

媒体文件播放系统及方法

技术领域

本发明实施例涉及媒体播放技术领域，更具体地说，涉及一种媒体文件播放系统及方法。

5 发明背景

目前，在 Window 平台上，对各种媒体文件的播放通常采用 DirectShow 技术框架，其使用一种叫 Filter Graph 的模型来管理整个多媒体数据流的处理过程，参与数据处理的各功能模块称为 Filter（滤镜）。各 Filter 在 Filter Graph 中按一定的顺序连接成一条“流水线”协同工作，
10 这条“流水线”通常称之为渲染线。按功能来分，Filter 大致分为三类：源滤镜（Source Filter）、转换滤镜（Transform Filter）和渲染滤镜（Rendering Filter）。源滤镜主要负责获取数据，并将数据向下一个滤镜传输；转换滤镜主要负责数据的格式转换，例如数据流分离/合成、解码/编码等，并将数据继续向下一个滤镜传输；渲染滤镜主要负责数据的最
15 终去向，将数据传送给显卡、声卡进行多媒体的演示。

Filter 含有被称为 Pin 的结构，一种是输出 Pin，一种是输入 Pin。一个 Filter 一般由一个或多个 Pin 组成，Filter 之间通过 Pin 相互连接。当上一个 Filter 要和下一个 Filter 连接时，上一个 Filter 的输出 Pin 要和下一个 Filter 的输入 Pin 协商，若输出 Pin 的媒体流格式能被下一个 Filter
20 的输入 Pin 支持，则两者可以成功建立连接。

播放器在播放媒体文件之前，会获取所有已知的 Filter 并按一定顺序连接出一条渲染线，从而达到播放媒体的目的。该过程称之为建立渲染线。每一个 Filter 都有一个称为优先级的属性，DirectShow 建立渲染

线的时候默认会把优先级高的 Filter 连入渲染线。然而, 优先级高的 Filter 不一定能连入成功, 因为该 Filter 的输入 Pin 可能不支持已连入渲染线中的上一个 Filter 的输出 Pin 输出的媒体流格式。因此, 达到播放媒体的目的, 需建立合适的渲染线。

5 目前在各种播放器中, 一般采用以下两种方式建立渲染线播放媒体。

(1) 直接调用 DirectShow 提供的接口方法 RenderFile(strFilePath)。该接口方法依赖于 DirectShow 技术框架内置的建立渲染线的默认逻辑, 为以 strFilePath 为全路径的媒体建立渲染线。这种方式是最简单的渲染策略, 在该方式中, 播放器可以完全不考虑渲染线中各 Filter 是什么及各 Filter 的顺序。该方法对于 DirectShow 内置的建立渲染线的默认逻辑能处理的媒体有效。然而, 很多媒体由于压制的方法、工具不同及传输时造成的局部错误, 导致 DirectShow 无法通过 RenderFile (strFilePath) 智能地为媒体建立适当的渲染线进行播放, 从而导致播放效果可能很

10 差。

15 差。

例如, 播放视频流为 H264 的媒体文件时, 系统中有解码器 CL264 Decoder8 和 MPCHC, 两者都可支持对 H264 视频流的解码 (即两者都可以连入渲染线), 且根据 DirectShow 技术框架内置的建立渲染线的默认逻辑, 前者比后者的优先级高。则 CL264 Decoder8 解码器优先连入了渲染线, 而 MPCHC 解码器没有机会连入渲染线。事实上, 在用显卡对媒体文件的播放进行高清加速时, 由于该文件自身的某些特性 (例如媒体文件分辨率、容器格式等), 使用 CL264 Decoder8 来解码效果很不好, 会花屏, 更严重的由 CL264 Decoder8 解码出的视频流可能根本无法显示到屏幕上, 而出现黑屏等现象; 而用 MPCHC 解码的效果反而更好。然

20 而 DirectShow 内置的建立渲染线的默认逻辑没有给 MPCHC 机会, 因此

25 而 DirectShow 内置的建立渲染线的默认逻辑没有给 MPCHC 机会, 因此

采用 DirectShow 内置的建立渲染线的默认逻辑, 未考虑到媒体文件自身特点, 不能为各种媒体建立适当的渲染线, 从而导致媒体文件的播放效果差。

(2) 依靠播放器编写者的经验, 在播放器代码中预先设置好各 Filter 的优先级, 以实现播放器能对 DirectShow 内置的建立渲染线的默认逻辑无法处理或者处理效果不好的媒体进行较好的播放, 该方法能克服第一种方法的缺点。这种做法表现为: 对不同的媒体格式, 有区别的设置 Filter 的优先级。比如, 对 H264 的视频, 设置 MPCHC 解码器的优先级高于 CL264 Decoder8 解码器; 而对于 mpeg2 的视频, 设置 CL264 Decoder8 解码器的优先级高于 MPCHC 的优先级。

然而, 这种方法也仅仅是部分考虑到了媒体文件自身的特点, 仍然具有以下缺陷: A. 通用性差。对于同一格式的媒体文件, 适用的 Filter 也有不同, 按照设置好的 Filter 优先级建立渲染线, 也会导致媒体的播放效果差。例如, 对于 H264 视频, 某些媒体文件使用 CL264 Decoder8 来解码比用 MPCHC 解码效果好, 而另一部 H264 视频流格式的媒体文件则使用 MPCHC 解码器比使用 CL264 Decoder8 解码器的效果好。针对这种情况, 播放器编写者必须对这些媒体文件进行测试分析, 然后修改或添加程序来应付播放这一部分媒体时如何设置 Filter 的优先级, 并把修改后的新版本发出去才能解决用户机器上播放这部分媒体时播放效果不好的问题。B. 修改频繁。由于不同的用户机器环境不同 (比如显卡加速能力不同), 而播放器编写者是无法实现预知的, 因此经过测试修改后发出去的播放器版本可能无法解决所有用户的问题。因不同用户机器环境对播放器的要求, 要解决播放效果差的问题, 则需频繁修改播放器的版本。

发明内容

本发明实施例提供一种媒体文件播放系统和方法，实现建立适合各种媒体文件的渲染线，从而得到更好的媒体播放效果。

本发明实施例提供的技术方案包括：

- 5 一种媒体文件播放系统，包括渲染规则库与播放器单元，渲染规则库中存储了渲染规则，播放器单元包括：规则管理模块、渲染策略环境模块和播放模块，其中，

所述规则管理模块用于从所述渲染规则库中获取渲染规则，并对获取的渲染规则进行解析；

- 10 所述渲染策略环境模块，用于根据所述规则管理模块解析出的渲染规则生成渲染线；

所述播放模块，用于利用所述渲染线播放媒体文件。

一种媒体文件播放方法，所述方法包括以下步骤：

- 15 A，从预先建立的渲染规则库中获取渲染规则，并对获取的渲染规则进行解析；

B，根据解析出的渲染规则生成渲染线；

C，应用所述渲染线播放媒体文件。

- 20 综上技术方案可以看出，上述媒体文件播放系统和方法，通过建立渲染规则，并将渲染规则集中存储在渲染规则库中，当播放器播放媒体文件时，遍历渲染规则库中的渲染规则，根据渲染规则中定义的逻辑因果关系执行规则中定义的操作，能达到调整渲染线的目的，建立适合媒体文件的渲染线，从而可以得到更好的媒体播放效果。

附图简要说明

图 1 是一个实施例中媒体文件播放系统的结构示意图；

- 图 2 是一个实施例中渲染策略环境模块的结构示意图；
图 3 是一个实施例中规则管理模块的结构示意图；
图 4 是一个实施例中媒体文件播放方法的流程图；
图 5 是一个实施例中生成渲染策略的方法流程图；
5 图 6 是一个实施例中解析渲染规则的方法流程图；
图 7 是一个实施例中解析条件节点的方法流程图；
图 8 是一个实施例中解析操作节点的方法流程图；
图 9 是一个实施例中条件二叉树的示意图；
图 10 是一个实施例中计算条件二叉树根节点的真假值的方法流程图；
10 图；
图 11 是一个实施例中创建原子条件类实例和原子操作类实例的示意图。

实施本发明的方式

- 本发明实施例中，将播放媒体时运用到的渲染线的各种建立逻辑统一抽象为渲染规则模型，并以此建立一个渲染规则库，这样，基于该渲染规则库可以实现建立适合各种媒体文件的渲染线，以便得到更好的媒体播放效果。

为了使本发明的目的、技术方案和优点更加清楚，下面结合附图和具体实施例对本发明进行详细描述。

- 20 参见图 1，图 1 示出了一个实施例中的媒体文件播放系统，该系统包括渲染规则库 10 和播放器单元 20，其中：

- 渲染规则库 10 中预先存储了渲染规则。渲染规则中定义了在任何条件下执行何种操作。所谓条件即影响播放效果的某些因素，包括以下内容：（1）媒体文件本身的因素，例如媒体的分辨率、媒体的容器格式、
25 视频流格式、音频流格式等；（2）Filter 本身的局限性，例如 Filter 在 Pin

的连接协商中声明支持某种视频或音频流格式，但实际支持的效果并不好；（3）操作系统软环境，如操作系统等。例如，在 Windows XP 系统下高清加速的时候，渲染滤镜一般采用 VMR9 就可以支持，但在 Vista 或 Win7 系统下，则必须用 EVR 渲染滤镜；（4）机器硬件能力，如显卡等，主要表现显卡能不能支持高清加速，支持高清加速的时候，显卡最多缓存多少参考帧等；（5）播放模式，即用户想让播放器以什么样的方式去播放媒体文件。例如用户播放时是否考虑占中央处理器资源，若用户在播放媒体文件时还想做其他事情，则播放器播放媒体时采用尽量少占中央处理器资源的智能加速方式等。条件可以是上述各种因素中的至少一个或者任意组合，本发明并不对此做限制。而相应地，操作包括设置高清加速和调整 Filter 的优先级等中的至少一个或者任意组合，其与条件对应。可以看出，渲染规则实质上描述了条件和操作的对应关系。

在一个实施方式中，渲染规则库 10 中描述渲染规则的数据是静态数据，其可存在播放器程序代码的一组硬编码的数据结构中，也可采用脚本文件存于播放器的安装目录下，或位于网络上的某个集中的数据库中，本发明实施例并不限定，但较佳地，需要保证该静态数据和具体程序逻辑分离，这样易于修改且管理。

播放器单元 20 包括规则管理模块 200、渲染策略环境模块 202 和播放模块 204。

其中，规则管理模块 200 用于获取并解析渲染规则，这里，规则管理模块 200 可从渲染规则库中获取渲染规则；至于规则管理模块 200 解析渲染规则，其具体为：获取渲染规则中的逻辑因果关系，并将该获取的逻辑因果关系的格式转换为符合格式比如计算机识别的格式。

渲染策略环境模块 202 根据规则管理模块 200 解析的渲染规则生成渲染线。

播放模块 204 用于利用所述渲染线播放媒体文件。

这里，规则管理模块 200 仅能通过解析渲染规则获取该渲染规则中的逻辑因果关系，该逻辑因果关系实质上为条件和操作的对应关系，其中，该逻辑因果关系中的条件包含各种条件，比如原子条件
5 （ServiceAtomCondition）和子条件（子 Condition Group），其中，原子条件为只有业务环境能够识别的关键字且其代表的是一个条件，其不可再细分；而子条件还可以进一步细分，比如细分为原子条件等，基于此，可将该逻辑因果关系中的条件简称为条件节点，相应地，将该逻辑因果关系中的操作简称为操作节点；这里，操作节点由一些原子操作
10 （ServiceAtomAction）构成，其中，原子操作是指只有业务环境能够识别的关键字并且代表的是一种操作，其不可再细分。但是，规则管理模块 200 并不能识别该条件节点中原子条件和该操作节点中原子操作的具体含义，在一个实施方式中，规则管理模块 200 通过调用渲染策略环境模块 202 预先提供的应用程序接口将获取的条件节点中的原子条件和操
15 作节点中的原子操作传送给渲染策略环境模块 202 进行识别。

基于此，图 2 示出了一个实施例中的渲染策略环境模块 202 的一个结构。如图 2 所示，渲染策略环境模块 202 包括条件生成模块 2020、条件判断模块 2022、操作生成模块 2026 和操作执行模块 2028。

其中，条件生成模块 2020 获取条件节点中原子条件的描述，对获取的原子条件的描述进行解析并创建原子条件类实例，以及将该原子条件类实例返回给规则管理模块 200。
20

条件判断模块 2022 获取媒体文件信息，根据媒体文件信息判断原子条件的真假值，并将原子条件的真假值返回至规则管理模块 200。例如：原子条件是“当前媒体文件的视频流格式是 H264”，根据媒体文件
25 信息可知当前媒体文件的视频流格式是 H264，基于此，可以看出条件

判断模块 2022 判断原子条件的真假值为“真”。

操作生成模块 2024 获取操作节点中的原子操作的描述，对原子操作的描述进行解析并创建原子操作类实例，以及将原子操作类实例返回至规则管理模块 200。

操作执行模块 2026 在接收到规则管理模块 200 发送的渲染规则中

条件节点成立的通知时，执行所述渲染规则中条件成立时对应的操作。

由于渲染线表示了条件和执行操作的对应关系，当操作执行模块

2026 执行操作时，很显然得到条件和执行操作的对应关系，即可生成对

应的渲染线。相应地，当操作执行模块 2026 不执行操作时，即该条件

和操作的不对应，也可生成对应的渲染线。

基于图 2 所示的渲染策略环境模块 202 的结构，本发明实施例还提

供了规则管理模块 200 的结构，具体如图 3 所示。在图 3 中，规则管理

模块 200 包括条件二叉树生成模块 2000 和条件二叉树计算模块 2002。

其中，条件二叉树生成模块 2000 根据渲染规则中的条件节点生成

条件二叉树。

由于渲染规则中的一个条件节点可能包含多个原子条件，其可由其

所包含的原子条件进行逻辑组合而成。条件二叉树则形象的表示出了每

个条件节点中所包含的原子条件以及各原子条件之间的逻辑关系。

条件二叉树计算模块 2002 根据条件判断模块 2022 返回的原子条件

的真假值递归计算条件二叉树的根节点的真假值。这里，条件二叉树的

根节点的真假值决定了所述渲染规则中条件节点是否成立，当条件二叉

树计算模块 2002 计算出条件二叉树的根节点的真假值为真时，发送渲

染规则中条件节点成立的通知给上述操作执行模块 2026，以便操作执行

模块 2026 执行相应地操作。

以上对本发明实施例提供的系统进行了描述，下面结合附图和实施

例对本发明提供的方法进行描述。

图 4 示出了一个实施例中媒体文件播放方法的流程,具体过程如下:

在步骤 S10 中, 获取渲染规则。

这里, 步骤 S10 可从渲染规则库中获取渲染规则。如上所述, 该获取的渲染规则中定义了何种条件下执行何种操作, 这里所述的条件为影响播放效果的因素。

在步骤 S20 中, 解析所述渲染规则。

具体地, 步骤 S20 为: 通过解析所述渲染规则获取渲染规则中的逻辑因果关系, 并将该获取的逻辑因果关系的格式转换为符合格式比如计算机识别的格式。

在步骤 S30 中, 根据解析出的渲染规则生成渲染线。

在步骤 S40 中, 应用所述渲染线播放媒体文件。

其中, 上述步骤 S30 具体操作可参见图 5 所示的流程。

图 5 示出了一个实施例中生成渲染线的方法流程, 具体过程如下:

在步骤 S200 中, 获取所述渲染规则中所有条件节点下原子条件的描述, 调用渲染策略环境模块 202 解析原子条件的描述并创建原子条件类实例。

在一个渲染规则中, 一个条件节点可能包含多个原子条件, 对于每个原子条件, 需调用渲染策略环境模块 202 解析原子条件的描述, 并创建原子条件类实例, 以及将原子条件类实例返回至规则管理模块 200。

在步骤 S202 中, 获取媒体文件信息和机器参数, 根据媒体文件信息和机器参数判断原子条件的真假值。

在一个实施方式中, 调用渲染策略环境模块 202 提供的应用程序接口计算原子条件的真假值, 这里, 当解析的原子条件与媒体文件信息或机器参数符合时, 则确定原子条件的真假值为“真”, 不符合时确定原

子条件的真假值为“假”，将确定出的原子条件的真假值返回规则管理模块 200。

需要说明的是，上述媒体文件信息包括媒体文件本身的特性（例如媒体文件格式、分辨率等），机器参数包括用户机器的操作系统软环境
5 和硬件参数等。

在步骤 S204 中，根据渲染规则中的条件节点生成条件二叉树。

如上所述，条件二叉树中表示了一个条件节点下的所有原子条件以及各原子条件之间的逻辑关系。

在步骤 S206 中，根据计算得到的原子条件的真假值，递归计算所述条件二叉树的根节点的真假值。
10

在一个实施方式中，规则管理模块 200 调用渲染策略环境模块 202 提供的应用程序接口计算原子条件的真假值，根据原子条件的真假值，对条件二叉树上的节点进行递归计算，从而得到根节点的真假值。

在步骤 S208 中，获取渲染规则中操作节点的原子操作的描述，调用渲染策略环境模块 202 解析原子操作的描述并创建原子操作类实例。
15

在渲染规则中，一个操作节点可能包含多个原子操作，原子操作依赖于业务环境，因此需调用渲染策略环境模块 202 解析原子操作的描述，创建的原子操作类实例返回至规则管理模块 200。

在步骤 S210 中，根据所述条件二叉树的根节点的真假值的计算结果，执行条件成立时对应的操作。
20

在一个实施方式中，计算得到条件二叉树的根节点的真假值为“真”，表示该渲染规则中的条件节点成立，则渲染策略环境模块 202 执行该条件节点下的操作，即步骤 S208 对应的操作，例如设置高清加速、调整 Filter 的优先级等。

可以看出，本实施例中，条件二叉树根节点的真假值决定了渲染规
25

则中条件是否能成立。由于渲染规则中定义了各种条件下所执行的操作，在符合渲染规则的情况下，操作执行模块 2026 则执行渲染规则中定义的操作，从而建立适合媒体的渲染线，以便播放单元 20 播放时得到更好的媒体播放效果。

5 至此，通过图 5 所示的步骤实现了渲染线的生成。为使本申请提供的图 4 和图 5 所示的技术方案更加清楚，下面以一个具体实施例进行详细描述。

该实施例中，采用可扩展标记语言（Extensible Markup Language，简称“XML”）描述渲染规则，生成脚本文件。该脚本文件的结构形式
10 如下：

```

    <Root>
        <Rule ID="ID1" >
            <ConditionGroup Operator="OP1">
                <ServiceAtomCondition1/>
                <ServiceAtomCondition2/>
15           ...
                <ServiceAtomCondition3/>
                ...
            <ConditionGroup Operator="OP2">
20           ....
            </ConditionGroup>
            </ConditionGroup>
            <Action>
                <ServiceAtomAction1/>
                <ServiceAtomAction2/>
25           </Action>
            </Rule>
```

```
....  
<Rule ID="IDn">  
....  
</Rule>  
5 </Root>
```

上述脚本文件的结构形式中，Root 包含各种渲染规则，是该脚本文件的根节点；Rule 节点表示一个渲染规则，即在何种条件下做何种操作的描述；ID 是渲染规则唯一的标识号；ConditionGroup 节点表示渲染规则中的条件，一个渲染规则中必定包含一个 ConditionGroup 节点。而一个 ConditionGroup 节点中可能包含子条件（ConditionGroup）以及各种原子条件（如上述结构中的 ServiceAtomCondition1、ServiceAtomCondition2 等），这些原子条件具有渲染策略环境模块 202 能识别的关键字。例如，ServiceAtomCondition1 可以是一个用来判断当前影片的视频流格式是否是 H264 的条件，形式如下：

```
15 <HDVideoFormat Operator="Eq" Value="H264"/>
```

该原子条件必须依赖渲染策略环境模块 202 才能识别，若根据渲染策略环境模块 202 得到当前媒体的视频流是 H264 时，则条件判断模块 2022 判断出该原子条件为真。

需要说明的是，ConditionGroup 节点中的 Operator 属性值可由规则管理模块 200 进行定义，该属性值可以是一些逻辑关系，例如，与（and）、或（or）关系等。当 Operator 属性值为“与”时，表示 ConditionGroup 节点下的各子节点所代表的条件是与的关系，即 ConditionGroup 节点下所有子节点所表示的条件都为真时，ConditionGroup 节点为真，否则为假；当 Operator 属性值为“或”时，表示 ConditionGroup 节点下的各子节点所代表的条件是或的关系，即 ConditionGroup 节点下所有子节点所

表示的条件有一个为真时，则 ConditionGroup 节点为真，否则为假。

还需要说明的是，ConditionGroup 节点下的子节点的 Operator 属性值不应与该 ConditionGroup 节点（称为父节点）的 Operator 属性值相同，否则，可以把子节点化简到父节点中。这可通过以下代码体现：

```

5      <ConditionGroup Operator=" And" >
          <HDVideoFormat Operator="Eq" Value="H264"/>
          <ConditionGroup Operator=" And" >
              <HDAudioFormat Operator="Eq" Value="EAC3"/>
              <System Operator="Eq" Value="WinVista"/>
10      </ConditionGroup>
    </ConditionGroup>
    表示的是
        if ( ( HDVideoFormat == H264 )
            && ( (HDAudioFormat == EAC3) && (System ==
15      WinVista) )
            )

```

在上述代码中可以看出，子节点的 Operator 属性值与父节点的 Operator 属性值相同，都为"And"，基于此，上述代码可以化简为：

```

        If( ( HDVideoFormat == H264 ) && ( HDAudioFormat ==
20      EAC3 ) && ( System == WinVista )
        )
    用脚本可表示为
    <ConditionGroup Operator="And">
        <HDVideoFormat Operator="Eq" Value="H264"/>视频流格
25      式是 H264
        <HDAudioFormat Operator="Eq" Value="EAC3"/>音频流格

```

式是 EAC3

<System Operator="Eq" Value="WinVista"/> 系统是
WinVista 系统

</ConditionGroup>。

5 Action 节点（操作节点）表示的是渲染规则中最顶层的
ConditionGroup 节点中条件的返回值是“真”时，需要执行的操作，该
Action 节点也可由一些原子操作（如上述结构中的 ServiveAtomAction1、
ServiceAtomAction2）进行逻辑组合构成。原子操作也具有渲染策略环
境模块 202 能识别的关键字，例如 ServiveAtomAction1 表现形式如下：

10 <SetTransformFilterPriority Name="WMVideo Decoder DMO"
Priority="PRIORITY_PREFERRED"/>

该描述可由渲染策略环境模块 202 识别出，该操作是要将名为
“WMVideo Decoder DMO”的解码器的优先级调为最高。

图 6 示出了该实施例中解析渲染规则的方法流程，具体过程如下：

15 在步骤 S60 中，获取渲染规则脚本中当前规则节点（Rule 节点）。

这里，渲染规则脚本包含多个规则节点（即上述脚本文件的结构中
含有多个 Rule 节点），因此需遍历脚本文件中的所有规则节点。

在步骤 S62 中，判断渲染规则脚本中是否还有当前规则节点的下一
个规则节点，若是，则进入步骤 S64，否则结束。

20 在步骤 S64 中，定位到当前规则节点的根 CondionGroup 节点。

在步骤 S66 中，解析当前 ConditonGroup 节点，生成原子条件类实
例并插入现有的条件二叉树中。

在步骤 S68 中，解析规则节点下的操作节点，将所述当前规则节点
的下一个规则节点作为当前规则节点，返回至步骤 S60。

25 具体地，步骤 S66 可通过图 7 所示。图 7 示出了一个实施例中解析

条件节点的方法流程，具体过程如下：

在步骤 S660 中，从该条件节点即 ConditionGroup 节点中确定当前子条件节点。

这里，一般将条件节点即 ConditionGroup 节点中的第一个子条件节点作为当前子条件节点。

在步骤 S662 中，判断所述当前子条件节点是否为 ConditionGroup 节点，若是，则进入步骤 S668，否则进入步骤 S664。

这里，ConditionGroup 节点为复合节点，其可进一步细分为原子节点。

在步骤 S664 中，将该当前子条件节点传给渲染策略环境模块 202 进行解析，生成原子条件类实例。

该实施例中，当子条件节点不是 ConditionGroup 节点时，说明该子条件节点下不再会有原子条件节点，则可将该子条件节点的描述传给渲染策略业务环境 202 进行解析。

在步骤 S666 中，将该原子条件类实例插入到条件二叉树中，之后，执行步骤 S669。

在步骤 S668 中，递归解析当前子条件节点，生成原子条件类实例并插入到现有的条件二叉树中，之后，执行步骤 S669。

在一个实施例中，当前的子条件节点是 ConditionGroup 节点时，也就说明该 ConditionGroup 节点下可能还含有原子条件节点，因此需递归解析该当前的 ConditionGroup 节点，直至最终得到原子条件节点为止。

在步骤 S669 中，判断当前子条件节点是否为条件节点即 ConditionGroup 节点中的最后一个子条件节点，如果是，结束，否则，将当前子条件节点的下一个子条件节点作为当前子条件节点，返回执行步骤 S660。

本实施例中，步骤 S68 所示的解析操作可通过图 8 所示。图 8 示出了一个实施例中解析操作节点的方法流程，具体过程如下：

在步骤 S680 中，定位到当前规则节点下的操作节点（即 Action 节点）。

5 在步骤 S682 中，从当前操作节点获取下一个原子操作节点。

在步骤 S684 中，判断是否还有原子操作节点，若是，则进入步骤 S686，否则结束。

在步骤 S686 中，将该原子操作节点传给渲染策略环境模块 202 进行解析，生成原子操作类实例。

10 该实施例中，对于脚本文件中 ConditionGroup 下的原子条件节点，以及 Aciton 节点下的原子操作节点，由于其具有与业务环境联系的语义，而规则管理模块 200 仅能处理原子条件之间的逻辑关系，以及在条件成立时执行何种操作的问题，因此规则管理模块 200 需将原子条件以及原子操作的 XML 节点传给渲染策略环境 202 模块进行解析，并创建
15 相应的原子条件类实例和原子操作类实例，返回给规则管理模块 200。

该实施例中，规则管理模块 200 采用条件二叉树表示脚本中的规则下的条件节点。例如，在一个脚本文件中，ContionGroup 节点的代码如下：

```
<ConditionGroup Operator="And">  
20   <HdVideoFormat Operator="Eq" Value="H264"/>  
    <HDAudioFormat Operator="Eq" Value="EAC3"/>  
    <System Operator="Eq" Value="WinVista"/>  
    <ConditionGroup Operator="Or">  
        <HDContainerFormat Operator="Eq" Value="MKV"/>  
25   <HDContainerFormat Operator="Eq" Value="MP4"/>  
    </ConditionGroup>  
</ConditionGroup>
```

</ConditionGroup>

将该 ConditionGroup 节点表示成条件二叉树如图 9 所示。该条件二叉树中，“&&”表示逻辑关系与（and），“||”表示逻辑关系或（or）。该条件二叉树从渲染规则的根 ConditionGroup 节点下的第一个原子条件节点开始遍历，条件二叉树的根节点为根 ConditionGroup 节点的 Operator 属性值（如图 9 中的根“&&”），用来表示条件二叉树左右两个原子条件节点的逻辑关系。在条件二叉树中，每个节点的左子节点是与业务环境相关的原子条件节点，当左子节点表示的原子条件节点在脚本中后续只有一个原子条件时，右子节点为后续的那个原子条件节点。当左子节点表示的原子条件在脚本中后续是一个子 ConditionGroup 节点，则右子节点为该子 ConditionGroup 节点的 Operator 属性值。当左子节点表示的原子条件在脚本中后续有不只一个子条件节点时，则右子节点为当前 ConditionGroup 节点的 Operator 属性值。

图 10 示出了一个实施例中计算条件二叉树的根节点的真假值的方法流程，具体过程如下：

在步骤 S100 中，调用渲染策略环境模块 202 计算左子节点所对应的原子条件的真假值。

在步骤 S102 中，判断右子节点是否是原子条件，若是，则进入步骤 S104，否则进入步骤 S106。

在步骤 S104 中，调用渲染策略环境模块 202 计算右子节点所对应的原子条件的真假值。

在步骤 S106 中，递归调用后续遍历的算法计算以右子节点为根节点的树所表示的条件的真假值。

该实施例中，当右子节点不是原子条件时，则右子节点是一个为逻辑关系的根节点，则需递归计算以该右子节点为根节点的树所表示的条

件的真假值。

在步骤 S108 中，根据各节点所表示的逻辑关系，对左右子节点的真假值进行逻辑运算，得到条件二叉树的根节点的真假值。

该实施例中，当条件二叉树的根节点的真假值为“真”时，表明渲染规则中的条件成立；若为“假”，则表明渲染规则中的条件不成立。

图 11 示出了一个实施例中创建原子条件类实例和原子操作类实例的示意图。

如图 11 所示，在规则管理模块 200 解析渲染规则时，会将表示原子条件的 XML 节点传给渲染策略环境模块 202，要求渲染策略环境模块 202 创建对应的原子条件类实例，渲染策略环境模块 202 则返回一个继承于 CQPAAtomCondition 抽象类的原子条件类实例。CQPAAtomCondition 类是一个含有虚方法 BOOL Fit() 的抽象类，其事先由规则管理模块 200 定义，渲染策略环境模块 202 根据脚本文件中表示原子条件的脚本创建出对应于该脚本的以 CQPAAtomCondition 为基类的原子条件类实例。例如，规则管理模块 202 传送代码“<Mode Operator=“Eq” Value=“RENDERMODE_CUSTOM” />”至渲染策略环境模块 202，要求渲染策略环境模块 202 创建对应的原子条件类实例，渲染策略环境模块 202 则返回 CQPHDVideoFormat 类，其为继承于 CQPAAtomCondition 抽象类的原子条件类实例。

在规则管理模块 200 解析操作节点时，会将表示原子操作 XML 节点传给渲染策略环境模块 202，要求渲染策略环境模块 202 创建对应的原子操作，渲染策略环境模块 202 则返回继承于 CQPAAtomAction 抽象类的原子操作类实例。CQPAAtomAction 类是一个含有虚方法 BOOL DoAction() 的抽象类，其事先由规则管理模块 200 定义，渲染策略环境模块 202 根据脚本文件中表示原子操作的脚本创建出对应于该脚本的以

CQPAAtomAction 为基类的各种原子条件类实例。例如,规则管理模块 200 传送代码 “<DXVA Value= “True” />” 给渲染策略环境模块 202, 要求渲染策略环境模块 202 创建对应的原子操作类实例, 渲染策略环境模块 202 则返回 CQPAAtomActionDXVA 类, 其为继承于 CQPAAtomAction 抽象类的原子条件类实例。

基于上述方法原理,规则管理模块 200 持有的各种原子条件类实例、原子操作类实例,都可以用抽象类 CQPAAtomCondition 和 CQPAAtomAction 的类指针进行表示。规则管理模块 200 判断条件是否满足时, 仅需调用抽象类的类指针即可。

需要说明的是, 根据上面描述, 可以知道, 由于规则脚本中含有依赖于业务环境的原子条件, 原子操作, 因此向具有解析创建规则的规则管理模块 200 提供一个用于播放器渲染策略业务环境是必要的。这里, 播放器渲染策略业务环境主要的功能是:

(1) 解析规则管理模块 200 传过来的表示原子条件或者原子操作的 XML 节点, 并创建出对应于该节点的原子条件类实例或原子操作类实例, 返回给规则管理模块 200。

(2) 向规则管理模块 200 提供业务能力, 当规则管理模块 200 在做规则的条件判断时, 播放器渲染策略业务环境提供给规则管理模块 200 判断原子条件真与假所需的业务 API。当规则的条件满足时, 播放器渲染策略业务环境向规则管理模块 200 提供执行操作时所需的业务 API。

下边通过代码具体举例说明:

```
<Rule ID="Rule_Set_DXVA_When_RENDERMODE_CUSTOM">  
  <ConditionGroup Operator="Null">  
    <Mode Operator="Eq" Value="RENDERMODE_CUSTOM"/>
```

```
</ConditionGroup>  
<Action>  
    <DXVA Value="True"/>  
</Action>
```

5 </Rule>。

以上对本发明实施例提供的方法进行了描述。

在上述媒体文件播放系统及方法中，可以将渲染规则库 10 置于网络上的某些运营机器上，在网络上更新新的渲染规则，并同步到播放器客户端，则无需频繁修改播放器版本。另外，针对特定的用户，还可以
10 根据用户的机器环境、播放习惯及要播放的特殊媒体文件，定做一套渲染规则，根据该渲染规则建立最适合媒体文件的渲染线，从而得到较好的媒体播放效果，极大地满足了用户的体验需求。

以上所述实施例仅表达了本发明的几种实施方式，其描述较为具体和详细，但并不能因此而理解为对本发明专利范围的限制。应当指出的是，对于本领域的普通技术人员来说，在不脱离本发明构思的前提下，
15 还可以做出若干变形和改进，这些都属于本发明的保护范围。因此，本发明的保护范围应以所附权利要求为准。

权利要求书

1、一种媒体文件播放系统，其特征在于，所述系统包括渲染规则库与播放器单元，所述渲染规则库中存储了渲染规则，所述播放器单元包括：规则管理模块、渲染策略环境模块和播放模块，其中，

5 所述规则管理模块用于从所述渲染规则库中获取渲染规则，并对获取的渲染规则进行解析；

 所述渲染策略环境模块，用于根据所述规则管理模块解析出的渲染规则生成渲染线；

 所述播放模块，用于利用所述渲染线播放媒体文件。

10 2、根据权利要求1所述的媒体文件播放系统，其特征在于，所述规则管理模块进一步获取渲染规则中的条件节点和操作节点，将所述条件节点和操作节点中的描述传送给所述渲染策略环境模块识别。

 3、根据权利要求2所述的媒体文件播放系统，其特征在于，所述渲染策略环境模块包括：

15 条件生成模块，获取所述条件节点中原子条件的描述，对原子条件的描述进行解析并创建原子条件类实例；

 操作生成模块，获取所述操作节点中的原子操作的描述，对所述原子操作的描述进行解析并创建原子操作类实例；

20 操作执行模块，根据所述规则管理模块发送的条件成立的通知，执行条件生成模块创建的条件成立时操作生成模块所创建的对应的操作。

 4、根据权利要求3所述的媒体文件播放系统，其特征在于，所述渲染策略环境模块还包括：

 条件判断模块，用于获取媒体文件信息和机器参数，根据所述媒体文件信息和机器参数判断所述原子条件的真假值，并发送给所述规则管

理模块;

所述规则管理模块包括:

条件二叉树生成模块, 根据所述渲染规则中的条件节点生成条件二叉树;

- 5 条件二叉树计算模块, 根据所述条件判断模块返回的原子条件的真假值, 递归计算所述条件二叉树的根节点的真假值, 在计算出的根节点的真假值为真时, 发送条件成立的通知给所述操作执行模块。

5、根据权利要求 1 所述的媒体文件播放系统, 其特征在于, 所述渲染规则库存在播放器程序代码中, 或采用脚本文件存于播放器的安装目
10 录下, 或位于网络上的集中的数据库中。

6、一种媒体文件播放方法, 其特征在于, 所述方法包括以下步骤:

A, 从预先建立的渲染规则库中获取渲染规则, 并对获取的渲染规则进行解析;

B, 根据解析出的渲染规则生成渲染线;

15 C, 应用所述渲染线播放媒体文件。

7、根据权利要求 6 所述的媒体文件播放方法, 其特征在于, 所述步骤 B 包括:

B1, 获取所述渲染规则中所有条件节点的原子条件的描述, 调用渲染策略环境解析所述原子条件的描述并创建原子条件类实例;

20 B2, 获取所述渲染规则中所有操作节点的原子操作的描述, 调用渲染策略环境解析所述原子操作的描述并创建原子操作类实例;

B3, 在接收的所述渲染规则中条件成立的通知时, 根据步骤 B1 和步骤 B2 执行该条件对应的操作, 并建立合适的渲染线。

8、根据权利要求 7 所述的媒体文件播放方法, 其特征在于, 所述步骤 B2 中调用渲染策略环境解析原子操作的描述并创建原子操作类实例
25

包括:

判断所述操作节点下是否具有原子操作, 如果有, 调用渲染策略环境解析原子操作的描述并创建原子操作类实例。

- 9、根据权利要求 7 所述的媒体文件播放方法, 其特征在于, 所述步骤 B3 之前进一步包括:

获取媒体文件信息和机器参数, 根据所述媒体文件信息和机器参数判断所述原子条件的真假值;

所述步骤 B3 中渲染规则中条件成立的通知通过以下操作确定:

根据所述渲染规则中的条件节点生成条件二叉树;

- 10 根据所述原子条件的真假值, 递归计算所述条件二叉树的根节点的真假值, 在所述根节点的真假值为真时, 发送所述渲染规则中条件成立的通知。

10、根据权利要求 9 所述的媒体文件播放方法, 其特征在于, 所述根据原子条件的真假值, 递归计算条件二叉树的根节点的真假值包括:

- 15 调用渲染策略环境计算所述条件二叉树上左子节点所对应的原子条件的真假值;

判断所述条件二叉树上右子节点是否是原子条件, 如果是, 调用渲染策略环境计算该右子节点所对应的原子条件的真假值; 如果否, 递归计算以该右子节点为根节点的树所表示的条件的真假值;

- 20 根据条件二叉树上左右两个子节点的逻辑关系对左右子节点的真假值进行逻辑运算, 得到条件二叉树的根节点的真假值。