导入文件的方法、装置、电子设备及计算机可读介质

技术领域

本申请涉及数据处理技术领域，尤其涉及一种导入文件的方法、装置，电子设备及计算机可读介质。

背景技术

随着计算机技术的发展，互联网和计算机使得用户可以通过例如在创建、使用虚拟形象的方式进行交互。这样的方式，允许用户基于其设计、构想，来创建物理环境中实际不存在的“对象”，通过这样的虚拟形象可以更为直观、多元以及趣味性地为用户之间提供更为新颖的交互体验。

而为了帮助用户实现虚拟形象的创建目的，内容创作平台、引擎应运而生。Unity引擎是一种可以提供实时三维互动内容创作和运营的平台，其可以帮助包括游戏开发、美术、建筑、汽车设计、影视在内的所有创作者，将创意变成现实。Unity引擎可以提供一整套完善的软件解决方案，以用于创作、运营和变现任何实时互动的二维和三维内容，支持平台包括手机、平板电脑、个人计算机（PC）、游戏主机、增强现实和虚拟现实设备。由此，如何提升Unity引擎的能力、适用范围，以及保障Unity引擎执行稳定、执行安全，是值得关注和迫切需求的。

发明内容

本申请的多个方面提供一种导入文件的方法、装置，电子设备及计算机可读存储介质，在接收到针对Unity引擎的待加载文件时，利用动画剪辑文件作为实际导入、被执行的文件，以保障替代待加载文件中的内容可以实际、可靠地被执行。这样的方式，不仅可以提升Unity引擎的文件格式适用范围，还能保障Unity引擎的执行稳定性。

本申请的一方面，提供一种导入文件的方法，包括：获取针对Unity引擎的待加载文件；如果待加载文件的文件类型为目标类型，解封装待加载文件，得到解封装结果，其中，解封装结果中至少包括待加载文件的视频数据；基于解封装结果，生成动画剪辑文件；以及向Unity引擎导入动画剪辑文件，以使得Unity引擎通过执行动画剪辑文件来替代执行待加载文件。

本申请的另一方面，提供一种用于导入文件的装置，包括：获取模块，被配置为获取针对Unity引擎的待加载文件；解封装模块，被配置为如果待加载文件的文件类型为目标类型，解封装待加载文件，得到解封装结果，其中，解封装结果中至少包括待加载文件的视频数据；生成模块，被配置为基于解封装结果，生成动画剪辑文件；以及导入模块，被配置为向Unity引擎导入动画剪辑文件，以使得Unity引擎通过执行动画剪辑文件来替代执行待加载文件。

本申请的另一方面，提供一种电子设备，该电子设备包括：至少一个处理器；以及与至少一个处理器通信连接的存储器；其中，存储器存储有可被至少一个处理器执行的指令，指令被至少一个处理器执行，以使至少一个处理器能够执行如上提供的导入文件的方法。

本申请的另一方面，提供一种计算机可读存储介质，其上存储有计算机程序指令，计算机程序指令可被处理器执行以实现如上提供的导入文件的方法。

本申请实施例提供的方案中，能够获取针对Unity引擎的待加载文件；如果待加载文件的文件类型为目标类型，解封装待加载文件，得到解封装结果，其中，解封装结果中至少包括待加载文件的视频数据；基于解封装结果，生成动画剪辑文件；以及向Unity引擎导入动画剪辑文件，以使得Unity引擎通过执行动画剪辑文件来替代执行待加载文件。由此，使得在接收到针对Unity引擎的待加载文件时，利用动画剪辑文件作为实际导入、被执行的文件，以保障替代待加载文件中的内容可以实际、可靠地被执行。这样的方式，不仅可以提升Unity引擎的文件格式适用范围，还能保障Unity引擎的执行稳定性。

附图说明

为了更清楚地说明本申请实施例中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作一简单地介绍，显而易见地，下面描述中的附图是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动性的前提下，还可以根据这些附图获得其它的附图。

通过阅读参照以下附图所作的对非限制性实施例所作的详细描述，本申请的其它特征、目的和优点将会变得更明显：

图1为本申请一实施例提供的导入文件的过程的示意图；

图2为本申请另一实施例提供的生成动画剪辑文件的过程的示意图；

图3为本申请一实施例提供的用于导入文件的装置的结构示意图；

图4为适用于实现本申请实施例中的方案的一种电子设备的结构示意图。

附图中相同或相似的附图标记代表相同或相似的部件。

具体实施方式

为使本申请实施例的目的、技术方案和优点更加清楚，下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例，都属于本申请保护的范围。

在本申请一个典型的配置中，终端、服务网络的设备均包括一个或多个处理器 (CPU)、输入/输出接口、网络接口和内存。

内存可能包括计算机可读介质中的非永久性存储器，随机存取存储器 (RAM) 和/或非易失性内存等形式，如只读存储器 (ROM) 或闪存(flash RAM)。内存是计算机可读介质的示例。

计算机可读介质包括永久性和非永久性、可移动和非可移动媒体，可以由任何方法或技术来实现信息存储。信息可以是计算机程序指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存 (PRAM)、静态随机存取存储器 (SRAM)、动态随机存取存储器 (DRAM)、其他类型的随机存取存储器 (RAM)、只读存储器 (ROM)、电可擦除可编程只读存储器 (EEPROM)、快闪记忆体或其他内存技术、只读光盘(CD-ROM)、数字多功能光盘 (DVD) 或其他光学存储、磁盒式磁带，磁带磁盘存储或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。

如上文说明的，如何提升Unity引擎的能力、适用范围，以及保障Unity引擎执行稳定、执行安全，是值得关注和迫切需求的。

在一些方案中，为了保障Unity引擎的执行稳定、安全，会选择预先配置允许被读取、使用的文件格式，使得Unity引擎仅对这些文件格式具有导入和处理能力，以避免对“安全性不可控”的文件进行处理，保障Unity引擎的执行安全。但这样的方式中，会使得Unity引擎的支持能力受到限制，例如，对于某些格式的文件，Unity引擎缺乏相应的处理能力，使得Unity引擎的能力、适用范围受到影响。

对此，本申请实施例提供了一种导入文件的方法，该方法获取针对Unity引擎的待加载文件；如果待加载文件的文件类型为目标类型，解封装待加载文件，得到解封装结果，其中，解封装结果中至少包括待加载文件的视频数据；基于解封装结果，生成动画剪辑文件；以及向Unity引擎导入动画剪辑文件，以使得Unity引擎通过执行动画剪辑文件来替代执行待加载文件。由此，使得在接收到针对Unity引擎的待加载文件时，利用动画剪辑文件作为实际导入、被执行的文件，以保障替代待加载文件中的内容可以实际、可靠地被执行。这样的方式，不仅可以提升Unity引擎的文件格式适用范围，还能保障Unity引擎的执行稳定性。

在实际场景中，该方法的执行主体可以是用户设备、或者用户设备与网络设备通过网络相集成所构成的设备，或者也可以是运行于上述设备的应用程序，用户设备包括但不限于计算机、手机、平板电脑、智能手表、手环等各类终端设备，网络设备包括但不限于如网络主机、单个网络服务器、多个网络服务器集或基于云计算的计算机集合等实现，可以用于实现设置闹钟时的部分处理功能。在此，云由基于云计算（Cloud Computing）的大量主机或网络服务器构成，其中，云计算是分布式计算的一种，由一群松散耦合的计算机集组成的一个虚拟计算机。

图1示出了本申请实施例提供的一种导入文件的过程100，过程100至少包括以下的处理步骤：

步骤S101，获取针对Unity引擎的待加载文件。

在本申请的实施例中，执行主体获取针对Unity引擎的待加载文件。在一些场景中，为了方便用户使用Unity引擎，获取待加载文件的执行主体中可以是用于配置Unity引擎的设备，由此，使得在用户可以直接在执行主体本机来使用Unity引擎。在一些实施例中，执行主体也可以是“Unity引擎”本身，即，执行主体可以是Unity引擎的组成部分（例如，是Unity引擎的一个单元），由此，以使得Unity引擎在本机便具有相应的处理能力，以达到简化配置的目的。在一些实施例中，Unity引擎可以处于运行状态，由此，使得本申请可以提供后续说明的导入方式，来向运行状态下的Unity引擎，以提升Unity引擎的使用效率。

在本申请的实施例中，用户可以基于其实际需求来提供任何形式的待加载文件，例如.fbx（Filmbox）、.obj（Wavefront OBJ）、.blend（Blender）和.dae（Collada）格式文件。.fbx是一种通用的三维模型文件格式，其支持保持Unity引擎的多种软件和平台之间的模型交换。.fbx文件可以包括虚拟对象的几何数据、材质、动画、骨骼等信息。.obj是一种三维模型文件格式，其是一种文本文件格式，可以被用于描述模型的几何形状、材质和纹理坐标等信息。.blend是Blender软件的原生文件格式，.blend文件可以包含模型的几何数据、材质、动画、骨骼等信息。在Unity引擎中，可以直接导入.blend文件进行编辑和使用。.dae是一种开放的三维模型文件格式，其支持包括Unity引擎的多种软件和平台之间的模型交换。.dae文件可以包含模型的几何数据、材质、动画、骨骼等信息等。

在一些实施例中，用户还可以提供VRM（Virtual Reality Markup Language）数据格式类型的待加载文件。VRM是一种基于glTF（GL Transmission Format）的开源文件格式，被广泛应用于创建和分享三维人物模型，特别在虚拟现实（VR）、增强现实（AR）以及三维（3D）动画领域。VRM模型通常包含形状、纹理、骨骼信息以及动画。glTF，是一种用于3D场景和模型的文件格式，被描述为“3D的JPEG”。glTF规定了标准的文件格式，可以存储3D模型的几何信息、纹理、材质以及动画等信息，能够高效、透明地传输3D图形内容。

步骤S102，如果待加载文件的文件类型为目标类型，解封装待加载文件，得到解封装结果。

在本申请的实施例中，执行主体在基于步骤S101获取到待加载文件后，可以解析待加载文件的类型。例如，执行主体可以基于文件格式确定待加载文件的文件类型。进一步地，如果执行主体确定待加载文件的文件类型为目标类型，则其可以解封装待加载文件，以得到解封装结果。在本公开的实施例中，目标类型可以预先被配置，通常目标类型可以包括无法被Unity引擎直接导入、运行，或者直接导入、运行会对Unity引擎的安全性存在不良影响的文件类型。在一些实施例中，目标类型也可以对于Unity引擎适配性较低，不宜为Unity引擎所处理的文件类型。

在一些实施例中，因为Unity引擎在对VRM数据格式类型进行处理时，需要借助VRM模型才能对例如动画数据、动作数据等进行解析，所以VRM数据格式类型的数据，在Unity引擎运行过程中、缺少VRM模型的情况下，可能难以被Unity引擎完全解析、运行和使用。在此情况下，可以将VRM数据格式类型确定为目标类型，以通过将其转化为可以被Unity引擎更好地使用的其他文件格式的方式，以替代直接执行VRM数据格式类型的文件。

进一步地，如果执行主体确定待加载文件的文件类型为目标类型，则可以解封装待加载文件，以获取至少包括待加载文件的视频数据的解封装结果。例如，执行主体可以通过对待加载文件的解封装操作，以提取其中的视频数据（例如，视频帧）等，以从中提取相应的视频数据、获取待加载文件的实质内容。在一些实施例中，解封装结果中还可以包括例如音频数据、音频轨道、字幕信息和字幕轨道等等。

在一些实施例中，封装结果中的视频数据可以是视频的关键帧（或者说，一组关键帧）。相应地，得到的解封装结果中还包括与关键帧对应的描述信息，以利用描述信息来指示相应关键帧数据的时间点（例如，关键帧的播放时间位置、关键帧位于视频播放轨道中的位置）以及关键帧数据所包括的关键点的动作信息（例如，关键帧包括的骨骼点、骨骼点所执行的动作，关键帧之间骨骼点的变化轨迹等等）。由此，可以通过关键帧和描述信息的方式来提供“视频内容”，以便于后续基于“关键帧”来还原视频中所涉及的动作（例如，虚拟对象所执行的动作）。

在一些实施例中，因为解封装操作得到的解封装结果并未被Unity引擎所实际执行，或者说，解封装操作并不会对Unity引擎的运行状况产生影响。所以，执行主体还可以利用针对Unity引擎的资源导入器加载待加载文件。例如，对于为Unity引擎提供可执行文件，或者例如Unity引擎本机的资源导入器，可以预先被赋予对于多个文件格式（例如，更为广泛的文件格式）的支持能力、解析权限，使得针对Unity引擎的资源导入器对更多的文件格式执行处理（例如，解封装）。由此，执行主体可以利用针对Unity引擎的资源导入器来解封装待加载文件，以简化配置要求。

为方便理解，以VRM数据格式类型的文件（为方便描述，将其简称为VRM文件）进行示例。VRM文件中的数据可以包括例如人物模型的几何信息、材质信息、动画数据以及其他相关属性，其可以将这些数据封装为一个文件，实现存储。例如，VRM文件通常可以使用二进制或JSON格式对这些数据的封装结果进行存储。在一些实施例中，VRM文件中的动画数据格式也可以默认使用二进制格式，例如VRM文件可以以glTF作为基础，以利用glTF文件格式本身支持使用二进制格式的特性，来存储动画数据。相应地，执行主体可以提供解封装VRM文件的方式，来获得其中的动画数据。

为方便理解，还可以同时参考图2。图2示出了本申请一实施例提供的生成动画剪辑文件的过程200。

在过程200中，VRM数据格式类型的待加载文件（例如，VRM文件210）时，对于属于目标文件类型（例如，VRM数据格式类型）的VRM文件210，执行主体可以解封装VRM文件210。

例如，在过程200中，VRM文件210被解封装后，至少可以得到一组关键帧220、关键帧的描述信息230。

步骤S103，基于解封装结果，生成动画剪辑文件。

在本申请的实施例中，执行主体在基于上述步骤S102得到解封装结果后，可以利用解封装结果（例如，视频数据）来生成动画剪辑文件。动画剪辑文件（Animation Clip）是一种用于存储和管理动画数据的格式，可以被在游戏引擎和动画软件中使用。动画剪辑文件是Unity引擎中的一种资源类型，可以包括例如虚拟对象的动画信息（例如，关键帧）以及其他相关属性。

在一些实施例中，基于解封装结果，生成动画剪辑文件，包括：将关键帧处理为Unity引擎的Unity关键帧；创建初始动画剪辑文件；以及将Unity关键帧添加至初始动画剪辑文件，生成动画剪辑文件。 具体地，执行主体可以处理解封装结果中的视频数据的关键帧，以将其转化为可以被Unity引擎所识别、使用的Unity关键帧（例如，可以通过对关键帧的格式转化、对关键帧的参数信息的调整，以将关键帧调整为Unity关键帧）。进一步地，执行主体可以创建一个初始动画剪辑文件，然后通过向初始动画剪辑文件中插入各个Unity关键帧的方式，来得到实际包括待加载文件的关键真内容的动画剪辑文件。动画剪辑文件可以基于关键帧动画的概念，来利用Unity关键帧定义虚拟对象在动画过程中的变换、位移、旋转、缩放等属性的变化。在一些实施例中，执行主体还可以基于Unity关键帧，通过在时间轴上插值的方式，来平滑地过渡物体的属性。在动画剪辑文件中，“Unity关键帧”可以与多个属性的曲线相关联，以实现例如位置、旋转、缩放等内容。每个属性曲线会根据Unity关键帧的时间戳和值定义该属性在动画过程中的变化方式。进一步地，动画剪辑文件中可以通过存储Unity关键帧的时间戳和属性值，来表征虚拟形象在一段时间内的动画状态。例如，Unity关键帧可以包括一个或多个属性的值，以用于描述物体的不同属性在该时间点上的状态。由此，通过在动画剪辑文件中调整和编辑属性曲线，例如调整Unity关键帧之间的缓动、旋转曲线的切线类型等，可以精确控制动画的效果和行为。

应当理解的是，对于一些文件格式，其关键帧可能直接可以满足Unity的使用要求，则在此情况下，Unity关键帧和关键帧可能实际上相同。

在一些实施例中，动画剪辑文件还可以被设置播放速度、循环模式、持续时间等参数，以对针对虚拟形象的动画的播放方式和时间范围进行调整、配置。在一些实施例中，动画剪辑文件作为游戏引擎或动画软件的一种资源类型，其也可以被与其他资源（如模型、材质）进行关联和管理。在此情况下，动画剪辑文件通过被分配给游戏对象或角色模型的方式，以使得可以在游戏或动画系统中播放和应用动画。

由此，可以通过将例如VRM模型文件里的动画数据在运行时转化为动画剪辑文件的方式，使得开发者在不了解例如VRM模型文件的细节的情况下，也能方便地利用Unity引擎来处理例如VRM数据格式类型的待加载文件。

示例性地，可以继续参考图2。在过程200中，执行主体可以在获取到一组关键帧220和对应的描述信息230后，通过向初始动画剪辑文件250连续插入关键帧的方式，以生成动画剪辑文件250’。

在一些实施例中，执行主体解封装待加载文件、得到解封装结果时，还会一并得到待加载文件中的音频数据。由此，以避免文件格式转换造成内容丢失，以使得其他类型的数据可以正常地被使用。

在一些实施例中，如果解封装结果中还包括待加载文件的音频数据，执行主体还可以获取视频数据位于动画剪辑文件中的视频播放轨道和音频数据的音频播放轨道。例如，执行主体可以获取视频数据位于动画剪辑文件中的视频播放轨道，和用于指示音频数据播放的音频播放轨道。进一步地，执行主体可以基于例如播放时间将视频播放轨道和音频播放轨道对齐后，参照对齐后的结果将音频数据添加至动画剪辑文件。例如，在获取视频播放轨道后，执行主体可以基于其中所指示的各个关键帧的播放位置，来对齐与这些播放帧对应的音频的播放位置，进而对齐音频播放轨道。由此，以避免因例如在生成动画剪辑文件时，调整关键帧的播放位置所导致的“音画不同步”，并使得在动画剪辑文件中，还可以基于待加载文件所使用的视频播放轨道和音频播放轨道来还原直接处理、使用待加载文件时所能带来的呈现效果。

示例性地，可以继续参考图2。在过程200中，执行主体通过对VRM文件210解封装后，还可以得到音频播放轨道240和音频数据250作为解封装结果。

进一步地，执行主体在基于一组关键帧220和描述信息230得到动画剪辑文件250’时，还可以利用视频播放轨道来对音频播放轨道240进行调整（例如，对其中指示的音频播放的时间位置进行调整），以得到调整后的音频播放轨道240’。进一步地，执行主体可以基于音频播放轨道240来向动画剪辑文件250’插入音频数据250。

步骤S104，向Unity引擎导入动画剪辑文件。

在本申请的实施例中，执行主体在基于上述步骤S103得到动画剪辑文件后，可以向Unity引擎导入动画剪辑文件。进一步地，执行主体，可以指示Unity引擎实际执行该动画剪辑文件，以替代执行待加载文件，以使得Unity引擎通过执行动画剪辑文件来替代执行待加载文件。由此，对于待加载文件而言，虽然其源文件无法被Unity引擎所执行，或者被Unity引擎所拒绝执行，但其内容仍可以被Unity引擎使用、处理，以提升Unity引擎的文件格式适用范围，并保障Unity引擎的执行稳定性。例如，对于动画剪辑文件，其可以被Unity引擎通过Animation或Animator组件来使用、播放，以替代地直接使用、播放待加载文件。

继而，本申请提供的导入文件的方法，获取针对Unity引擎的待加载文件；如果待加载文件的文件类型为目标类型，解封装待加载文件，得到解封装结果，其中，解封装结果中至少包括待加载文件的视频数据；基于解封装结果，生成动画剪辑文件；以及向Unity引擎导入动画剪辑文件，以使得Unity引擎通过执行动画剪辑文件来替代执行待加载文件。由此，使得在接收到针对Unity引擎的待加载文件时，利用动画剪辑文件作为实际导入、被执行的文件，以保障替代待加载文件中的内容可以实际、可靠地被执行。这样的方式，不仅可以提升Unity引擎的文件格式适用范围，还能保障Unity引擎的执行稳定性。

本申请实施例还提供了一种用于导入文件的装置，该装置的结构如图3所示出的装置300。装置300包括：获取模块310，被配置为获取针对Unity引擎的待加载文件；解封装模块320，被配置为如果待加载文件的文件类型为目标类型，解封装待加载文件，得到解封装结果，其中，解封装结果中至少包括待加载文件的视频数据；生成模块330，被配置为基于解封装结果，生成动画剪辑文件；以及导入模块340，被配置为向Unity引擎导入动画剪辑文件，以使得Unity引擎通过执行动画剪辑文件来替代执行待加载文件。

在一些实施例中，解封装待加载文件，包括：利用针对Unity引擎的资源导入器加载待加载文件。

在一些实施例中，视频数据包括关键帧，解封装结果中还包括与关键帧对应的描述信息，描述信息用于指示相应关键帧数据的时间点以及关键帧数据所包括的关键点的动作信息。

在一些实施例中，基于解封装结果，生成动画剪辑文件，包括：将关键帧处理为Unity引擎的Unity关键帧；创建初始动画剪辑文件；以及将Unity关键帧添加至初始动画剪辑文件，生成动画剪辑文件。

在一些实施例中，解封装结果中还包括待加载文件的音频数据。

在一些实施例中，如果解封装结果中还包括待加载文件的音频数据，装置300还包括：添加模块，被配置为获取视频数据位于动画剪辑文件中的视频播放轨道和音频数据的音频播放轨道；基于视频播放轨道和音频播放轨道的对齐结果，将音频数据添加至动画剪辑文件。

在一些实施例中，目标类型包括VRM数据格式类型。

基于同一发明构思，本申请实施例中还提供了一种电子设备，电子设备对应的方法可以是前述实施例中的导入文件的方法，并且其解决问题的原理与该方法相似。本申请实施例提供的电子设备包括：至少一个处理器；以及与至少一个处理器通信连接的存储器；其中，存储器存储有可被至少一个处理器执行的指令，指令被至少一个处理器执行，以使至少一个处理器能够执行前述本申请的多个实施例的方法和/或技术方案。

电子设备可以是用户设备、或者用户设备与网络设备通过网络相集成所构成的设备，或者也可以是运行于上述设备的应用程序，用户设备包括但不限于计算机、手机、平板电脑、智能手表、手环等各类终端设备，网络设备包括但不限于如网络主机、单个网络服务器、多个网络服务器集或基于云计算的计算机集合等实现，可以用于实现设置闹钟时的部分处理功能。在此，云由基于云计算（Cloud Computing）的大量主机或网络服务器构成，其中，云计算是分布式计算的一种，由一群松散耦合的计算机集组成的一个虚拟计算机。

图4示出了适用于实现本申请实施例中的方法和/或技术方案的一种电子设备的结构，该电子设备400包括中央处理单元（CPU，Central Processing Unit）401，其可以根据存储在只读存储器（ROM，Read Only Memory）402中的程序或者从存储部分408加载到随机访问存储器（RAM，Random Access Memory）403中的程序而执行各种适当的动作和处理。在RAM 403中，还存储有系统操作所需的各种程序和数据。CPU 401、ROM 402以及RAM 403通过总线404彼此相连。输入/输出（I/O，Input / Output）接口405也连接至总线404。

以下部件连接至I/O接口405：包括键盘、鼠标、触摸屏、麦克风、红外传感器等的输入部分406；包括诸如阴极射线管（CRT，Cathode Ray Tube）、液晶显示器（LCD，Liquid Crystal Display）、LED显示器、OLED显示器等以及扬声器等的输出部分407；包括硬盘、光盘、磁盘、半导体存储器等一个或多个计算机可读介质的存储部分408；以及包括诸如LAN（局域网，Local Area Network）卡、调制解调器等的网络接口卡的通信部分409。通信部分409经由诸如因特网的网络执行通信处理。

特别地，本申请实施例中的方法和/或实施例可以被实现为计算机软件程序。例如，本申请公开的实施例包括一种计算机程序产品，其包括承载在计算机可读介质上的计算机程序，该计算机程序包含用于执行流程图所示的方法的程序代码。在该计算机程序被中央处理单元（CPU）401执行时，执行本申请的方法中限定的上述功能。

本申请另一实施例还提供了一种计算机可读存储介质，其上存储有计算机程序指令，计算机程序指令可被处理器执行以实现前述本申请的任意一个或多个实施例的方法和/或技术方案。

具体来说，本实施例可以采用一个或多个计算机可读介质的任意组合。计算机可读介质可以是计算机可读信号介质或者计算机可读存储介质。计算机可读存储介质例如可以是包括但不限于电、磁、光、电磁、红外线、或半导体的系统、装置或器件，或者任意以上的组合。计算机可读存储介质的更具体的例子（非穷举的列表）包括：具有一个或多个导线的电连接、便携式计算机磁盘、硬盘、随机存取存储器（RAM）、只读存储器(ROM)、可擦式可编程只读存储器（EPROM或闪存）、光纤、便携式紧凑磁盘只读存储器（CD-ROM）、光存储器件、磁存储器件、或者上述的任意合适的组合。在本文件中，计算机可读存储介质可以是任何包含或存储程序的有形介质，该程序可以被指令执行系统、装置或者器件使用或者与其结合使用。

计算机可读信号介质可以包括在基带中或者作为载波一部分传播的数据信号，其中承载了计算机可读的程序代码。这种传播的数据信号可以采用多种形式，包括但不限于电磁信号、光信号或上述的任意合适的组合。计算机可读的信号介质还可以是计算机可读存储介质以外的任何计算机可读介质，该计算机可读介质可以发送、传播或者传输用于由指令执行系统、装置或者器件使用或者与其结合使用的程序。

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

可以以一种或多种程序设计语言或其组合来编写用于执行本申请操作的计算机程序代码，程序设计语言包括面向对象的程序设计语言—诸如Java、Smalltalk、C++，还包括常规的过程式程序设计语言—诸如“C”语言或类似的程序设计语言。程序代码可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中，远程计算机可以通过任意种类的网络，包括局域网（LAN）或广域网（WAN）连接到用户计算机，或者，可以连接到外部计算机（例如利用因特网服务提供商来通过因特网连接）。

附图中的流程图或框图示出了按照本申请各种实施例的设备、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段、或代码的一部分，该模块、程序段、或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意，在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个接连地表示的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或操作的专用的针对硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

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，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，例如多个单元或页面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。

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。上述集成的单元既可以采用硬件的形式实现，也可以采用硬件加软件功能单元的形式实现。

上述以软件功能单元的形式实现的集成的单元，可以存储在一个计算机可读取存储介质中。上述软件功能单元存储在一个存储介质中，包括若干指令用以使得一个计算机设备（可以是个人计算机，服务器，或者网络设备等）或处理器（processor）执行本申请各个实施例方法的部分步骤。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。

最后应说明的是：以上实施例仅用以说明本申请的技术方案，而非对其限制；尽管参照前述实施例对本申请进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本申请各实施例技术方案的精神和范围。

此外，显然“包括”一词不排除其他单元或步骤，单数不排除复数。装置权利要求中陈述的多个单元或装置也可以由一个单元或装置通过软件或者硬件来实现。第一，第二等词语用来表示名称，而并不表示任何特定的顺序。

1. 一种导入文件的方法，包括：

获取针对Unity引擎的待加载文件；

如果所述待加载文件的文件类型为目标类型，解封装所述待加载文件，得到解封装结果，其中，所述解封装结果中至少包括所述待加载文件的视频数据；

基于所述解封装结果，生成动画剪辑文件；以及

向所述Unity引擎导入所述动画剪辑文件，以使得所述Unity引擎通过执行所述动画剪辑文件来替代执行所述待加载文件。

2. 根据权利要求1所述的方法，其中解封装所述待加载文件，包括：

利用针对所述Unity引擎的资源导入器加载所述待加载文件。

3. 根据权利要求2所述的方法，其中所述视频数据包括关键帧，所述解封装结果中还包括与所述关键帧对应的描述信息，所述描述信息用于指示相应关键帧数据的时间点以及所述关键帧数据所包括的关键点的动作信息。

4. 根据权利要求3所述的方法，其中所述基于所述解封装结果，生成动画剪辑文件，包括：

将所述关键帧处理为所述Unity引擎的Unity关键帧；

创建初始动画剪辑文件；以及

将所述Unity关键帧添加至所述初始动画剪辑文件，生成动画剪辑文件。

5. 根据权利要求1所述的方法，其中所述解封装结果中还包括所述待加载文件的音频数据。

6. 根据权利要求4所述的方法，如果所述解封装结果中还包括所述待加载文件的音频数据，还包括：

获取所述视频数据位于所述动画剪辑文件中的视频播放轨道和所述音频数据的音频播放轨道；

基于所述视频播放轨道和所述音频播放轨道的对齐结果，将所述音频数据添加至所述动画剪辑文件。

7. 根据权利要求1-6中任一项所述的方法，其中所述目标类型包括VRM数据格式类型。

8. 一种用于导入文件的装置，包括：

获取模块，被配置为获取针对Unity引擎的待加载文件；

解封装模块，被配置为如果所述待加载文件的文件类型为目标类型，解封装所述待加载文件，得到解封装结果，其中，所述解封装结果中至少包括所述待加载文件的视频数据；

生成模块，被配置为基于所述解封装结果，生成动画剪辑文件；以及

导入模块，被配置为向所述Unity引擎导入所述动画剪辑文件，以使得所述Unity引擎通过执行所述动画剪辑文件来替代执行所述待加载文件。

9. 一种电子设备，所述电子设备包括：

至少一个处理器；以及

与所述至少一个处理器通信连接的存储器；其中，

所述存储器存储有可被所述至少一个处理器执行的指令，所述指令被所述至少一个处理器执行，以使所述至少一个处理器能够执行权利要求1至7中任一项所述的方法。

10. 一种计算机可读介质，其上存储有计算机程序指令，所述计算机程序指令可被处理器执行以实现如权利要求1至7中任一项所述的方法。

本申请提供了一种导入文件的方法、装置、电子设备及计算机可读介质，该申请获取针对Unity引擎的待加载文件；如果待加载文件的文件类型为目标类型，解封装待加载文件，得到解封装结果，其中，解封装结果中至少包括待加载文件的视频数据；基于解封装结果，生成动画剪辑文件；以及向Unity引擎导入动画剪辑文件，以使得Unity引擎通过执行动画剪辑文件来替代执行待加载文件。由此，使得在接收到针对Unity引擎的待加载文件时，利用动画剪辑文件作为实际导入、被执行的文件，以保障替代待加载文件中的内容可以实际、可靠地被执行。这样的方式，不仅可以提升Unity引擎的文件格式适用范围，还能保障Unity引擎的执行稳定性。



图1



图2



图3



图4

