

一种基于虚拟场景的渲染方法、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 202111419447.9、申请日为 2021 年 11 月 26 日的中国专利申请提出，并要求该中国专利申请的优先权，该中国专利申请的全部内容在此引入本申请作为参考。

技术领域

本申请涉及计算机应用领域中的渲染技术，尤其涉及一种基于虚拟场景的渲染方法、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

在虚拟场景中，常常存在需要渲染海洋面和水面等液体面、以及与液体面交互的虚拟对象的情况。一般来说，为了实现液体面、以及与液体面交互的虚拟对象的渲染，通常通过图形处理器（Graphics Processing Unit，GPU）解算液体面的波动数据以渲染出高置信度的液体面，并通过回读 GPU 中的波动数据以渲染出与液体面交互的虚拟对象。
15 然而，上述渲染过程中，存在回读 GPU 数据的处理，而回读 GPU 数据存在性能波动问题，影响了渲染性能的稳定性。

发明内容

本申请实施例提供一种基于虚拟场景的渲染方法、装置、电子设备、计算机可读存储介质及计算机程序产品，能够提升渲染性能的稳定性。
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基于虚拟场景的渲染方法，所述方法包括：
图形处理器解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与所述待渲染模型对应的顶点波动数据集合；
中央处理器将所述待渲染模型划分为多个待渲染栅格，并解算每个所述待渲染栅格
25 对应的栅格波动数据，得到与多个所述待渲染栅格对应的栅格波动数据集合；
所述图形处理器基于所述顶点波动数据集合渲染虚拟场景中的待渲染液体面；
所述中央处理器基于所述栅格波动数据集合渲染与所述待渲染液体面交互的虚拟对象。
本申请实施例提供一种基于虚拟场景的渲染装置，所述渲染装置包括：
30 第一解算模块，配置为调用图形处理器执行以下处理：解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与所述待渲染模型对应的顶点波动数据集合；
第二解算模块，配置为调用中央处理器执行以下处理：将所述待渲染模型划分为多个待渲染栅格，并解算每个所述待渲染栅格对应的栅格波动数据，得到与多个所述待渲染栅格对应的栅格波动数据集合；

数据渲染模块，配置为调用所述图形处理器执行以下处理：基于所述顶点波动数据集合渲染虚拟场景中的待渲染液体面，并调用所述中央处理器执行以下处理：基于所述栅格波动数据集合渲染用于与所述待渲染液体面交互的虚拟对象。

本申请实施例提供一种基于液体面进行渲染的电子设备，所述电子设备包括：

5 存储器，用于存储计算机可执行指令；

处理器，包括图形处理器和中央处理器，用于执行所述存储器中存储的计算机可执行指令时，实现本申请实施例提供的基于虚拟场景的渲染方法。

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，所述计算机可执行指令用于被处理器执行时，实现本申请实施例提供的基于虚拟场景的渲染方法。

10 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基于虚拟场景的渲染方法。

本申请实施例至少具有以下有益效果：由于通过 GPU 实现的待渲染液体面的渲染是基于 GPU 解算的数据（称为顶点波动数据集合）实现的，通过 CPU 实现的与待渲染液体面交互的虚拟对象的渲染是基于 CPU 解算的数据（称为栅格波动数据集合）实现的；也就是说，CPU 和 GPU 各自依据自身解算的数据就能实现各自所要渲染的内容，从而能够降低 CPU 和 GPU 交互的资源消耗，因此，能够提升渲染性能的稳定性。

附图说明

- 图 1 是一种示例性的格斯特纳波的波形示意图；
- 20 图 2 是一种示例性的格斯特纳波的渲染示意图；
- 图 3 是本申请实施例提供的基于虚拟场景的渲染系统的架构示意图；
- 图 4 是本申请实施例提供的图 3 中的一种终端的组成结构示意图；
- 图 5 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图一；
- 图 6a 是本申请实施例提供的获取栅格波动数据集合的流程示意图；
- 25 图 6b 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图二；
- 图 6c 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图三；
- 图 6d 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图四；
- 图 6e 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图五；
- 图 6f 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图六；
- 30 图 6g 是本申请实施例提供的渲染虚拟对象的流程示意图；
- 图 7 是本申请实施例提供的一种示例性的液体面的渲染流程图；
- 图 8 是本申请实施例提供的一种示例性的水波模拟模块的功能实现示意图；
- 图 9 是本申请实施例提供的一种示例性的运行在 CPU 的子模块的实现流程示意图；
- 图 10 是本申请实施例提供的一种示例性的运行在 GPU 的子模块的实现流程示意图；
- 35 图 11 是本申请实施例提供的一种示例性的着色模拟模块实现水体着色的示意图；
- 图 12 是本申请实施例提供的一种示例性的着色渲染模块的实现流程示意图；
- 图 13 是本申请实施例提供的一种示例性的水体波动的模拟示意图；
- 图 14 是本申请实施例提供的另一种示例性的水体波动的模拟示意图；
- 图 15 是本申请实施例提供的又一种示例性的水体波动的模拟示意图；
- 40 图 16 是本申请实施例提供的一种示例性的水体波动与虚拟对象的交互示意图；
- 图 17 是本申请实施例提供的一种示例性的水面光照渲染效果示意图；
- 图 18 是本申请实施例提供的另一种示例性的水面光照渲染效果示意图。

具体实施方式

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。

5 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。

10 除非另有定义，本申请实施例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。本申请实施例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。

15 1) 流体力学，是流体力学的一个分支，用于研究作为连续介质的流体在力作用下的运动规律及其与边界的相互作用，还包括流体和其他运动形态的相互作用；其中，流体力学与流体静力学的差别在于流体力学研究运动中的流体；流体力学与流体运动学的差别在于流体力学用于研究作用在流体上的力；另外，流体力学包括液体动力学和气体动力学两大部分，本申请实施例中涉及液体动力学，描述了液体面在液体动力学涉及到的力的作用下产生的波纹。

2) 波纹，是指风力等环境力对水面产生的流体力学效应，包括格斯特纳波 (Gerstner Wave, Gerstner 波)，又称为摆动波 (Trochoidal Wave)，Gerstner 波的本质是液体表面周期重力波的欧拉方程的精确解。其中，重力波 (Gravity Waves) 是指在流体力学中，25 在液体介质内或两种介质界面间 (比如，大气与海洋间) 的一种波；当一小团液体离开液面 (两种介质界面类型) 或者在液体中到了一个液体密度不同的区域 (液体介质内类型)，透过重力作用，这团液体会以波动形式在平衡态之间摆动，就形成了重力波。欧拉方程，即运动微分方程，属于无黏性流体力学中的基本方程，是指对无黏性流体微团应用牛顿第二定律得到的运动微分方程；在流体力学中，欧拉方程是一组支配无黏30 性流体运动的方程。在本申请实施例中，所涉及的各种波动数据用于渲染液体面的波纹。

3) CPU 超线程，是指利用特殊字符的硬件指令，把多个逻辑内核模拟成物理芯片，让单个处理器能使用线程级并行计算，从而兼容多线程并行计算，进而兼容多线程操作系统和软件，提高运行性能。在本申请实施例中，CPU 所执行的波动数据的解算可以是采用 CPU 超线程实现的。

35 4) 物理引擎，用于通过为刚性物体赋予真实的物理属性的方式来计算运动、旋转和碰撞反映；在本申请实施例中，与液体面交互的虚拟对象的运动信息可以由物理引擎确定的。

5) 着色器，用来替代固定渲染管线实现图像渲染的可编辑程序，具有可编辑性，从而能够实现各种各样的图像渲染效果而不用受显卡的固定渲染管线限制。在本申请实40 施例中，着色器包括计算着色器 (Compute Shader) 和片元着色器 (又称为像素着色器，Pixel Shader)。其中，计算着色器用于利用着色器的资源来处理一些与渲染不相关的计算工作，在本申请实施例中，计算着色器用于对大量 (大于第一数量阈值) 的顶点数据进行并行处理；片元着色器，用来进行漫反射和/或镜面反射的光照计算。

6) 渲染管线, 用于定义 GPU 上的计算方法和流程, 比如, 几何管线。

7) 光栅化引擎, 用于实现光栅化; 其中, 光栅化是把顶点数据转换为片元的过程, 片元中的每一个元素可以对应于帧缓冲区中的一个像素, 从而光栅化即为将几何图元变为二维图像的过程。

5 8) 虚拟场景, 是指利用设备输出的区别于现实世界的场景, 通过裸眼或设备的辅助能够形成对虚拟场景的视觉感知; 例如, 通过显示屏幕输出的二维影像, 通过立体投影、虚拟现实和增强现实技术等立体显示技术来输出的三维影像; 此外, 还可以通过各种可能的硬件形成听觉感知、触觉感知、嗅觉感知和运动感知等各种模拟现实世界的感知。

10 还需要说明的是, 应用程序在终端设备上运行时显示 (或提供) 的虚拟场景, 可以是对真实世界的仿真环境, 也可以是半仿真半虚构的虚拟环境, 还可以是纯虚构的虚拟环境。虚拟场景可以是二维虚拟场景、2.5 维虚拟场景或者三维虚拟场景中的任意一种, 本申请实施例对虚拟场景的维度不加以限定。例如, 虚拟场景可以包括天空、陆地、海洋等, 该陆地可以包括沙漠、城市和建筑等环境元素, 虚拟对象可以在用户或人工智能
15 (Artificial Intelligence, AI) 的控制下, 在该虚拟场景中进行移动或执行其他操作 (如攻击操作)。

9) 虚拟对象, 虚拟场景中可以进行交互的各种人和物的形象, 或在虚拟场景中的可活动对象。该可活动对象可以是虚拟人物、虚拟动物和动漫人物等, 比如, 在虚拟场景中显示的人物、动物、植物、油桶、墙壁、石块和建筑等。该虚拟对象可以是该虚拟
20 场景中的一个虚拟的用于代表用户的虚拟形象。虚拟场景中可以包括多个虚拟对象, 每个虚拟对象在虚拟场景中具有自身的形状和体积, 占据虚拟场景中的一部分空间。在本申请实施例中, 涉及的虚拟对象为与待渲染液体面交互的虚拟对象。

一般来说, 为了实现液体面的波纹的渲染, 通常是通过渲染 Gerstner 波来实现, 参见图 1, 图 1 是一种示例性的格斯特纳波的波形示意图; 如图 1 所示, 渲染的为在风向
25 1-1 的作用下的格斯特纳波 1-2。由于 Gerstner 波的渲染过程中, 占用的是可以缩放的运算资源, 渲染性能和实时性较好, 因此得到了广泛应用。然而, 由于 Gerstner 波是通过调度大量 (大于第二数量阈值) 线程实现的渲染, 因此通常是通过 GPU 渲染 Gerstner 波。参见图 2, 图 2 是一种示例性的格斯特纳波的渲染示意图; 如图 2 所示, 基于区域
30 2-1 中设置的参数, 通过渲染 Gerstner 波而渲染出的水面 2-2。

然而, 在通过 GPU 渲染 Gerstner 波、快速傅立叶变换 (Fast Fourier Transform, FFT) 波和波粒子模拟的波等波纹来实现液体面的渲染时, 如果还需要渲染与液体面交互的虚拟对象的运动效果, 则需要通过 GPU 总线 (GPU BUS) 传输 GPU 片上的波动数据至内存以供 CPU 进行处理, 这是因为用于渲染与液体面交互的虚拟对象的运动效果的功能
35 模块 (比如, 浮力结算模块, 物理引擎等) 是通过 CPU 实现的; 如此是一种回读处理, 每帧的数据回读将导致毫秒级 (比如, 5 毫秒) 的性能波动; 所以, 通过 GPU 渲染 Gerstner 波等波纹来实现液体面的渲染、以及与液体面交互的虚拟对象影响了渲染的稳定性。

另外, 当通过 GPU 渲染 Gerstner 波等波纹来实现液体面的渲染的过程中, CPU 往往处于空载状态, 以等待 GPU 的处理, 如此, 造成了 CPU 资源的浪费。

基于此, 本申请实施例提供一种基于虚拟场景的渲染方法、装置、电子设备、计算机可读存储介质及计算机程序产品, 能够提升渲染稳定性和资源利用率。下面说明本申请
40 实施例提供的基于液体面进行渲染的电子设备 (以下简称为渲染设备) 的示例性应用。本申请实施例提供的渲染设备可以实施为智能手机、智能手表、笔记本电脑、平板电脑、台式计算机、智能电视、机顶盒、智能车载设备、便携式音乐播放器、个人数字助理、专用消息设备、便携式游戏设备和智能音箱等各种类型的终端, 也可以实施为服务器。

下面，将说明渲染设备实施为终端时的示例性应用。

参见图 3，图 3 是本申请实施例提供的基于虚拟场景的渲染系统的架构示意图；如图 3 所示，为支撑一个基于虚拟场景的渲染应用，在基于虚拟场景的渲染系统 100 中，终端 400（称为渲染设备，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，以及服务器 200 用于向终端 400 提供计算服务。另外，该基于虚拟场景的渲染系统 100 中还包括数据库 500，用于向服务器 200 提供数据支持；并且，图 3 中示出的为数据库 500 独立于服务器 200 的一种情况，此外，数据库 500 还可以集成在服务器 200 中，本申请实施例对此不作限定。

终端 400，用于通过图形处理器，解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与待渲染模型对应的顶点波动数据集合；通过中央处理器将待渲染模型划分为多个待渲染栅格，并解算每个待渲染栅格对应的栅格波动数据，得到与多个待渲染栅格对应的栅格波动数据集合；在虚拟场景中，由图形处理器基于顶点波动数据集合渲染待渲染液体面，并由中央处理器基于栅格波动数据集合渲染与待渲染液体面交互的虚拟对象（参见终端 400-1 中示出的水面、以及水面上漂浮的箱体，参见终端 400-2 中示出的海面、以及海面上漂浮的船只）。

在一些实施例中，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、内容分发网络（CDN，Content Delivery Network）、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 可以是智能手机、智能手表、笔记本电脑、平板电脑、台式计算机、智能电视、机顶盒、智能车载设备、便携式音乐播放器、个人数字助理、专用消息设备、便携式游戏设备和智能音箱等，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，本申请实施例中不作限制。

需要说明的是，当服务器 200 为云端游戏服务器时，可以由服务器 200 渲染待渲染液体面、以及与所述待渲染液体面交互的虚拟对象，并将表征渲染出的待渲染液体面、以及与所述待渲染液体面交互的虚拟对象的音视频信息发送至终端 400 进行播放。

参见图 4，图 4 是本申请实施例提供的图 3 中的一种终端的组成结构示意图，图 4 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 4 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。

5 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;

10 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 到达其他电子设备, 示例性的网络接口 420 包括: 蓝牙、无线相容性认证 (Wi-Fi)、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);

15 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基于虚拟场景的渲染装置可以采用软件方式实现,图 4 示出了存储在存储器 450 中的渲染装置 455,其可以是程序和插件等形式的软件,包括以下软件模块:第一解算模块 4551、第二解算模块 4552、数据渲染模块 4553、着色渲染模块 4554 和参数获取模块 4555, 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。其中,模块为由 CPU、GPU 运行的软件、程序等软件形式。

20 在一些实施例中,本申请实施例提供的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基于虚拟场景的渲染方法,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。

30 在一些实施例中,终端或服务器可以通过运行计算机程序来实现本申请实施例提供的基于虚拟场景的渲染方法。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,如视频播放 APP 或者游戏应用 APP (包括本地游戏应用 APP 和云游戏应用 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。

下面,将结合本申请实施例提供的渲染设备的示例性应用和实施,说明本申请实施例提供的基于虚拟场景的渲染方法;其中,该基于虚拟场景的渲染方法由渲染设备执行,并且,该渲染设备包括中央处理器和图形处理器。另外,该渲染设备可以为同时包括中央处理器和图形处理器的一个设备,还可以是包括中央处理器的设备和包括图形处理器的设备的综合,又可以是两者的结合,等等,本申请实施例对此不作限定;比如,手机或台式机,内部同时设置有中央处理器和图形处理器;又比如,台式机通过扩充界面 (ExpressCard 接口) 或者板卡 (mini-PCIe 接口) 外接一个或多个图形处理器,同时使用内置和外接的图形处理器;再比如,台式机中设置中央处理器,不设置图形处理器,

通过 ExpressCard 接口或者 mini-PCIe 接口外接一个或多个图形处理器；等等。

参见图 5，图 5 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图一，将结合图 5 示出的步骤进行说明。

5 S501、图形处理器解算待渲染模型的每个待渲染顶点对应的顶点波动数据，得到与待渲染模型对应的顶点波动数据集合。

在本申请实施例中，渲染设备在渲染待渲染液体面、以及待渲染液体面交互的虚拟对象时，是通过 CPU 和 GPU 的并行处理实现的。在 CPU 和 GPU 的并行处理过程中，GPU 用于解算出待渲染模型对应的细粒度的波动数据，这里称为顶点波动数据集合，该顶点波动数据集合用于渲染待渲染液体面的波纹。

10 这里，渲染设备在渲染待渲染液体面、以及待渲染液体面交互的虚拟对象时，调用 GPU，以使 GPU 以待渲染顶点为单位对待渲染模型的波动数据进行解算；当渲染设备解算出每个待渲染顶点对应的波动数据（称为顶点波动数据）时，组合待渲染模型中的每个待渲染顶点对应的顶点波动数据，也就获得了与待渲染模型对应的顶点波动数据集合。

15 需要说明的是，待渲染液体面是一种待渲染液体的表面，比如，水体的表面（海洋的表面，湖泊的表面），其他溶液的表面，等等；其中，待渲染模型为用于渲染待渲染液体面的模型，可以是三维结构模型，也可以是二维结构模型，等等，本申请实施例对此不作限定。与待渲染液体面交互的虚拟对象可以是待渲染液体面所承载的虚拟对象，比如，船只，漂浮的人或物，等等。另外，待渲染模型是由网格组成的，而网格是由多边形构成的，多边形的顶点即为待渲染顶点，从而，待渲染模型包括多个待渲染顶点。
20 此外，顶点波动数据是待渲染顶点的波动数据；顶点波动数据集合是待渲染液面的细粒度（称为待渲染顶点粒度）的波动数据，且顶点波动数据集合是由与多个待渲染顶点一一对应的多个顶点波动数据构成的集合。

25 S502、中央处理器将待渲染模型划分为多个待渲染栅格，并解算每个待渲染栅格对应的栅格波动数据，得到与多个待渲染栅格对应的栅格波动数据集合。

在本申请实施例中，在渲染设备的 CPU 和 GPU 并行渲染的过程中，CPU 用于解算出待渲染模型对应的粗粒度（称为待渲染栅格粒度）的波动数据，这里称为栅格波动数据集合；该栅格波动数据集合用于确定与待渲染液体交互的虚拟对象的运动信息，进而基于运动信息渲染出与待渲染液体面交互的虚拟对象。

30 这里，渲染设备在渲染待渲染液体面、以及待渲染液体面交互的虚拟对象时，调用 GPU，以使 GPU 以待渲染顶点为单位对待渲染模型的波动数据进行解算，还调用 CPU，以使 CPU 将待渲染模型划分为多个待渲染栅格，进而以待渲染栅格为单位对待渲染模型的波动数据进行解算；当渲染设备解算出每个待渲染栅格对应的波动数据（称为栅格波动数据）时，组合待渲染模型中的每个待渲染栅格对应的栅格波动数据，也就获得了与待渲染模型对应的栅格波动数据集合。
35

需要说明的是，待渲染栅格是通过对待渲染模型进行区域划分获得的，是一个指定大小的区域，并且待渲染栅格比待渲染顶点的粒度粗；比如，待渲染栅格包括至少两个待渲染顶点。另外，栅格波动数据是待渲染栅格的波动数据；栅格波动数据集合是待渲染模型的粗粒度（待渲染栅格粒度）的波动数据，且栅格波动数据集合是由与多个待渲染栅格一一对应的多个栅格波动数据构成的集合。从而，GPU 的解算粒度比 CPU 的解算粒度细。
40

还需要说明的是，S501 和 S502 在执行顺序上不分先后；并且 S501 和 S502 是并行的。也就是说，S501 的执行是在 S502 的执行过程中执行的，即是在中央处理器获取栅格波动数据集合的过程中执行的；S502 的执行是在 S501 的执行过程中执行的，即是在

通过图形处理器获取顶点波动数据集合的过程中执行的。从而，S502 中渲染设备通过中央处理器将待渲染模型划分为多个待渲染栅格，并解算每个待渲染栅格对应的栅格波动数据，得到与多个待渲染栅格对应的栅格波动数据集合，包括：渲染设备在通过图形处理器获取顶点波动数据集合的过程中，通过中央处理器将待渲染模型划分为多个待渲染栅格，并解算每个待渲染栅格对应的栅格波动数据，得到与多个待渲染栅格对应的栅格波动数据集合。

S503、图形处理器基于顶点波动数据集合渲染虚拟场景中的待渲染液体面，以及，中央处理器基于栅格波动数据集合渲染与待渲染液体面交互的虚拟对象。

在本申请实施例中，渲染设备的 GPU 获得了用于渲染待渲染液体面的波纹的顶点波动数据集合之后，由 GPU 基于顶点波动数据集合渲染待渲染液体面的波纹，也就实现了待渲染液体面在虚拟场景中的渲染。以及，渲染设备的 CPU 获得了用于渲染与待渲染液体面交互的虚拟对象的姿态的栅格波动数据集合之后，由 CPU 基于栅格波动数据集合渲染与待渲染液体面交互的虚拟对象的姿态，也就实现了与待渲染液体面交互的虚拟对象在虚拟场景中的渲染。

需要说明的是，待渲染液体面的渲染是由 GPU 执行的，与待渲染液体面交互的虚拟对象的渲染是由 CPU 执行的。

可以理解的是，在 GPU 解算待渲染液体面的波动数据（称为顶点波动数据集合）的过程中，同时通过 CPU 对待渲染液体面的波动数据进行粗粒度的解算；从而，在 GPU 实现待渲染液体面的精准渲染的同时，CPU 能够基于 CPU 解算出的粗粒度波动数据（称为栅格波动数据集合）渲染与待渲染液体面交互的虚拟对象；如此，由于通过 GPU 实现的待渲染液体面的渲染是基于 GPU 解算的数据实现的，通过 CPU 实现的与待渲染液体面交互的虚拟对象的渲染是基于 CPU 解算的数据实现的；也就是说，CPU 和 GPU 各自依据自身解算的数据就能实现各自所要渲染的内容，从而能够降低 CPU 和 GPU 交互的资源消耗，因此，能够提升渲染性能的稳定性。另外，在 GPU 解算用于渲染待渲染液体面的波纹的波动数据的同时，CPU 解算用于渲染与待渲染液体面交互的虚拟对象的姿态的波动数据，从而能够实现待渲染液体面、以及待渲染液体面交互的虚拟对象的并行渲染，提升了资源利用率。

参见图 6a，图 6a 是本申请实施例提供的获取栅格波动数据集合的流程示意图；如图 6a 所示，在本申请实施例中，S502 中渲染设备中的 CPU 解算每个待渲染栅格对应的栅格波动数据，得到与多个待渲染栅格对应的栅格波动数据集合，包括 S5021 至 S5023，下面对各步骤分别进行说明。

S5021、为每个待渲染栅格创建第一波动数据解算任务，得到与多个待渲染栅格对应的多个第一波动数据解算任务。

在本申请实施例中，CPU 可以离散地、也可以并行地对各个待渲染栅格对应的各个栅格波动数据进行解算。这里，当 CPU 为多核心时，能够实现超线程，此时，CPU 可以通过并行处理获得栅格波动数据集合。

需要说明的是，CPU 针对每个待渲染栅格创建一个用于解算对应的栅格波动数据的任务，也就创建出了第一波动数据解算任务；从而，针对多个待渲染栅格，也就创建出了多个第一波动数据解算任务。其中，第一波动数据解算任务用于解算与每个待渲染栅格对应的栅格波动数据，多个待渲染栅格与多个第一波动数据解算任务一一对应。

S5022、将多个第一波动数据解算任务划分为至少一个波动数据解算线程。

在本申请实施例中，CPU 将构建的多个第一波动数据解算任务分配给至少一个线程（称为至少一个波动数据解算线程）；这里，一个线程对应一个任务队列，波动数据解算线程为一个线程，并且该波动数据解算线程对应于由多个第一波动数据解算任务中的

至少一个第一波动数据解算任务构成的任务队列。其中，多个第一波动数据解算任务中的第一波动数据解算任务的数量，大于等于至少一个波动数据解算线程中的波动数据解算线程的数量。

S5023、执行至少一个波动数据解算线程，得到栅格波动数据集合。

5 在本申请实施例中，当至少一个波动数据解算线程为多个波动数据解算线程时，CPU 可以并行执行多个波动数据解算线程，也可以串行执行多个波动数据解算线程，也可以是并行和串行的组合，等等，本申请实施例对此不作限定。

需要说明的是，CPU 执行多个波动数据解算线程的过程，即为执行每个第一波动数据解算任务解算出每个待渲染栅格对应的栅格波动数据的过程；从而，当 CPU 完成
10 多个波动数据解算线程的执行时，也就获得了待渲染液体面中的每个待渲染栅格对应的栅格波动数据，从而也就获得了由待渲染模型中的每个待渲染栅格对应的栅格波动数据组成的栅格波动数据集合。

可以理解的是，通过利用 CPU 的多核心和超线程处理，实现了在 CPU 解算待渲染模型的粗粒度波动数据的过程，进而提升了渲染液体面的稳定性和资源利用率。

15 参见图 6b，图 6b 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图二；如图 6b 所示，在本申请实施例中，S501 中渲染设备的 GPU 解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与待渲染模型对应的顶点波动数据集合，包括 S5011 和 S5012，下面对各步骤分别进行说明。

S5011、遍历待渲染模型对应的待渲染顶点集合。

20 需要说明的是，待渲染模型包括的多个待渲染顶点即为待渲染顶点集合。这里，GPU 对待渲染顶点集合中的多个待渲染顶点进行遍历，以逐个解算出待渲染顶点的顶点波动数据。

S5012、解算每个遍历到的待渲染顶点对应的顶点波动数据，得到与待渲染模型对应的顶点波动数据集合。

25 需要说明的是，GPU 针对每个遍历到的待渲染顶点进行波动数据的解算，也就解算出了待渲染模型中的每个待渲染顶点对应的顶点波动数据，从而，通过组合待渲染模型中的每个待渲染顶点对应的顶点波动数据，也就获得了顶点波动数据集合。

在本申请实施例中，S5011 和 S5012 可通过 GPU 中的几何管线（即为片元着色器和顶点着色器中的至少一种）实现。

30 参见图 6c，图 6c 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图三；如图 6c 所示，在本申请实施例中，S501 中渲染设备的 GPU 解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与待渲染模型对应的顶点波动数据集合，包括 S5013 和 S5015，下面对各步骤分别进行说明。

35 S5013、基于划分单位，对待渲染液体面对应的待渲染顶点集合进行划分，得到多个待渲染网格。

在本申请实施例中，GPU 可以通过将待渲染模型划分为多个待渲染网格，其中，划分单位包括指定数量的待渲染顶点，从而每个待渲染网格包括指定数量的待渲染顶点。其中，待渲染网格与待渲染栅格的不同之处在于，待渲染网格是基于待渲染顶点的数量划分的，待渲染栅格是基于区域划分的；以及，指定数量为线程束中线程的数量，比如，
40 32*32，64*64，等等。

S5014、为每个待渲染网格中的每个待渲染顶点创建第二波动数据解算任务，得到与待渲染网格对应的第二波动数据解算任务组，从而得到与多个待渲染网格对应的多个第二波动数据解算任务组。

在本申请实施例中，GPU 针对每个待渲染顶点创建一个用于解算对应的顶点波动数

据的任务，也就创建出了第二波动数据解算任务；从而，针对待渲染网格包括的指定数量的待渲染顶点，也就创建出了指定数量的第二波动数据解算任务；这里，将指定数量的第二波动数据解算任务组成一个第二波动数据解算任务组。从而，针对多个待渲染网格，也就创建出了多个第二波动数据解算任务组。其中，第二波动数据解算任务用于解算与每个待渲染顶点对应的顶点波动数据，多个待渲染网格与多个第二波动数据解算任务组一一对应。

S5015、执行多个第二波动数据解算任务组，得到与待渲染液体面对应的顶点波动数据集合。

在本申请实施例中，GPU 可以并行执行多个第二波动数据解算任务组，也可以串行执行多个第二波动数据解算任务组，也可以是并行和串行的组合，等等，本申请实施例对此不作限定。这里，GPU 可以将每个第二波动数据解算任务组对应于一个线程束，通过执行一个线程束来实现每个第二波动数据解算任务组的执行。

需要说明的是，GPU 执行多个第二波动数据解算任务组的过程，即为执行每个第二波动数据解算任务组解算出每个待渲染顶点对应的顶点波动数据的过程；从而，当 GPU 完成多个第二波动数据解算任务组的执行时，也就获得了待渲染模型中的每个待渲染顶点对应的顶点波动数据，从而也就获得了由待渲染模型中的每个待渲染顶点对应的顶点波动数据组成的顶点波动数据集合。

在本申请实施例中，S5013 和 S5015 可通过 GPU 中的计算着色器实现。另外，渲染设备的 GPU 可以在无法通过 S5013 和 S5015 获得顶点波动数据集合的时候，通过 S5011 和 S5012 获得顶点波动数据集合。

可以理解的是，通过利用 GPU 的线程束处理，将线程束对应在了有一定规模（指定数量的待渲染顶点）的待渲染网格上，实现了合批计算，进而实现了轻量级（线程数量小于指定线程数量）的线程调度，能够提升解算待渲染液体面的细粒度的波动数据效率。

参见图 6d，图 6d 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图四；如图 6d 所示，在本申请实施例中，S501 之后还包括 S504；也就是说，渲染设备中的 GPU 解算待渲染液体面的每个待渲染顶点对应的顶点波动数据，得到与待渲染液体面对应的顶点波动数据集合之后，该基于虚拟场景的渲染方法还包括 S504，下面对该步骤进行说明。

S504、通过图形处理器，确定顶点波动数据集合对应的着色渲染数据。

需要说明的是，渲染设备除了渲染待渲染液体面的波纹，还渲染待渲染液体面的着色效果。从而，渲染设备的 GPU 还用于确定顶点波动数据集合对应的着色渲染数据，其中，着色渲染数据用于渲染待渲染液体面的着色效果。

相应地，在本申请实施例中，S503 中 GPU 基于顶点波动数据集合渲染待渲染液体面，包括：GPU 结合着色渲染数据和顶点波动数据集合，渲染虚拟场景中的待渲染液体面。也就是说，GPU 基于着色渲染数据渲染待渲染液体面的着色效果，并基于顶点波动数据集合渲染待渲染液体面的波纹，也就完成了待渲染液体面的渲染。

可以理解的是，GPU 通过获取着色渲染数据，渲染出了光在液体中传播的过程，故能够提升液体波纹颜色渲染的置信度；并且，由于是通过 GPU 和 CPU 的并行处理实现的待渲染液体面的波纹的渲染、以及有待渲染液体面交互的虚拟对象的姿态的渲染，能够在性能稳定的情况下提升待渲染液体面渲染的置信度。

继续参见图 6d，在本申请实施例中，S504 之前还包括 S505 至 S507；也就是说，渲染设备通过图形处理器，确定顶点波动数据集合对应的着色渲染数据之前，该基于虚拟场景的渲染方法还包括 S505 至 S507，下面对各步骤分别进行说明。

S505、CPU 采集光照环境数据。

需要说明的是，渲染模块通过 CPU 进行与着色相关的数据的采集，也就采集到了光照环境数据。这里，光照环境数据包括场景环境数据、光照强度、光照颜色、光照角度和光照行为数据和的至少一种；其中，场景环境数据比如为虚拟场景的天空环境数据或室内环境数据等；光照行为数据比如为虚拟场景中虚拟对象的状态等。

5 S506、CPU 将光照环境数据转换为光渲染参数。

需要说明的是，CPU 对采集到的光照环境数据进行预处理和预计算，以将光照环境数据转换为用于实现光照渲染的数据，也就获得了光渲染参数。

S507、CPU 将光渲染参数发送至图形处理器。

10 在本申请实施例中，CPU 将转换成的光渲染参数通过参数上传者上传至 GPU，也就完成了向 GPU 发送光渲染参数的处理。

相应地，在本申请实施例中，S504 中 GPU 确定顶点波动数据集合对应的着色渲染数据，包括：GPU 基于光渲染参数，确定顶点波动数据集合对应的着色渲染数据。也就是说，当 CPU 将光渲染参数发送至 GPU 时，GPU 也就接收到了光渲染参数，GPU 基于接收到的光渲染参数确定用于渲染待渲染液体面的着色的数据，也就获得了着色渲染数据。

15 继续参见图 6d，在本申请实施例中，S504 可通过 S5041 和 S5042 实现；也就是说，GPU 确定顶点波动数据集合对应的着色渲染数据，包括 S5041 和 S5042，下面对各步骤分别进行说明。

S5041、遍历顶点波动数据集合。

20 在本申请实施例中，GPU 对解算到的顶点波动数据集合以待渲染顶点对应的顶点波动数据为单位进行遍历。

S5042、对每个遍历到的顶点波动数据进行光照处理，得到与顶点波动数据集合对应的着色渲染数据。

25 在本申请实施例中，GPU 对每个遍历到的顶点波动数据进行光照处理时，可以是基于接收到的 CPU 发送的光渲染参数实现的；这里，当 GPU 对顶点波动数据集合中的每个顶点波动数据均完成了光照处理之后，也就获得了与顶点波动数据集合对应的着色渲染数据。这里，光照处理包括表面下散射处理、次表面散射处理和表面反射处理中的至少一种；其中，表面下光线散射处理是指待渲染液体面下的光线散射处理，表面反射处理是指待渲染液体面上的光线反射处理。

30 可以理解的是，渲染设备通过表面下散射处理、次表面散射处理和表面反射处理中的至少一种对每个待渲染顶点的顶点波动数据进行光照处理，实现了从能量守恒和几何光学渲染待渲染液体的表面的波纹颜色，提升了待渲染液体面的逼真度，从而提升了待渲染液体面的渲染效果。

35 参见图 6e，图 6e 是本申请实施例提供的基于虚拟场景的渲染方法的流程图五；如图 6e 所示，在本申请实施例中，S501 之前还包括 S508 至 S510；也就是说，渲染设备的 GPU 解算待渲染液体面的每个待渲染顶点对应的顶点波动数据之前，该液体面的渲染方法还包括 S508 至 S510，下面对各步骤分别进行说明。

S508、CPU 获取波动环境数据。

40 需要说明的是，波动环境数据是指影响待渲染液体面的波纹的数据，包括风向、风力、浪高、波长和波动行为数据中的至少一种；其中，波动行为数据为虚拟对象在虚拟场景中的与待渲染液体面的波动关联的行为。

S509、CPU 将波动环境数据转换为波动环境参数。

需要说明的是，CPU 对采集到的波动环境数据进行预处理和预计算，以将波动环境数据转换为用于实现波纹渲染的数据，也就获得了波动环境参数。

S510、CPU 将波动环境参数发送至图形处理器。

在本申请实施例中，CPU 将转换成的波动环境参数通过参数上传者上传至 GPU，也就完成了向 GPU 发送波动环境参数的处理。这里，当 CPU 将波动环境参数发送至 GPU 时，GPU 也就接收到了波动环境参数。

5 在本申请实施例中，CPU 响应于虚拟场景中待渲染液体面的第 i 帧渲染请求，通过 CPU 执行 S508 至 S510，其中，i 为单调递增的正整数变量。也就是说，本申请实施例提供的基于虚拟场景的渲染方法是逐帧进行待渲染液体面、以及有待渲染液体面交互的虚拟对象的渲染的，能够将实时获得的渲染参数反馈至渲染结果上。

相应地，在本申请实施例中，S501 中渲染设备的 GPU 解算待渲染模型的每个待渲染顶点对应的顶点波动数据，包括：GPU 基于波动环境参数，解算待渲染模型的每个待渲染顶点对应的顶点波动数据。以及，S502 中渲染设备的 CPU 解算每个待渲染栅格对应的栅格波动数据，包括：CPU 基于波动环境参数，解算每个待渲染栅格对应的栅格波动数据。

可以理解的是，CPU 在待渲染液体面、以及有待渲染液体面交互的虚拟对象的渲染过程中，实时响应于每帧的渲染请求，渲染虚拟场景中的待渲染液体面、以及有待渲染液体面交互的虚拟对象，降低了渲染时延，并提升了渲染性能。

参见图 6f，图 6f 是本申请实施例提供的基于虚拟场景的渲染方法的流程示意图六；如图 6f 所示，在本申请实施例中，S502 中渲染设备的 CPU 将待渲染模型划分为多个待渲染栅格，包括 S5024 和 S5025，下面对各步骤分别进行说明。

20 S5024、基于线程处理规模和待渲染液体面确定栅格划分单位。

需要说明的是，线程处理规模为中央处理器所处理的最大线程数量，栅格划分单位与线程处理规模负相关、且有待渲染液体面的规模正相关；这里，CPU 所确定栅格划分单位为每个待渲染栅格的规模。

S5025、基于栅格划分单位，将待渲染模型划分为多个待渲染栅格。

25 需要说明的是，CPU 划分出来的多个待渲染栅格中的待渲染栅格的数量与线程处理规模正相关。

可以理解的是，渲染设备中的 CPU 通过基于自身的线程处理规模，确定用于划分待渲染模型的栅格划分单位，能够最大限度地利用 CPU 的处理资源，提升栅格波动数据集合的获取效率，进而提升渲染效率。

30 参见图 6g，图 6g 是本申请实施例提供的渲染虚拟对象的流程示意图；如图 6g 所示，在本申请实施例中，S503 中渲染设备的 CPU 基于栅格波动数据集合渲染有待渲染液体面交互的虚拟对象，包括 S5031 至 S5034，下面对各步骤分别进行说明。

S5031、基于栅格波动数据集合，确定浮力参数。

35 需要说明的是，CPU 确定浮力参数包括排水量等参数，用于确定待渲染液体面的浮力。

S5032、基于浮力参数确定待渲染液体面对应的浮力信息。

S5033、调用物理引擎执行以下处理：基于浮力信息，确定有待渲染液体面交互的虚拟对象的运动信息。

S5034、基于运动信息渲染虚拟对象。

40 在本申请实施例中，CPU 通过自身解算出的栅格波动数据集合，确定待渲染液体面与虚拟对象交互的作用力（比如，浮力信息等），进而基于该作用力渲染有待渲染液体面交互的虚拟对象的状态或姿态。其中，物理引擎由中央处理器调用。

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。该示例性应用描述了水体渲染领域中水体表面（水面，称为待渲染液体面）的渲染方法，比如，运行

在用户端或云端服务器上的游戏应用中的三维（Three Dimensions，3D）渲染模块实现的水面的渲染。

需要说明的是，水面的渲染包括水体波纹模拟和波纹光照着色；其中，水体波纹模拟是指模拟水面受风向、风力、浪高和波长等环境力的影响而产生流体力学效应所出现的波纹，这里，通过 CPU 和 GPU 同时解算水面因环境力的推动产生的波纹所对应的水体波动数据。水体作为一种复杂介质，这里通过波纹光照着色体现光学着色特性，其中，波纹光照着色是指模拟光在水体传播的过程，以提升渲染出的水体波纹颜色的置信度。

参见图 7，图 7 是本申请实施例提供的一种示例性的液体面的渲染流程图；如图 7 所示，首先是从虚拟对象的状态（称为虚拟对象在虚拟环境中的行为，比如，虚拟船只的行驶）7-1 和所在虚拟环境 7-2 中收集环境参数 7-3；然后水面渲染系统 7-4 中的水波模拟模块 7-41 和着色模拟模块 7-42 基于环境参数 7-3 共同渲染出水面 7-5。其中，水面渲染系统 7-4 即为实现水面的渲染的功能模块，水波模拟模块 7-41 用于通过模拟水面波动的情况实现水面波纹的模拟，着色模拟模块 7-42 用于通过模拟水面波纹在光照环境下的着色效果实现波纹光照着色。另外，水体波纹模拟除了受风向、风力、浪高和波长等环境力的影响，虚拟对象在虚拟环境中的行为也可以影响水体波纹模拟；这是因为虚拟对象在虚拟环境中的行为改变了环境力，进而影响水面的渲染结果，因此，虚拟对象在虚拟环境中的行为可以对水面的模拟产生影响。

在本申请实施例中，图 7 中的水波模拟模块 7-41 包括运行在 GPU 的子模块和运行在 CPU 上的子模块。参见图 8，图 8 是本申请实施例提供的一种示例性的水波模拟模块的功能实现示意图；如图 8 所示，水波模拟模块 7-41 中的运行在 CPU 的子模块 8-1 一方面用于实现水面区域的栅格划分，得到栅格划分结果 8-11（称为多个待渲染栅格）；另一方面用于以栅格级别进行波动数据的解算，得到粗粒度波动结算结果 8-12（称为栅格波动数据集合）。水波模拟模块 7-41 中的运行在 GPU 上的子模块 8-2 一方面用于读取水面顶点 8-21（称为待渲染顶点集合）；另一方面用于遍历水面顶点 8-21，逐顶点（称为待渲染顶点）进行波动数据的解算，得到细粒度波动解算结果 8-22（称为顶点波动数据集合）。这里，粗粒度波动结算结果 8-12 用于通过运行在 CPU 的浮力模块 8-3 计算排水量等参数（称为浮力参数），以获得浮力 8-4（称为浮力信息），进而，CPU 上运行的物理引擎 8-5 基于浮力 8-4 确定与水面交互的虚拟对象的运动信息 8-6；细粒度波动解算结果 8-22 用于通过 GPU 的水面渲染模块渲染水体波动效果，即渲染出水面波纹 8-7。

可以理解的是，图 7 中的水波模拟模块 7-41 确保了水面渲染的波几何效果以及物理解算时的排水量等参数的正确性。通过水波模拟模块 7-41，可以按照风向、风力、浪高以及波长等参数对水面波纹进行真实模拟，并且水面波纹能够同时正确地反馈到渲染动画以及交互物体（称为与待渲染液体面交互的虚拟对象）的物理受力和运动姿态上。

下面对图 8 中的运行在 CPU 的子模块 8-1 的实现流程进行说明。参见图 9，图 9 是本申请实施例提供的一种示例性的运行在 CPU 的子模块的实现流程示意图；如图 9 所示，该示例性的运行在 CPU 的子模块的实现流程包括 S901 至 S907，下面对各步骤分别进行说明。

S901、终端（称为渲染设备）启动水面渲染系统。

在本申请实施例中，水面渲染系统 7-4 启动，以开始水面的渲染的流程。

S902、CPU 判断水面渲染的帧迭代是否结束。如果是则执行 S907。如果否则执行 S903。

需要说明的是，水面、以及水面交互的虚拟对象是逐帧渲染的。

S903、CPU 对水面进行栅格划分。

这里，栅格划分为体素化的过程。

S904、CPU 为每个栅格创建一个格斯特纳波的解算任务（称为第一波动数据解算任务）。

5 S905、CPU 将多个栅格对应的多个解算任务划分至不同的解算线程中进行执行，得到粗粒波动解算数据。

S906、CPU 将粗粒波动解算数据发送至浮力模块。之后执行 S902。

需要说明的是，浮力模块是运行在 CPU 上的用于计算浮力的模块；这里，浮力模块基于粗粒波动解算数据计算浮力，并将浮力发送至物理引擎，以使物理引擎基于浮力确定水面上的虚拟对象的运动数据。其中，物理引擎也是运行在 CPU 上的用于确定交互物体的运动信息的模块。

10 S907、终端关闭水面渲染系统。

需要说明的是，终端通过关闭水面渲染系统，结束水面渲染流程。

下面对图 8 中的运行在 GPU 的子模块 8-2 的实现流程进行说明。基于图 9，参见图 10，图 10 是本申请实施例提供的一种示例性的运行在 GPU 的子模块的实现流程示意图；如图 10 所示，该示例性的运行在 GPU 的子模块的实现流程包括 S901、S902 和 S907，以及 S1001 至 S1008，下面对各步骤分别进行说明。

S1001、GPU 接收 CPU 上传的波动环境参数（包括环境力和虚拟对象在虚拟环境中的行为）。

这里，当执行 S902 确定的结果是否为时，执行 S1001。

20 S1002、GPU 判断计算着色器是否可用。如果是则执行 S1003。如果否则执行 S1006。

S1003、GPU 通过计算着色器读取水面的网格（称为待渲染网格）。

S1004、GPU 将线程束固定在网格上。

S1005、GPU 通过线程束合批解算一定规模（称为指定数量的待渲染顶点）的网格上的各顶点的波动数据。之后执行 S1008。

25 S1006、GPU 通过几何管线遍历每个顶点。

S1007、GPU 解算每个顶点的波动数据。

S1008、GPU 将所有顶点对应的波动数据通过几何管线提交至水面渲染模块渲染水面波纹。之后执行 S902。其中，水面渲染模块为水波模拟模块 7-41 中的运行在 GPU 上的模块。

30 需要说明的是，S903 至 S906 的执行过程，与 S1001 至 S1008 的执行过程，两者是并行处理的。

在本申请实施例中，图 7 中的着色模拟模块 7-42 包括 GPU 功能模块（比如 GPU 程序）、参数上传器和参数收集器；其中，GPU 功能模块运行在 GPU 上，参数上传器和参数收集器运行在 CPU 上。参见图 11，图 11 是本申请实施例提供的一种示例性的着色模拟模块实现水体着色的示意图；如图 11 所示，着色模拟模块 7-42 中的参数收集器 11-1，用于收集渲染和仿真的着色数据 11-2（称为光照环境数据）；着色模拟模块 7-42 中的参数上传器 11-3，用于预处理和预计算收集的着色数据 11-2，得到着色参数 11-4（称为光渲染参数），还用于将着色参数 11-4 上传至着色模拟模块 7-42 中的着色渲染模块 11-5。着色模拟模块 7-42 中的着色渲染模块 11-5（称为 GPU 功能模块）运行在 GPU 上，用于基于着色参数 11-4 对几何管线读取到细粒度波动解算数据进行光照处理（包括水下散射处理、水次表面散射处理和水面反射处理），得到着色渲染数据 11-6；还用于基于着色渲染数据 11-6 渲染水面波纹颜色 11-7。

下面对图 11 中的运行在 GPU 的着色渲染模块 11-5 的实现流程进行说明。基于图 9，参见图 12，图 12 是本申请实施例提供的一种示例性的着色渲染模块的实现流程示意

图;如图 12 所示,该示例性的运行在 GPU 的着色渲染模块的实现流程包括 S901、S902 和 S907,以及 S1201 至 S1205,下面对各步骤分别进行说明。

S1201、读取 CPU 上传的着色参数。

这里,当执行 S902 确定的结果是否为否时,执行 S1201。

5 S1202、基于着色参数和细粒度波动解算数据进行水面反射处理。

S1203、对水面反射处理结果进行水面散射处理。

S1204、对水面散射处理结果进行次表面散射处理。

S1205、基于次表面散射处理结果渲染水面波纹。之后执行 S902。

需要说明的是,渲染等处理采用 GPU 上的光栅化引擎和几何管线实现,而环境数据由 CPU 收集,再将收集的环境参数经由参数上传器上传到 GPU 上,以供 GPU 上的着色器程序进行读取和访问。

下面通过图 13 至图 16 对图 7 中的水波模拟模块 7-41 所实现的水体波动进行说明。参见图 13,图 13 是本申请实施例提供的一种示例性的水体波动的模拟示意图;如图 13 所示,水面 13-1 是风向 13-21 为向前(-6)、风力 13-22 为 1 级、以及水面波动强度系数 13-23 为 0.238 时的渲染结果,其中,风向 13-21、风力 13-22 和水面波动强度系数 13-23 为图 7 中的环境参数 7-3。

参见图 14,图 14 是本申请实施例提供的另一种示例性的水体波动的模拟示意图;如图 14 所示,水面 14-1 是风向 14-21 为向右(80)、风力 14-22 为 5 级、以及水面波动强度系数 14-23 为 0.456 时的渲染结果。

20 需要说明的是,图 13 和图 14 是环境参数驱动水体波动的表现,也就是说,可以根据虚拟环境中的风力、风向和水面波动强度系数产生不同的波动表现。

参见图 15,图 15 是本申请实施例提供的又一种示例性的水体波动的模拟示意图;如图 15 所示,水面 15-1 是在风况改变的情况下模拟出的拍岸波。

可以理解的是,通过图 7 中的水波模拟模块 7-41,能够响应环境的快速改变,产生多样的水面波纹。

参见图 16,图 16 是本申请实施例提供的一种示例性的水体波动与虚拟对象的交互示意图;如图 16 所示,示出了水面 16-1 上,受到侧方来波而倾斜的船只 16-2。

可以理解的是,通过本申请实施例提供的基于 GPU 和 CPU 并行解算波动数据的水面渲染方法,能够降低物理模拟的延迟,并且能够实时响应到物理引擎中,降低渲染延迟,提升浮力效果的逼真度。

下面通过图 17 和图 18 对图 7 中的着色模拟模块 7-42 所实现的水体波动在光照环境下的着色效果进行说明。

参见图 17,图 17 是本申请实施例提供的一种示例性的水面光照渲染效果示意图;如图 17 所示,水面 17-1 是晴朗昼午环境下的光照射色模拟结果。

35 参见图 18,图 18 是本申请实施例提供的另一种示例性的水面光照渲染效果示意图;如图 18 所示,水面 18-1 是多风浪的黄昏环境下的光照射色模拟结果。

可以理解的是,通过本申请实施例提供的着色模拟模块,能够基于场景(天空环境)、主光照强度、颜色、角度和用户设置等变化,渲染出各种不同的光照环境下的渲染效果。

40 还可以理解的是,通过本申请实施例提供的水面波纹和波纹颜色的模拟方法,能够结合几何和光学特性提升水面渲染的置信度,提升水面的渲染效果。另外,通过在 CPU 和 GPU 并行进行波动数据的解算,能够快速获得与水面交互的虚拟对象的姿态,提升与水面交互的虚拟对象的渲染效率。

下面继续说明本申请实施例提供的基于虚拟场景的渲染装置 455 的实施为软件模块

的示例性结构，在一些实施例中，如图4所示，存储在存储器450的渲染装置455中的软件模块可以包括：

第一解算模块4551，配置为调用图形处理器执行以下处理：解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与所述待渲染模型对应的顶点波动数据集；

5 第二解算模块4552，配置为调用中央处理器执行以下处理：将所述待渲染模型划分为多个待渲染栅格，并解算每个所述待渲染栅格对应的栅格波动数据，得到与多个所述待渲染栅格对应的栅格波动数据集；

数据渲染模块4553，配置为调用所述图形处理器执行以下处理：基于所述顶点波动数据集渲染虚拟场景中的待渲染液体面，并调用所述中央处理器执行以下处理：基于
10 所述栅格波动数据集渲染与所述待渲染液体面交互的虚拟对象。

在本申请实施例中，所述第二解算模块4552，还配置为调用所述中央处理器执行以下处理：对每个所述待渲染栅格创建第一波动数据解算任务，得到与多个所述待渲染栅格对应的多个所述第一波动数据解算任务，其中，所述第一波动数据解算任务用于解算所述待渲染栅格的所述栅格波动数据；将多个所述第一波动数据解算任务划分为至少一个波动数据解算线程；执行至少一个所述波动数据解算线程，得到所述栅格波动数据集
15 合。

在本申请实施例中，所述第一解算模块4551，还配置为调用所述图形处理器执行以下处理：遍历所述待渲染模型对应的待渲染顶点集合；解算每个遍历到的所述待渲染顶点对应的所述顶点波动数据，得到与所述待渲染模型对应的所述顶点波动数据集。

20 在本申请实施例中，所述第一解算模块4551，还配置为调用图形处理器执行以下处理：基于划分单位，对所述待渲染模型对应的待渲染顶点集合进行划分，得到多个待渲染网格，其中，所述划分单位包括指定数量的所述待渲染顶点；为每个所述待渲染网格中的每个所述待渲染顶点创建第二波动数据解算任务，得到与所述待渲染网格对应的第二波动数据解算任务组，其中，多个所述待渲染网格对应多个第二波动数据解算任务组，
25 所述第二波动数据解算任务用于解算与每个所述待渲染顶点对应的所述顶点波动数据；执行多个所述第二波动数据解算任务组，得到与所述待渲染模型对应的所述顶点波动数据集合。

在本申请实施例中，所述渲染装置455还包括着色渲染模块4554，配置为调用所述图形处理器执行以下处理：确定所述顶点波动数据集对应的着色渲染数据。

30 在本申请实施例中，所述数据渲染模块4553，还配置为调用所述图形处理器执行以下处理：结合所述着色渲染数据和所述顶点波动数据集，渲染所述待渲染液体面。

在本申请实施例中，所述着色渲染模块4554，还配置为调用所述中央处理器执行以下处理：采集光照环境数据，其中，所述光照环境数据包括场景环境数据、光照强度、光照颜色、光照角度和光照行为数据中的至少一种；将所述光照环境数据转换为光渲染参数；将所述光渲染参数发送至所述图形处理器。
35

在本申请实施例中，所述着色渲染模块4554，还配置为调用所述图形处理器执行以下处理：基于所述光渲染参数，确定所述顶点波动数据集对应的所述着色渲染数据。

在本申请实施例中，所述着色渲染模块4554，还配置为调用所述图形处理器执行以下处理：遍历所述顶点波动数据集；对每个遍历到的所述顶点波动数据进行光照处理，得到与所述顶点波动数据集对应的所述着色渲染数据，其中，所述光照处理包括表面
40 下散射处理、次表面散射处理和表面反射处理中的至少一种。

在本申请实施例中，所述渲染装置455还包括参数获取模块4555，配置为调用所述中央处理器执行以下处理：响应于所述虚拟场景中所述待渲染液体面的第*i*帧渲染请求，通过所述中央处理器执行以下处理，其中，*i*为单调递增的正整数变量：获取波动环境

数据, 其中, 所述波动环境数据包括风向、风力、浪高、波长和波动行为数据中的至少一种; 将所述波动环境数据转换为波动环境参数; 将所述波动环境参数发送至所述图形处理器。

在本申请实施例中, 所述第一解算模块 4551, 还配置为调用所述图形处理器执行以下处理: 基于所述波动环境参数, 解算所述待渲染模型中的每个所述待渲染顶点对应的所述顶点波动数据。

在本申请实施例中, 所述第二解算模块 4552, 还配置为调用所述中央处理器执行以下处理: 基于所述波动环境参数, 解算每个所述待渲染栅格对应的所述栅格波动数据。

在本申请实施例中, 所述第二解算模块 4552, 还配置为调用所述中央处理器执行以下处理: 基于线程处理规模和所述待渲染模型确定栅格划分单位, 其中, 所述线程处理规模为所述中央处理器所处理的最大线程数量; 基于所述栅格划分单位, 将所述待渲染模型划分为多个所述待渲染栅格。

在本申请实施例中, 所述数据渲染模块 4553, 还配置为调用所述中央处理器执行以下处理: 基于所述栅格波动数据集合, 确定浮力参数; 基于所述浮力参数确定所述待渲染液体面对应的浮力信息; 调用物理引擎执行以下处理: 基于所述浮力信息, 确定与所述待渲染液体面交互的所述虚拟对象的运动信息, 其中, 所述物理引擎由所述中央处理器调用; 基于所述运动信息渲染所述虚拟对象。

在本申请实施例中, 所述第二解算模块 4552, 还配置为在调用所述图形处理器获取所述顶点波动数据集合的过程中, 调用所述中央处理器执行以下处理: 将所述待渲染模型划分为多个所述待渲染栅格, 并解算每个所述待渲染栅格对应的所述栅格波动数据, 得到与多个所述待渲染栅格对应的所述栅格波动数据集合。

本申请实施例提供了一种计算机程序产品或计算机程序, 该计算机程序产品或计算机程序包括计算机可执行指令, 该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 (称为渲染设备) 的处理器 (包括图形处理器和中央处理器) 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基于虚拟场景的渲染方法。

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基于虚拟场景的渲染方法, 例如图 5 示出的基于虚拟场景的渲染方法。

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 (此时, 这一个电子设备即为渲染设备), 或者在位于一个地点的多个电子设备上执行 (此时, 位于一个地点的多个电子设备即为渲染设备), 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（此时，分布在多个地点且通过通信网络互连的多个电子设备即为渲染设备）。

可以理解的是，在本申请实施例中，涉及到的相关的数据，当本申请实施例运用到具体产品或技术中时，需要获得用户许可或者同意，且相关数据的收集、使用和处理需要遵守相关国家和地区的相关法律法规和标准。

综上所述，本申请实施例在 GPU 解算待渲染液体面的波动数据的过程中，同时通过 CPU 对待渲染液体面的波动数据进行粗粒度的解算；从而，在 GPU 实现待渲染液体面的高置信度渲染的同时，CPU 能够基于 CPU 解算出的粗粒度波动数据渲染与待渲染液体面交互的虚拟对象；如此，由于通过 GPU 实现的待渲染液体面的渲染是基于 GPU 解算的数据实现的，通过 CPU 实现的与待渲染液体面交互的虚拟对象的渲染是基于 CPU 解算的数据实现的；也就是说，CPU 和 GPU 仅依据自身解算的数据就能实现渲染，从而能够提升渲染性能的稳定性。另外，在 GPU 解算用于渲染待渲染液体面的波纹的波动数据的同时，CPU 解算用于渲染与待渲染液体面交互的虚拟对象的姿态的波动数据，从而能够实现待渲染液体面、以及与待渲染液体面交互的虚拟对象的并行渲染，提升了资源利用率。

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、一种基于虚拟场景的渲染方法，所述方法包括：
- 图形处理器解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与所述待渲染模型对应的顶点波动数据集合；
- 5 中央处理器将所述待渲染模型划分为多个待渲染栅格，并解算每个所述待渲染栅格对应的栅格波动数据，得到与多个所述待渲染栅格对应的栅格波动数据集合；
- 所述图形处理器基于所述顶点波动数据集合渲染虚拟场景中的待渲染液体面；
- 所述中央处理器基于所述栅格波动数据集合渲染用于与所述待渲染液体面交互的虚拟对象。
- 10 2、根据权利要求1所述的方法，其中，所述解算每个所述待渲染栅格对应的栅格波动数据，得到与多个所述待渲染栅格对应的栅格波动数据集合，包括：
- 为每个所述待渲染栅格创建第一波动数据解算任务，得到与多个所述待渲染栅格对应的多个所述第一波动数据解算任务，其中，所述第一波动数据解算任务用于解算所述待渲染栅格的所述栅格波动数据；
- 15 将多个所述第一波动数据解算任务划分为至少一个波动数据解算线程；
- 执行至少一个所述波动数据解算线程，得到所述栅格波动数据集合。
- 3、根据权利要求1或2所述的方法，其中，所述解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与所述待渲染模型对应的顶点波动数据集合，包括：
- 遍历所述待渲染模型对应的待渲染顶点集合；
- 20 解算每个遍历到的所述待渲染顶点对应的所述顶点波动数据，得到与所述待渲染模型对应的所述顶点波动数据集合。
- 4、根据权利要求1或2所述的方法，其中，所述解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与所述待渲染模型对应的顶点波动数据集合，包括：
- 基于划分单位，对所述待渲染模型对应的待渲染顶点集合进行划分，得到多个待渲染网格，其中，所述划分单位包括指定数量的所述待渲染顶点；
- 25 为每个所述待渲染网格中的每个所述待渲染顶点创建第二波动数据解算任务，得到与所述待渲染网格对应的第二波动数据解算任务组，其中，多个所述待渲染网格对应多个第二波动数据解算任务组，所述第二波动数据解算任务组用于解算与每个所述待渲染顶点对应的所述顶点波动数据；
- 30 执行多个所述第二波动数据解算任务组，得到与所述待渲染模型对应的所述顶点波动数据集合。
- 5、根据权利要求1或2所述的方法，其中，所述解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与所述待渲染模型对应的顶点波动数据集合之后，所述方法还包括：
- 35 所述图形处理器确定所述顶点波动数据集合对应的着色渲染数据；
- 所述基于所述顶点波动数据集合渲染虚拟场景中的待渲染液体面，包括：
- 结合所述着色渲染数据和所述顶点波动数据集合，渲染所述虚拟场景中的所述待渲染液体面。
- 6、根据权利要求5所述的方法，其中，所述确定所述顶点波动数据集合对应的着色渲染数据之前，所述方法还包括：
- 40 所述中央处理器执行以下处理：
- 采集光照环境数据，其中，所述光照环境数据包括场景环境数据、光照强度、光照颜色、光照角度和光照行为数据中的至少一种；

- 将所述光照环境数据转换为光渲染参数；
将所述光渲染参数发送至所述图形处理器；
所述确定所述顶点波动数据集合对应的着色渲染数据，包括：
基于所述光渲染参数，确定所述顶点波动数据集合对应的所述着色渲染数据。
- 5 7、根据权利要求5所述的方法，其中，所述确定所述顶点波动数据集合对应的着色渲染数据，包括：
遍历所述顶点波动数据集合；
对每个遍历到的所述顶点波动数据进行光照处理，得到与所述顶点波动数据集合对应的所述着色渲染数据，其中，所述光照处理包括表面下散射处理、次表面散射处理和表面反射处理中的至少一种。
- 10 8、根据权利要求1或2所述的方法，其中，所述解算待渲染模型中的每个待渲染顶点对应的顶点波动数据之前，所述方法还包括：
所述中央处理器响应于所述虚拟场景中所述待渲染液体面的第i帧渲染请求，执行以下处理，其中，i为单调递增的正整数变量：
15 获取波动环境数据，其中，所述波动环境数据包括风向、风力、浪高、波长和波动行为数据中的至少一种；
将所述波动环境数据转换为波动环境参数；
将所述波动环境参数发送至所述图形处理器；
所述解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，包括：
20 基于所述波动环境参数，解算所述待渲染模型中的每个所述待渲染顶点对应的所述顶点波动数据；
所述解算每个所述待渲染栅格对应的栅格波动数据，包括：
基于所述波动环境参数，解算每个所述待渲染栅格对应的所述栅格波动数据。
- 25 9、根据权利要求1或2所述的方法，其中，所述将所述待渲染模型划分为多个待渲染栅格，包括：
基于线程处理规模和所述待渲染模型确定栅格划分单位，其中，所述线程处理规模为所述中央处理器所处理的最大线程数量；
基于所述栅格划分单位，将所述待渲染模型划分为多个所述待渲染栅格。
- 30 10、根据权利要求1或2所述的方法，其中，所述基于所述栅格波动数据集合渲染用于与所述待渲染液体面交互的虚拟对象，包括：
基于所述栅格波动数据集合，确定浮力参数；
基于所述浮力参数确定所述待渲染液体面对应的浮力信息；
调用物理引擎执行以下处理：基于所述浮力信息，确定用于与所述待渲染液体面交互的所述虚拟对象的运动信息；
35 基于所述运动信息渲染所述虚拟对象。
- 11、根据权利要求1或2所述的方法，其中，所述中央处理器将所述待渲染模型划分为多个待渲染栅格，并解算每个所述待渲染栅格对应的栅格波动数据，得到与多个所述待渲染栅格对应的栅格波动数据集合，包括：
在所述图形处理器获取所述顶点波动数据集合的过程中，所述中央处理器将所述待渲染模型划分为多个所述待渲染栅格，并解算每个所述待渲染栅格对应的所述栅格波动数据，得到与多个所述待渲染栅格对应的所述栅格波动数据集合。
- 40 12、一种基于虚拟场景的渲染装置，所述渲染装置包括：
第一解算模块，配置为调用图形处理器执行以下处理：解算待渲染模型中的每个待渲染顶点对应的顶点波动数据，得到与所述待渲染模型对应的顶点波动数据集合；

第二解算模块，配置为调用中央处理器执行以下处理：将所述待渲染模型划分为多个待渲染栅格，并解算每个所述待渲染栅格对应的栅格波动数据，得到与多个所述待渲染栅格对应的栅格波动数据集合；

- 5 数据渲染模块，配置为调用所述图形处理器执行以下处理：基于所述顶点波动数据集合渲染虚拟场景中的待渲染液体面，并调用所述中央处理器执行以下处理：基于所述栅格波动数据集合渲染用于与所述待渲染液体面交互的虚拟对象。

13、一种基于液体面进行渲染的电子设备，所述电子设备包括：

存储器，用于存储计算机可执行指令；

- 10 处理器，包括图形处理器和中央处理器，用于执行所述存储器中存储的计算机可执行指令时，实现权利要求 1 至 11 任一项所述的基于虚拟场景的渲染方法。

14、一种计算机可读存储介质，存储有计算机可执行指令，所述计算机可执行指令用于被处理器执行时，实现权利要求 1 至 11 任一项所述的基于虚拟场景的渲染方法。

- 15 15、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 11 任一项所述的基于虚拟场景的渲染方法。