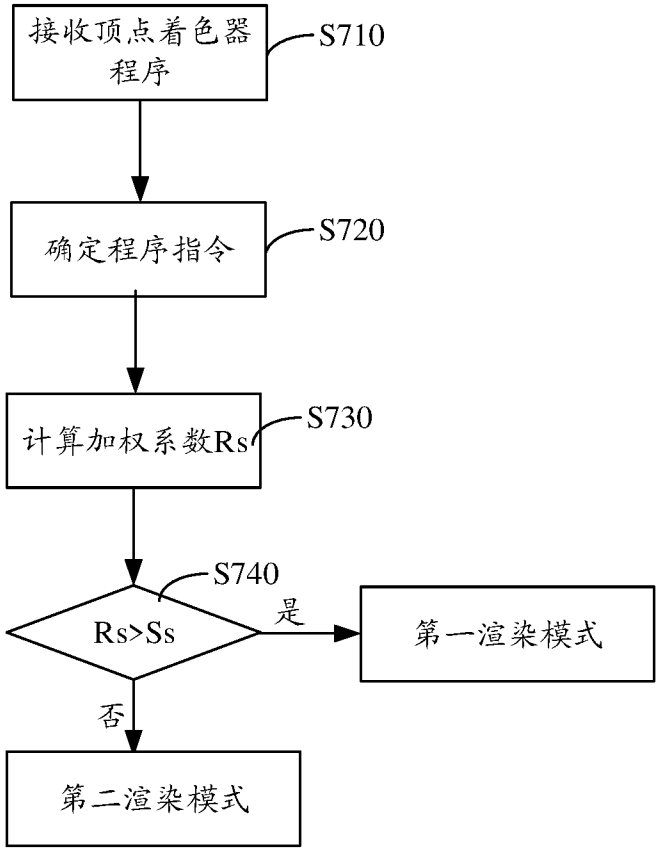
接收顶点着色器程序
S710
确定程序指令
S720
计算加权系数Rs
S730
S740
Rs>Ss
是
第一渲染模式
否
第二渲染模式

图 7

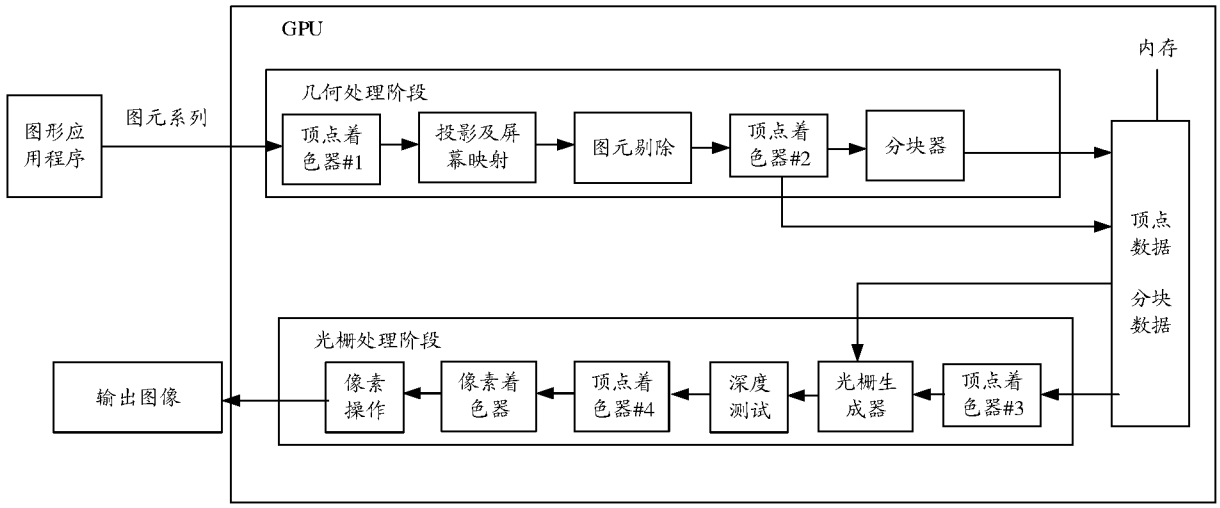
GPU
内存
图形应用程序
图元系列
几何处理阶段
顶点着色器#1
投影及屏幕映射
图元剔除
顶点着色器#2
分块器
顶点数据
分块数据
光栅处理阶段
输出图像
像素操作
像素着色器
顶点着色器#4
深度测试
光栅生成器
顶点着色器#3

图 8

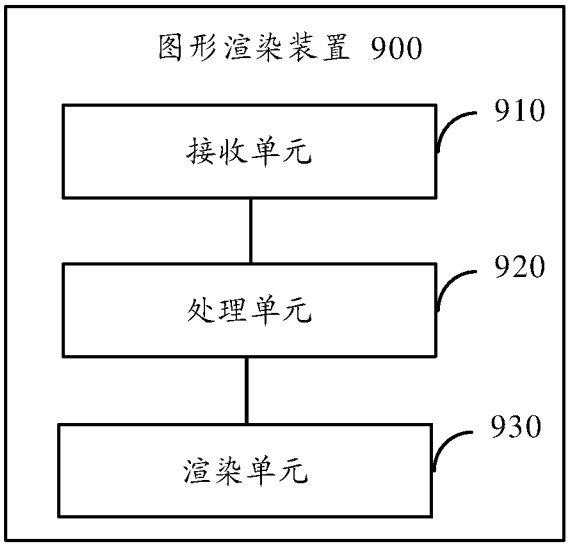
图形渲染装置 900
接收单元
910
处理单元
920
渲染单元
930

图 9

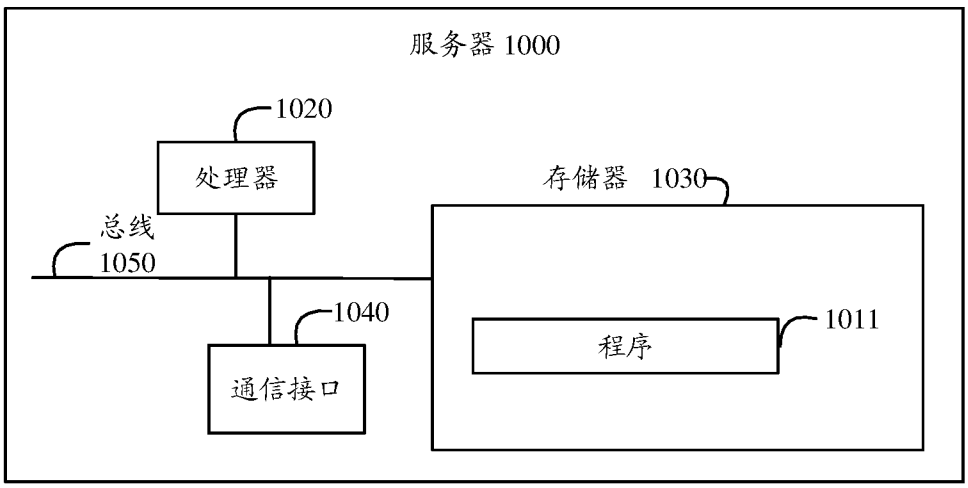
服务器 1000
1020
处理器
存储器 1030
总线
1050
1040
通信接口
程序
1011

图 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2021/101386

A. CLASSIFICATION OF SUBJECT MATTER

G06T 15/00(2011.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT, CNKI, WPI, EPODOC: 图形, 渲染, 模式, 内存, 图形处理器, graphics, render, mode, memory, GPU

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 112348929 A (TENCENT TECHNOLOGY (SHENZHEN) CO., LTD.) 09 February 2021 (2021-02-09) description, paragraphs 76-234	1-18
A	CN 109325899 A (SHANGHAI ZHAOXIN INTEGRATION CIRCUIT CO., LTD.) 12 February 2019 (2019-02-12) entire document	1-18
A	CN 109242756 A (SHANGHAI ZHAOXIN INTEGRATION CIRCUIT CO., LTD.) 18 January 2019 (2019-01-18) entire document	1-18
A	US 7385609 B1 (NVIDIA CORP.) 10 June 2008 (2008-06-10) entire document	1-18
A	US 2020111247 A1 (ARM LIMITED) 09 April 2020 (2020-04-09) entire document	1-18

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 March 2022	**22 March 2022**
Name and mailing address of the ISA/CN **China National Intellectual Property Administration (ISA/CN)** **No. 6, Xitucheng Road, Jimenqiao, Haidian District, Beijing 100088, China** Facsimile No. **(86-10)62019451**	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2021/101386

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 112348929 A	09 February 2021	None	
CN 109325899 A	12 February 2019	US 2020082492 A1	12 March 2020
CN 109242756 A	18 January 2019	US 2020082493 A1	12 March 2020
US 7385609 B1	10 June 2008	None	
US 2020111247 A1	09 April 2020	GB 2580740 A	29 July 2020

国际检索报告

国际申请号
PCT/CN2021/101386

A. 主题的分类

G06T 15/00(2011.01)i

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

G06T

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用））

CNPAT,CNKI,WPI,EPODOC:图形，渲染，模式，内存，图形处理器，graphics, render, mode, memory, GPU

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
A	CN 112348929 A (腾讯科技深圳有限公司) 2021年2月9日 (2021-02-09) 说明书第76-234段	1-18
A	CN 109325899 A (上海兆芯集成电路有限公司) 2019年2月12日 (2019-02-12) 全文	1-18
A	CN 109242756 A (上海兆芯集成电路有限公司) 2019年1月18日 (2019-01-18) 全文	1-18
A	US 7385609 B1 (NVIDIA CORPORATION) 2008年6月10日 (2008-06-10) 全文	1-18
A	US 2020111247 A1 (ARM LIMITED) 2020年4月9日 (2020-04-09) 全文	1-18

☐ 其余文件在C栏的续页中列出。

☑ 见同族专利附件。

* 引用文件的具体类型：

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件（如具体说明的）

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
2022年3月10日	2022年3月22日
ISA/CN的名称和邮寄地址 中国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088 传真号 (86-10)62019451	受权官员 吴黄飞 电话号码 86-(10)-53961430

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2021/101386

检索报告引用的专利文件	公布日 (年/月/日)	同族专利	公布日 (年/月/日)
CN 112348929 A	2021年2月9日	无	
CN 109325899 A	2019年2月12日	US 2020082492 A1	2020年3月12日
CN 109242756 A	2019年1月18日	US 2020082493 A1	2020年3月12日
US 7385609 B1	2008年6月10日	无	
US 2020111247 A1	2020年4月9日	GB 2580740 A	2020年7月29日