

Media File Playing System and Media File Playing Method

Technical Field

The invention relates to media playing technologies, and more particularly, to a system and method for playing media files.

Background of the Invention

At present, a DirectShow technical framework is generally adopted when playing various media files in a Window platform. A module called Filter Graph may be employed by the DirectShow technical framework to manage processing procedure of the whole multimedia data stream. Each functional module participating in data processing may be referred to as Filter. Each Filter may be connected in the Filter Graph according to a certain sequence to form an “assembly line” to work in collaboration. The “assembly line” is generally referred to as rendering line. Filter may be divided into three categories approximately according to different functions. The three categories may be Source Filter, Transform Filter and Rendering Filter. The Source Filter is mainly responsible for obtaining data, and transmitting the obtained data to the next Filter. The Transform Filter is mainly responsible for transforming data format, such as data stream separation/synthesis, encoding/decoding, etc, and continuously transmitting the data to the next Filter. The Rendering Filter is mainly responsible for the final destination of the data, e.g., sending the data to sound card and video card for multimedia presentation.

Filter includes a structure known as Pin. One kind is output Pin, the other kind is input Pin. A Filter is generally formed by one or more Pins. Filters may be connected with each other via a Pin. When a Filter is to be connected to a next Filter, the output Pin of the Filter needs to negotiate with the input Pin of the next Filter. When media streaming format of the output Pin may be supported by the input Pin of the next Filter, a connection may be successfully established between the two Filters.

Before playing a media file, a player may obtain and connect all the known Filters in a certain order, to obtain a rendering line, so as to achieve the objective of playing media. The above process is referred to as establishing a rendering line. Each

Filter has an attribute referred to as priority. When establishing a rendering line, the DirectShow may connect a Filter with higher priority to the rendering line by default. However, the Filter with higher priority may not connect successfully. Since the input Pin of the Filter with higher priority may not support media streaming format outputted by an output Pin of a Filter, which has already been connected to the rendering line. Therefore, to achieve the objective of playing media, an appropriate rendering line needs to be established.

At present, the following two modes are generally adopted in various players to establish rendering line and play media.

(1) Directly call an interface method provided by the DirectShow, RenderFile (strFilePath). The interface method depends on a default logic for establishing a rendering line, which is built within the technical framework of the DirectShow, so as to establish a rendering line for a media with a whole path of strFilePath. This mode is the simplest rendering strategy. In the mode, a player may not take into account what is each Filter in a rendering line and sequence of each Filter. The above method is effective to media, which may be processed by the default logic for establishing a rendering line built within the DirectShow. However, the DirectShow may not be enabled to intelligently establish an appropriate rendering line for media with the RenderFile (strFilePath), so as to play the media, resulted from local error generated during transmission, or generated by different compression methods, different compression tools of many media. Thus, playing effect may be poor.

For example, when playing a media file with video stream format of H264, there are two decoders CL264 Decoder 8 and MPCHC in a system, both of which support H264 video stream decoding (that is, both of which may be connected to the rendering line). According to the default logic for establishing a rendering line built within the technical framework of the DirectShow, priority of the CL264 Decoder 8 is higher than that of the MPCHC. And then, CL264 Decoder8 may be connected to the rendering line firstly. While, the MPCHC decoder may have no chance to connect to the rendering line. In fact, when executing High Definition (HD) acceleration for playing media file with video card, because certain characteristics of the file itself (for example, media file resolution, container format, and so on), decoding effect of CL264 Decoder8 may be poor, such as blurred screen. More serious, the video stream

decoded by the CL264 Decoder 8 may not be displayed on a screen, thus, phenomenon such as black screen may occur. On the contrary, decoding effect of the MPCHC may be better. However, the default logic for establishing a rendering line built within the DirectShow doesn't give a chance to the MPCHC. Thus, when adopting the default logic for establishing a rendering line built within the DirectShow, characteristics of a media file itself are not taken into consideration. Appropriate rendering line cannot be established for various kinds of media. Thus, playing effect of media file may be poor.

(2) Preset priority of each Filter in player code, according to experience of player author, so as to enable a player to implement better playing of media, which cannot be processed by the default logic for establishing a rendering line built within the DirectShow or processing effect of which is poor. This method may overcome deficiencies of the first method. This method is as follows. Regarding different media formats, set priority for each Filter with differences. For example, regarding H264 video, set priority of MPCHC decoder higher than that of the CL264 Decoder 8. Regarding the mpeg2 video, set priority of CL264 Decoder 8 higher than that of the MPCHC decoder.

However, the above method only partially takes characteristics of a media file into account, which still possesses the following deficiencies. A, poor universal property, regarding media files with a same format, appropriate Filter thereof may be different. Poor playing effect of media may occur, resulted from a rendering line established according to preset Filter priority. For example, regarding H264 video, a better decoding effect may be achieved, when some media files adopting CL264 Decoder 8 to decode instead of adopting MPCHC to decode. While, regarding some other media files with H264 video format, a better decoding effect may be achieved when adopting the MPCHC decoder to decode instead of adopting the CL264 Decoder 8 to decode. In view of this situation, a player author must test and analyze these media files, modify or add programs to cope with how to set Filter priority when playing these media, and then send out the modified new version, so as to solve the problem of poor playing effect when playing these media on a user machine. B, frequent modifications, because of different environment of different user machines (such as different acceleration capabilities of video card), a player author is

impossible to learn in advance. Therefore, a player version after testing and modifying may still not be able to solve problems of all the users. Thus, to solve the problem of poor playing effect, player version needs to be frequently modified, because of requirements for player put forward by different user machines.

Summary of the Invention

Embodiments of the invention provide a system and method for playing a media file, to establish a rendering line applicable to various media files, so as to achieve a better media playing effect.

The technical solution provided by an embodiment of the invention is as follows.

A system for playing a media file, wherein the system includes a rendering rule library and a player unit, the rendering rule library stores rendering rules, the player unit includes a rule managing module, a rendering strategy environment module and a playing module,

the rule managing module is to obtain a rendering rule from the rendering rule library, and analyze the rendering rule obtained;

the rendering strategy environment module is to generate a rendering line, according to the rendering rule analyzed by the rule managing module; and

the playing module is to play the media file utilizing the rendering line.

A method for playing a media file, including:

A, obtaining a rendering rule from a rendering rule library previously established, analyzing the rendering rule obtained;

B, generating a rendering line, according to the rendering rule analyzed; and

C, playing the media file by applying the rendering line.

From above technical solution, it can be seen that in the above system and method for playing a media file, rendering rules may be established and centralized stored in a rendering rule library. When playing a media file, a player may traverse rendering rules in the rendering rule library, execute actions defined in a rule

according to a logical causality defined in a rendering rule, to achieve the objective of adjusting rendering line and establishing a rendering line applicable to the media file. Thus, better media playing effect may be obtained.

Brief Descriptions of the Drawings

FIG.1 is a schematic diagram illustrating structure of a system for playing a media file in accordance with an embodiment of the invention.

FIG.2 is a schematic diagram illustrating structure of a rendering strategy environment module in accordance with an embodiment of the invention.

FIG.3 is a schematic diagram illustrating structure of a rule managing module in accordance with an embodiment of the invention.

FIG.4 is a flowchart illustrating a method for playing a media file in accordance with an embodiment of the invention.

FIG.5 is a flowchart illustrating a method for generating a rendering rule in accordance with an embodiment of the invention.

FIG.6 is a flowchart illustrating a method for analyzing a rendering rule in accordance with an embodiment of the invention.

FIG.7 is a flowchart illustrating a method for analyzing a condition node in accordance with an embodiment of the invention.

FIG.8 is a flowchart illustrating a method for analyzing an action node in accordance with an embodiment of the invention.

FIG.9 is a schematic diagram illustrating a condition binary tree in accordance with an embodiment of the invention.

FIG.10 is a flowchart illustrating a method for computing true and false value for a root node in a condition binary tree in accordance with an embodiment of the invention.

FIG.11 is a schematic diagram illustrating how to establish a ServiceAtomCondition class instance and a ServiceAtomAction class instance in accordance with an embodiment of the invention.

Embodiments of the Invention

In the embodiments of the invention, various establishment logics of rendering line used when playing media may be unified and abstracted as a rendering rule module, based on which a rendering rule library may be established. Thus, a rendering line applicable to various media files may be established based on the rendering rule library, so as to achieve better media playing effect.

To make objectives, technical solutions and advantages of the invention more clear, detailed descriptions of the invention are given in the following accompanying with attached figures and specific embodiments.

With reference to FIG.1, FIG.1 shows a system for playing media files in accordance with an embodiment of the invention. The system includes a rendering rule library 10 and a player unit 20.

The rendering rule library 10 stores rendering rules in advance. A rendering rule may define to execute which actions under each condition. The condition refers to certain factors affecting playing effect, which includes the follows. (1) Factors of a media file itself, such as media resolution, media container format, video stream format, audio stream format, and so on. (2) Limitation of Filter itself, for example, a Filter announces to support a certain video or audio stream format during connection negotiation with Pin, however, actual support effect is poor. (3) Software environment of operating system, such as operating system, and so on. For example, when executing HD acceleration in Windows XP system, a rendering Filter generally adopts Video Mixing Renderer 9 (VMR9) to provide support. However, in a Vista or Win7 system, an Enhanced Video Render (EVR) rendering filter needs to be adopted. (4) Machine hardware capabilities, such as video card, which mainly demonstrate that whether video card may support HD acceleration, when supporting the HD acceleration, the maximum number of reference frames which may be buffered by the video card, and so on. (5) Playing mode, that is, a user would like a player to adopt which mode to play media files. For example, a user considers whether playing

occupies Central Processing Unit (CPU) resources, if the user wants to do other things during the process of playing media files, the player may adopt an intelligent acceleration mode to minimize occupation of CPU resources, and so on. The condition may be any of or arbitrary combination of each factor above, which is not limited by embodiments of the invention. Correspondingly, actions include at least one of or arbitrary combination of setting HD acceleration or adjusting Filter priority, which corresponds to the condition. It can be seen that, a rendering rule actually describes a corresponding relationship between condition and operation.

In an implementation mode, data describing rendering rules in the rendering rule library 10 is static data, which may be stored in a group of hard-coded data structure of player program code. Alternatively, the static data may be stored in a script file in the installation directory of the player, or may be located in a certain centralized database of network, which is not limited in the embodiment of the invention. Preferably, it needs to ensure logic separation between the static data and specific program, which makes it easy to modify and manage.

The player unit 20 includes a rule managing module 200, a rendering strategy environment module 202 and a playing module 204.

The rule managing module 200 is configured to obtain and analyze rendering rules. The rule managing module 200 may obtain the rendering rules from the rendering rule library. The process of the rule managing module 200 analyzing a rendering rule may be as follows. Obtain a logical causality from the rendering rule, and convert format of the obtained logical causality into a suitable format, such as a format which may be identified by a computer.

The rendering strategy environment module 202 may generate a rendering line, according to the rendering rule analyzed by the rule managing module 200.

The playing module 204 is configured to utilize the rendering line to play a media file.

Here, the rule managing module 200 may only obtain the logical causality from the rendering rule by analyzing the rendering rule. The logical causality actually is a corresponding relationship between condition and action. The condition in the logical

causality may include various conditions, such as ServiceAtomCondition and Sub-Condition Group. The ServiceAtomCondition is a keyword, which may only be identified by service environment. The ServiceAtomCondition demonstrates a condition, which cannot be further sub-divided. While, Sub-Condition Group may be further sub-divided, e.g., be further sub-divided into ServiceAtomCondition. Based on this, the condition in the logical causality may be referred to as condition node for short. Correspondingly, the action in the logical causality may be referred to as operation node for short. The action node here consists of some ServiceAtomActions. A ServiceAtomAction is a keyword, which may only be identified by service environment. The ServiceAtomAction demonstrates an operation, which cannot be further sub-divided. However, the rule managing module 200 cannot identify specific meanings of ServiceAtomCondition in a Condition node and meanings of ServiceAtomAction in an action node. In an implementation mode, the rule managing module 200 may call an application program interface provided by the rendering strategy environment module 202 in advance, send the ServiceAtomCondition and ServiceAtomAction, which are respectively obtained from the condition node and action node to the rendering strategy environment module 202 for identifying.

Based on this, FIG.2 illustrates structure of the rendering strategy environment module 202 in accordance with an embodiment of the invention. As shown in FIG.2, the rendering strategy environment module 202 includes a condition generating module 2020, a condition determining module 2022, an action generating module 2026 and an action executing module 2028.

The condition generating module 2020 obtains and analyzes descriptions about ServiceAtomCondition in a condition node, creates a ServiceAtomCondition class instance, and returns the ServiceAtomCondition class instance to the rule managing module 200.

The condition determining module 2022 obtains media file information, determines true and false value for the ServiceAtomCondition according to the media file information, returns the true and false value of the ServiceAtomCondition to the rule managing module 200. For example, when the ServiceAtomCondition is “video stream format of current media file is H264”, it can be seen that the video stream format of current media file is H264 according to the media file information. Based on

this, it can be seen that the true and false value of the ServiceAtomCondition determined by the condition determining module 2022 is “true”.

The action generating module 2024 obtains and analyzes descriptions about ServiceAtomAction in an action node, creates a ServiceAtomAction class instance, and returns the ServiceAtomAction class instance to the rule managing module 200.

After receiving a notification indicating establishment of a condition node in the rendering rule, which is sent by the rule managing module 200, the action executing module 2026 executes actions corresponding to the established condition in the rendering rule.

Since a rendering line demonstrates a corresponding relationship between condition and executed action, when the action executing module 2026 executes actions, it is obvious that the corresponding relationship between condition and executed action may be obtained. That is, a corresponding rendering line may be generated. Correspondingly, when the action executing module 2026 doesn't execute actions, that is, the condition and action don't correspond to each other, a corresponding rendering line may also be generated.

Based on the structure of the rendering strategy environment module 202 shown in FIG.2, an embodiment of the invention also provides structure of rule managing module 200, which is shown in FIG.3. In FIG.3, the rule managing module 200 includes a condition binary tree generating module 2000 and a condition binary tree computing module 2002.

The condition binary tree generating module 2000 may generate a condition binary tree, according to condition nodes in the rendering rules.

Since one condition node in the rendering rule may include multiple ServiceAtomConditions, which may be formed by performing logic combination on ServiceAtomCondition included therein. The condition binary tree demonstrates ServiceAtomCondition included in each condition node, and a logic relationship among each ServiceAtomCondition.

The condition binary tree computing module 2002 recursively computes true and false value for root node of the condition binary tree, according to the true and false

value of the ServiceAtomCondition returned by the condition determining module 2022. Here, the true and false value of root node in the condition binary tree determines whether the condition node in the rendering rule is established. When the true and false value of root node in the condition binary tree computed by the condition binary tree computing module 2002 is true, a notification indicating that the condition node in the rendering rule is established will be sent to the above action executing module 2026, to facilitate the action executing module 2026 to execute corresponding actions.

In the foregoing, descriptions are given for a system provided by an embodiment of the invention. In the following, a method provided by the invention will be described accompanying with attached figures and embodiments.

FIG.4 is a flowchart illustrating a method for playing a media file in accordance with an embodiment of the invention. The specific procedure is as follows.

In block S10, obtain a rendering rule.

Here, in block S10, the rendering rule may be obtained from a rendering rule library. As described in the foregoing, the obtained rendering rule may define to execute which actions under which condition. The condition here is a factor which affects playing effect.

In block S20, analyze the rendering rule.

Specifically, block S20 may be as follows. Obtain a logical causality from the rendering rule by analyzing the rendering rule, and convert format of the obtained logical causality into a suitable format, such as a format identified by a computer.

In block S30, generate a rendering line according to the analyzed rendering rule.

In block S40, play a media file by applying the rendering line.

Specific operations of above block S30 may refer to the flow shown in FIG.5.

FIG.5 is a flowchart illustrating a method for generating a rendering line in accordance with an embodiment of the invention. The specific procedure is as follows.

In block S200, obtain descriptions about ServiceAtomCondition in all the condition nodes of the rendering rule, call the rendering strategy environment module 202 to analyze the descriptions about the ServiceAtomCondition, and establish a ServiceAtomCondition class instance.

In a rendering rule, one condition node may include multiple ServiceAtomConditions. Regarding each ServiceAtomCondition, it is necessary to call the rendering strategy environment module 202 to analyze the descriptions about the ServiceAtomCondition, establish a ServiceAtomCondition class instance, and return the ServiceAtomCondition class instance to the rule managing module 200.

In block S202, obtain media file information and machine parameters. Determine the true and false value for a ServiceAtomCondition, according to the media file information and the machine parameters.

In an implementation mode, call an application program interface provided by the rendering strategy environment module 202 to compute the true and false value of the ServiceAtomCondition. Here, when the analyzed media file information or machine parameters of the ServiceAtomCondition fits, determine the true and false value of the ServiceAtomCondition is “true”; otherwise, determine the true and false value of the ServiceAtomCondition is “false”. Return the determined true and false value of the ServiceAtomCondition to the rule managing module 200.

It should be noted that, above media file information includes characteristics of a media file itself (such as media file format, resolution, etc). The machine parameters include operating system software environment and hardware parameters of a user machine, and so on.

In block S204, generate a condition binary tree according to condition nodes in the rendering rule.

As mentioned above, the condition binary tree denotes all the ServiceAtomConditions in a condition node and logical relationship among each ServiceAtomCondition.

In block S206, recursively compute the true and false value of root node in the condition binary tree, according to the true and false value of ServiceAtomCondition obtained after computing.

In an implementation mode, the rule managing module 200 calls the application program interface provided by the rendering strategy environment module 202, to compute the true and false value of the ServiceAtomCondition, performs recursively computation on nodes in the condition binary tree, according to the true and false value of the ServiceAtomCondition, so as to obtain the true and false value of the root node.

In block S208, obtain descriptions about ServiceAtomAction in an action node of the rendering rule, call the rendering strategy environment module 202 to analyze the descriptions about the ServiceAtomAction, and establish a ServiceAtomAction class instance.

In a rendering rule, an action node may include multiple ServiceAtomActions. The ServiceAtomAction depends on service environment. Thus, it is necessary to call the rendering strategy environment module 202 to analyze descriptions about the ServiceAtomAction, create a ServiceAtomAction class instance, and return the ServiceAtomAction class instance to the rule managing module 200.

In block S210, execute actions corresponding to established conditions, according to computation result of the true and false value of the root node in the condition binary tree.

In an implementation mode, the true and false value of root node in the condition binary tree obtained after computing is “true”, it means that the condition node in the rendering rule is established. And then, the rendering strategy environment module 202 executes actions in the condition node, that is, actions corresponding to block S208, e.g., set HD acceleration, adjust Filter priority, etc.

It can be seen that, in the embodiment, the true and false value of root node in the condition binary tree determines whether the condition in the rendering rule is established or not. Since a rendering rule defines actions executed under various conditions, when a rendering rule is satisfied, the action executing module 2026

executes actions defined in the rendering rule, so as to establish a rendering line applicable to media. Thus, better media playing effect may be obtained, when the playing unit 20 plays.

Thus, a rendering line may be generated by executing blocks shown in FIG.5. To make technical solutions illustrated in FIG.4 and FIG.5 of the application clearer, detailed descriptions are provided in the following with a specific embodiment.

In the embodiment, an Extensible Markup Language (XML) is adopted to describe a rendering rule and generate a script file. The structure format of the script file is as follows.

```
<Root>

  <Rule ID="ID1" >

    <ConditionGroup Operator="OP1">

      <ServiceAtomCondition1/>

      <ServiceAtomCondition2/>

      ...

      <ServiceAtomCondition3/>

      ...

    <ConditionGroup Operator="OP2">

      ....

    </ConditionGroup>

  </ConditionGroup>

  <Action>

    <ServiceAtomAction1/>

    <ServiceAtomAction2/>
```

```

    </Action>

</Rule>

....

<Rule ID="IDn">

....

</Rule>

</Root>

```

In the above structure format of the script file, Root includes various rendering rules, which is the root node of the script file. Rule node denotes a rendering rule, that is, descriptions about to execute which action under which condition. ID is a unique identification number in the rendering rule. ConditionGroup node denotes conditions in the rendering rule. Each rendering rule must include a ConditionGroup node. While, each ConditionGroup node may include sub-ConditionGroup, and various ServiceAtomConditions (such as ServiceAtomCondition 1, ServiceAtomCondition 2 in above structure, and so on). These ServiceAtomConditions possess keywords, which may be identified by the rendering strategy environment module 202. For example, ServiceAtomCondition 1 may be a condition, which is used for determining whether video stream format of current movie is H264, the format thereof is as follows.

```

<HDVideoFormat Operato="Eq" Value="H264"/>

```

The ServiceAtomCondition must rely on the rendering strategy environment module 202 to be identified. When video stream of current media obtained by the rendering strategy environment module 202 is H264, the condition determining module 2022 determines the ServiceAtomCondition is true.

It should be noted that, Operator attribute value in the ConditionGroup node may be defined by the rule managing module 200. The attribute value may be some logical relationships, such as and, or. When Operator attribute value is “and”, it means that the relationship among each condition demonstrated by each child node in the

ConditionGroup node is “and”. In other words, when each condition denoted by each child node in the ConditionGroup node is true, the ConditionGroup node is true; otherwise, the ConditionGroup node is false. When Operator attribute value is “or”, it means that relationship among each condition denoted by each child node in the ConditionGroup node is “or”. That is, when there is a condition denoted by one of all the child nodes in the ConditionGroup node is true, the ConditionGroup node is true; otherwise, the ConditionGroup node is false.

Furthermore, it should be noted that, the Operator attribute value of a child node in the ConditionGroup node should not be same as the Operator attribute value of the ConditionGroup node (referred to as parent node). Otherwise, the child node may be simply put into the parent node, which may be denoted by the following codes.

```
<ConditionGroup Operator="And">

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

<ConditionGroup Operator="And">

<HDAudioFormat Operator="Eq" Value="EAC3"/>

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

</ConditionGroup>

</ConditionGroup>
```

denote that

```
if ( ( HDVideoFormat == H264 )

&& ( (HDAudioFormat == EAC3) && (System == WinVista) )

)
```

From above codes, it can be seen that, the Operator attribute value of child node is same as the Operator attribute value of the parent node, both of which are “And”. Based on this, the above codes may be simplified as follows.

```

    If( ( HDVideoFormat == H264 ) && ( HDAudioFormat == EAC3 ) &&
    ( System == WinVista )
    )

```

which may be denoted as follows with script

```

<ConditionGroup Operator="And">

<HDVideoFormat Operator="Eq" Value="H264"/>video stream format is H264

<HDAudioFormat Operator="Eq" Value="EAC3"/>audio stream format is
EAC3

<System Operator="Eq" Value="WinVista"/>system is WinVista system

</ConditionGroup>。

```

Action node (operation node) denotes actions necessary to be executed, when return value of a condition in ConditionGroup node of a top level of rendering rule is “true”. The Action node may also be constituted by performing logical combination on some ServiceAtomActions (such as ServiceAtomAction 1, ServiceAtomAction 2 in above structure). The ServiceAtomAction also possesses a keyword, which may be identified by the rendering strategy environment module 202. For example, demonstrate mode of ServiceAtomAction 1 is as follows.

```

<SetTransformFilterPriority Name="WMVideo Decoder DMO"

Priority="PRIORITY_PREFERRED"/>

```

The descriptions may be identified by the rendering strategy environment module 202. The action is to adjust priority of a decoder named “WMVideo Decoder DMO” to the highest priority.

FIG.6 is a flowchart illustrating a method for analyzing a rendering rule in accordance with an embodiment of the invention. The specific process is as follows.

In block S60, obtain current rule node from a rendering rule script.

Here, the rendering rule script includes multiple rule nodes (that is, structure of above script file includes multiple rule nodes). Thus, it is necessary to traverse all the rule nodes in the script file.

In block S62, determine whether there is next rule node of current rule node in the rendering rule script, if yes, proceed to block S64; otherwise, end the procedure.

In block S64, locate to root ConditionGroup node of current rule node.

In block S66, analyze current ConditionGroup node, generate a ServiceAtomCondition class instance, and insert the ServiceAtomCondition class instance into existed condition binary tree.

In block S68, analyze action nodes of rule node. Take the next rule node of current rule node as current rule node, and return to block S60.

Specifically, block S66 may be shown in Fig.7. FIG.7 is a flowchart illustrating a method for analyzing a condition node in accordance with an embodiment. The specific process is as follows.

In block S660, determine current sub-condition node from the condition node, that is, the ConditionGroup node.

Here, the first sub-condition node in the condition node, namely, ConditionGroup node, is generally taken as current sub-condition node.

In block S662, determine whether current sub-condition node is ConditionGroup node, if yes, proceed to block S668; otherwise, proceed to block S664.

Here, the ConditionGroup node is a composite node, which may be subdivided into atomic nodes.

In block S664, send current sub-condition node to the rendering strategy environment module 202 for analyzing, and generate a ServiceAtomCondition class instance.

In the embodiment, when the sub-condition node is not the ConditionGroup node, it means that there is no ServiceAtomCondition node of the sub-condition node. And

then, descriptions about the sub-condition node may be sent to the rendering strategy service environment 202 for analyzing.

In block S666, insert the ServiceAtomCondition class instance into the condition binary tree, and then, proceed to block S669.

In block S668, recursively analyze current sub-condition node, generate a ServiceAtomCondition class instance, and insert the ServiceAtomCondition class instance into the existed condition binary tree, and then, proceed to block S669.

In an embodiment, when current sub-condition node is ConditionGroup node, it also means that there may be ServiceAtomCondition nodes in the ConditionGroup node. Thus, it is necessary to recursively analyze current ConditionGroup node, until obtaining the ServiceAtomCondition node finally.

In block S669, determine whether current sub-condition node is condition node, that is, the last sub-condition node of the ConditionGroup node. If yes, end the procedure. Otherwise, take the next sub-condition node of current sub-condition node as current sub-condition node, and return to execute block S660.

In the embodiment, the analyzing actions illustrated by block S68 may be shown in FIG.8. FIG.8 is a flowchart illustrating a method for analyzing an action node in accordance with an embodiment of the invention. The specific process is as follows.

In block S680, locate to an action node of current rule node.

In block S682, obtain next ServiceAtomAction node from current action node.

In block S684, determine whether there is a ServiceAtomAction node, if yes, proceed to block S686; otherwise, end the procedure.

In block S686, send the ServiceAtomAction node to the rendering strategy environment module 202 for analyzing, and generate a ServiceAtomAction class instance.

In the embodiment, regarding the ServiceAtomCondition node in ConditionGroup of a script file and ServiceAtomAction node in the Action node, since they have semantics associated with service environment, while the rule

managing module 200 may only process logical relationships among ServiceAtomConditions, and the problem to execute which actions when a condition is established. Thus, the rule managing module 200 needs to send the ServiceAtomCondition and XML nodes of ServiceAtomActions to the rendering strategy environment module 202 for analyzing, creating corresponding ServiceAtomCondition class instance and ServiceAtomAction class instance, returning the ServiceAtomCondition class instance and ServiceAtomAction class instance to the rule managing module 200.

In the embodiment, the rule managing module 200 adopts the condition binary tree to denote condition nodes in script rules. For example, in a script file, code of the ConditionGroup node is as follows.

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

Denote the ConditionGroup node with a condition binary tree, which is shown in FIG.9. In the condition binary tree, “&&” denotes a logical relationship “and”, “||” denotes a logical relationship “or”. The condition binary tree starts traversal from the first ServiceAtomCondition node of root ConditionGroup node in the rendering rule. The root node of the condition binary tree is Operator attribute value of the root ConditionGroup node (as the root “&&” in FIG.9), which is used for denoting a logical relationship between left and right ServiceAtomCondition nodes of the

condition binary tree. In the condition binary tree, the left child node of each node is a ServiceAtomCondition node associated with service environment. When the ServiceAtomCondition node denoted by the left child node only has one ServiceAtomCondition subsequently in the script, the right child node is the subsequent ServiceAtomCondition node. When the ServiceAtomCondition denoted by the left child node in the script is a sub-ConditionGroup node subsequently, the right child node is Operator attribute value of the sub-ConditionGroup node. When the ServiceAtomCondition denoted by the left child node in the script has at least two sub-ConditionGroup nodes subsequently, the right child node is Operator attribute value of current ConditionGroup node.

FIG.10 is a flowchart illustrating a method for computing true and false value of a root node in a condition binary tree, according to an embodiment of the invention. The specific process is as follows.

In block S100, call the rendering strategy environment module 202 to compute the true and false value of a ServiceAtomCondition corresponding to the left child node.

In block S102, determine whether the right child node is a ServiceAtomCondition, if yes, proceed to block S104; otherwise, proceed to block S106.

In block S104, call the rendering strategy environment module 202 to compute the true and false value of ServiceAtomCondition corresponding to the right child node.

In block S106, recursively call subsequent traversal algorithm to compute the true and false value of a condition, which is denoted by a tree, the root node of which is the right child node.

In the embodiment, when the right child node is not the ServiceAtomCondition, the right child node is a root node of a logical relationship. And then, it is necessary to recursively compute the true and false value of a condition, denoted by a tree, the root node of which is the right child node.

In block S108, perform a logical computation on the true and false values of the left and right child nodes, according to the logical relationship denoted by each node, to obtain the true and false value of root node of the condition binary tree.

In the embodiment, when the true and false value of the root node in the condition binary tree is “true”, it means that condition in the rendering rule is established. When the true and false value of the root node in the condition binary tree is “false”, it means that the condition in the rendering rule is not established.

FIG.11 is a schematic diagram illustrating how to create a ServiceAtomCondition class instance and a ServiceAtomAction class instance, in accordance with an embodiment of the invention.

As shown in FIG.11, when analyzing a rendering rule, the rule managing module 200 may send an XML node denoting a ServiceAtomCondition to the rendering strategy environment module 202, to require the rendering strategy environment module 202 to create a corresponding ServiceAtomCondition class instance. The rendering strategy environment module 202 may then return a ServiceAtomCondition class instance, inherited from CQPAtomCondition abstract class. CQPAtomCondition class is an abstract class including a virtual method BOOL Fit(), which is defined by the rule managing module 200 in advance. The rendering strategy environment module 202 creates a ServiceAtomCondition class instance corresponding to a script, according to the script denoting the ServiceAtomCondition in the script file. The base class of the created ServiceAtomCondition class instance is CQPAtomCondition. For example, the rule managing module 202 sends code “<Mode Operator= “Eq” Value= “RENDERMODE_CUSTOM” />” to the rendering strategy environment module 202, to require the rendering strategy environment module 202 to create corresponding ServiceAtomCondition class instance. And then, the rendering strategy environment module 202 may return CQPHDVideoFormat class, which is a ServiceAtomCondition class instance inherited from CQPAtomCondition abstract class.

When analyzing an action node, the rule managing module 200 may send an XML node denoting a ServiceAtomAction to the rendering strategy environment module 202, to require the rendering strategy environment module 202 to create

corresponding ServiceAtomAction. And then, the rendering strategy environment module 202 may return a ServiceAtomAction class instance, which is inherited from CQPAtomAction abstract class. The CQPAtomAction class is an abstract class including a virtual method BOOL DoAction(), which is defined by the rule managing module 200 in advance. The rendering strategy environment module 202 creates various ServiceAtomCondition class instances corresponding to script, the base class of which is CQPAtomAction, according to the script denoting ServiceAtomAction in a script file. For example, the rule managing module 200 sends code “<DXVA Value= “True” />” to the rendering strategy environment module 202, to require the rendering strategy environment module 202 to create corresponding ServiceAtomAction class instance. And then, the rendering strategy environment module 202 may return CQPAtomActionDXVA class, which is a ServiceAtomCondition class instance inherited from CQPAtomAction abstract class.

Based on above method principle, each kind of ServiceAtomCondition class instance and ServiceAtomAction class instance possessed by the rule managing module 200, may be denoted by class pointer of abstract classes CQPAtomCondition and CQPAtomAction. When determining whether a condition is satisfied, the rule managing module 200 may only call class pointer of abstract class.

It should be noted that, based on above descriptions, it can be seen that since a rule script includes ServiceAtomCondition and ServiceAtomAction, both of which dependent on service environment, it is necessary to provide the rule managing module 200, which is configured to analyze creation rule, with rendering strategy service environment used for player. Here, functions of rendering strategy service environment of a player are as follows.

(1) Analyze an XML node denoting a ServiceAtomCondition or ServiceAtomAction sent by the rule managing module 200, create a ServiceAtomCondition class instance or a ServiceAtomAction class instance corresponding to the node, and return the ServiceAtomCondition class instance or the ServiceAtomAction class instance to the rule managing module 200.

(2) Provide service capabilities for the rule managing module 200, when the rule managing module 200 makes condition determination about rules, the rendering

strategy service environment of a player provides the rule managing module 200 with a service Application Programming Interface (API), which is necessary for determining true and false of a ServiceAtomCondition. When a rule condition is satisfied, the rendering strategy service environment of the player provides the rule managing module 200 with service API necessary for executing actions.

In the following, descriptions are provided with a specific example accompanying with codes.

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

In the foregoing, descriptions are given for the method provided by an embodiment of the invention.

In the foregoing system and method for playing media files, the rendering rule library 10 may be deployed on some operational machines of a network. New rendering rules may be updated in a network, and synchronized to player client. Thus, it is not necessary to frequently modify player version. In addition, regarding a specified user, a set of rendering rules may be made according to order, according to the user's machine environment, playing habits and a special media file to be played. And then, a rendering line best applicable to the media file may be established according to the rendering rule. Thus, better media playing effect may be obtained, and user's experience requirements may be greatly satisfied.

The foregoing embodiments only express several implementation modes of the invention, the descriptions thereof are relatively specific and detailed. However, they should not be understood as limitation to patent scope of the invention. It should be pointed out that, regarding a person having ordinary skill in the art, several deformation and improvements may be made without departing from conception of the invention, which are covered by the protection scope of the invention. Therefore, the patent protection scope of the invention should be determined by the attached claims.

Claims

1, A system for playing a media file, wherein the system comprises a rendering rule library and a player unit, the rendering rule library stores rendering rules, the player unit comprises a rule managing module, a rendering strategy environment module and a playing module,

the rule managing module is to obtain a rendering rule from the rendering rule library, and analyze the rendering rule obtained;

the rendering strategy environment module is to generate a rendering line, according to the rendering rule analyzed by the rule managing module; and

the playing module is to play the media file utilizing the rendering line.

2, The system according to claim 1, wherein the rule managing module is further to obtain a condition node and an action node from the rendering rule, send descriptions about the condition node and descriptions about the action node to the rendering strategy environment module for identifying.

3, The system according to claim 2, wherein the rendering strategy environment module comprises:

a condition generating module, which is to obtain descriptions about a ServiceAtomCondition in the condition node, analyze the descriptions about the ServiceAtomCondition and create a ServiceAtomCondition class instance;

an action generating module, which is to obtain descriptions about a ServiceAtomAction in the action node, analyze the descriptions about the ServiceAtomAction, and create a ServiceAtomAction class instance; and

an action executing module, which is to execute actions created by the action generating module corresponding to an established condition, according to a notification about the established condition sent by the rule managing module, wherein the condition is created by the condition generating module.

4, The system according to claim 3, wherein the rendering strategy environment module further comprises:

a condition determining module, which is to obtain media file information and machine parameters, determine true and false value for the ServiceAtomCondition according to the media file information and the machine parameters, and send the true and false value to the rule managing module;

the rule managing module comprises:

a condition binary tree generating module, which is to generate a condition binary tree according to the condition node in the rendering rule; and

a condition binary tree computing module, which is to recursively compute the true and false value of root node in the condition binary tree, according to the true and false value of the ServiceAtomCondition returned by the condition determining module, when the computed true and false value of the root node is true, send the notification about the established condition to the action executing module.

5, The system according to claim 1, wherein the rendering rule library is stored in player program code, or in an installation catalog of player with a script file, or located in a centralized database of network.

6, A method for playing a media file, comprising:

A, obtaining a rendering rule from a rendering rule library previously established, analyzing the rendering rule obtained;

B, generating a rendering line, according to the rendering rule analyzed; and

C, playing the media file by applying the rendering line.

7, The method according to claim 6, wherein step B comprises:

B1, obtaining descriptions about ServiceAtomCondition of all the condition nodes in the rendering rule, calling rendering strategy environment to analyze the descriptions about the ServiceAtomCondition and create a ServiceAtomCondition class instance;

B2, obtaining descriptions about ServiceAtomAction of all the action nodes in the rendering rule, calling the rendering strategy environment to analyze the

descriptions about the ServiceAtomAction and create a ServiceAtomAction class instance;

B3, after receiving a notification about established condition in the rendering rule, executing actions corresponding to the condition in blocks B1 and B2, and creating an applicable rendering line.

8, The method according to claim 7, wherein calling the rendering strategy environment to analyze the descriptions about the ServiceAtomAction and create the ServiceAtomAction class instance in block B2 comprises:

determining whether there is a ServiceAtomAction in the action node, if yes, calling the rendering strategy environment to analyze the descriptions about the ServiceAtomAction and create the ServiceAtomAction class instance.

9, The method according to claim 7, before block B3, further comprising:

obtaining media file information and machine parameter, determining true and false value of the ServiceAtomCondition according to the media file information and the machine parameter;

wherein the notification about the established condition in the rendering rule in block B3 is determined by implementing the following steps:

generating a condition binary tree according to condition nodes in the rendering rule;

recursively computing the true and false value of root node in the condition binary tree, according to the true and false value of the ServiceAtomCondition; when the true and false value of the root node is true, sending the notification about the established condition in the rendering rule.

10, The method according to claim 9, wherein recursively computing the true and false value of the root node in the condition binary tree, according to the true and false value of the ServiceAtomCondition comprises:

calling the rendering strategy environment to compute the true and false value of the ServiceAtomCondition corresponding to a left child node in the condition binary tree;

determining whether a right child node in the condition binary tree is the ServiceAtomCondition, if yes, calling the rendering strategy environment to compute the true and false value of the ServiceAtomCondition corresponding to the right child node; otherwise, recursively computing the true and false value of a condition denoted by a tree, the root node of which is the right child node;

performing a logical computation on the true and false values of the left and right child nodes, according to a logical relationship between the left and right child nodes in the condition binary tree, to obtain the true and false value of the root node in the condition binary tree.