Method for muting audio through pluggable components of a media pipeline

Disclosed is a method for muting audio through pluggable components of a media pipeline. Benefits include improved functionality and improved performance.

Background

Conventionally, audio muting support is provided by several components, including the following (see Figure 1):

- Pipeline processor
- Audio render
- Audio driver
- Audio digital-to-analog converter (DAC)

Audio is muted by sending special MUTE_AUDIO events down the media stream or by invocation of a special application program interface. The MUTE_AUDIO event must be sent downstream to the audio renderer, which translates the MUTE_AUDIO command for the audio driver. It ramps down the power level slowly to the actual audio hardware, such as the digital-to-audio converter and speakers. However, this procedure can be prone to failures. For example, if the audio driver does not power the speakers down slowly, an audible pop may occur when the speaker suddenly loses power. Events can be lost or a delay can be introduced when the MUTE_AUDIO event is sent in place of special buffers.

Trick-mode processing controls the pause, fast forward, and rewind functions. Conventional approaches do not support trick-mode processing when a pipeline processor is not ready or does not provide ways to mute audio with a pluggable element.

Description

The disclosed method mutes audio through pluggable components of a media pipeline. The method improves the reliability of media trick modes processing by using special audio buffers. Muting support is moved to pluggable elements, including a splitter and audio decoder.

The key elements of the disclosed method include:

- Media pipeline
- Splitter component
- Audio decoder component
- Special audio buffers

The disclosed method emulates normal playback processing during mode processing by sending special audio buffers from the splitter component to the audio decoder component. Additionally, buffers filled with zeroes are sent to the audio renderer. The special buffers can contain current timestamp data and other information that is not transferred in conventional buffers.

The splitter component transmits the special and conventional buffers (see Figure 2).

The audio decoder creates the conventional buffers and fills them with zeroes, taking into account the received current timestamp and the current sample rate. This approach does not require audio clock resetting during trick-mode processing (see Figure 3).

The disclosed method enables media trick-mode processing when normal media playback is supported by a pipeline processor.

Advantages

The disclosed method provides advantages, including:

- Improved functionality due to providing audio muting using pluggable components of a media pipeline
- Improved functionality due to enabling trick-mode processing when normal media playback is supported by a pipeline processor
- Improved functionality due to providing special buffers for current timestamp data and other information that is not transferred in conventional buffers
- Improved performance due to avoiding audio clock resetting during trick-mode processing

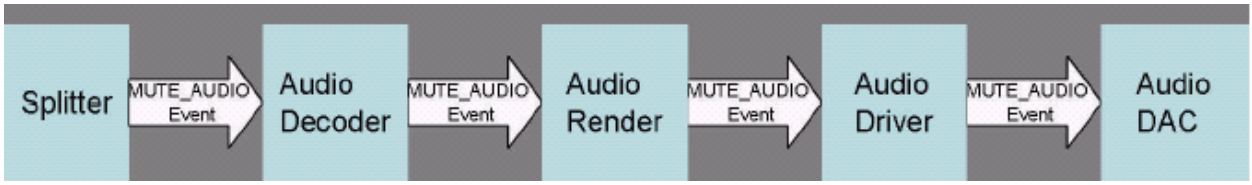

Fig. 1

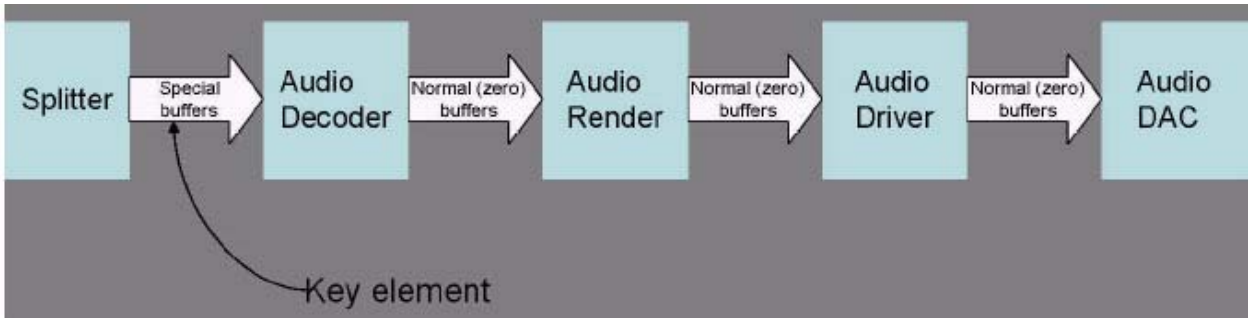

Fig. 2

```
…
if (is_audio_muted())
 send_audio_special(timestamp_current);
else
 send_audio_normal(buffer_audio);
…
```
Fig. 3

```
…
if (is_special_buffer(buffer_received))
 send_zero_buffer(buffer_get_timestamp(buffer_received));
else
 process_normal(buffer_received);
…
```
Fig. 4

Disclosed anonymously