Audio mixing in Java ME

This article explains how to use a simple MIDlet for testing audio mixing support in Nokia devices. It enables testing of simultaneous playback of two MP3 and two WAV files.

29 Sep 2013

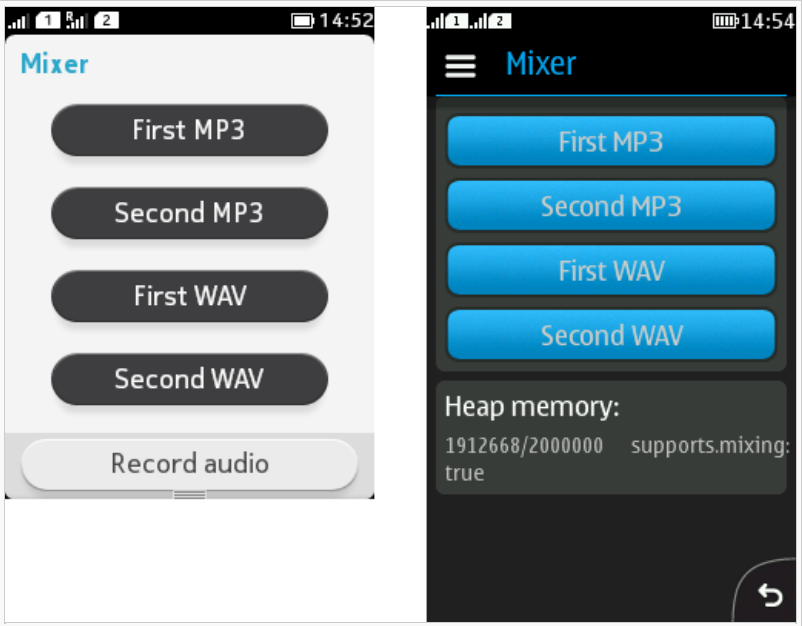
Warning: Audio mixing is not supported in Nokia Asha software platform at time of writing (Nokia Asha Platform v1.0).

Introduction

Audio mixing enables simultaneously playing of background music and sound effects - an important feature especially in games. Audio mixing has been supported in Series 40 since Series 40 5th Edition but is not supported in Nokia Asha software platform 1.0.

This article introduces a MIDlet, which can be used for testing the audio mixing capabilities in a Java ME enabled device. The MIDlet suite consists of two MIDlets, Mixer and Mixer2. Both MIDlets have four buttons for playing two MP3 files and two WAV files. The Mixer MIDlet plays the files in continuous loop. Pressing the buttons starts and stops the playback of the file. There is also a possibility to try simultaneous recording. Currently it is not possible to record audio while audio is played back in Nokia Series 40 and Nokia Asha devices.

The Mixer2 MIDlet can also play four simultaneous audio files. The difference to the Mixer MIDlet is that MP3 files are played back in continuous loop, but WAV files only once. In addition to that, the latency time is measured for WAV audio playback start, from the button press to the "started" event from `PlayerListener`.



Mixer MIDlet in Nokia 501 and Nokia 311 devices

Checking whether mixing is supported

Whether or not audio mixing is supported depends on the device features, like MMAPI (JSR-135) implementation, processor speed and heap memory amount on the device. It is possible to check if audio mixing is supported using MMAPI system property `supports.mixing`.

```
System.out.println("supports.mixing: " + System.getProperty("supports.mixing"))
```

Playing audio in MIDlets

Playing audio in MIDlets is done by using MMAPI (JSR-135) and its `Player` class. For effective use of MMAPI we recommend you read the MMAPI specification and the `Player` life cycle.

The following code shows how an MP3 file can be played:

(C) Copyright Nokia 2014. All rights reserved.

```
/**
 * Plays back the MP3 file in the selected player in continuous loop.
 * @param file the MP3 file to be played back
 * @param player the Player to be used for the playback
 */
private void playMP3(String file, Player player) {
    try {
        InputStream is = getClass().getResourceAsStream(file);
        if (player == player1) {
            if (player1 == null) player1 = Manager.createPlayer(is, "audio/mp3");
            player1.setLoopCount(-1);
            player1.start();
        }
        else if (player == player2) {
            if (player2 == null) player2 = Manager.createPlayer(is, "audio/mp3");
            player2.setLoopCount(-1);
            player2.start();
        }
    } catch (IOException ioe) {
        form.append("playMP3(): IOException: " + ioe.getMessage());
    } catch (IllegalStateException ise) {
        form.append("playMP3(): IllegalStateException: " + ise.getMessage());
    } catch (MediaException me) {
        form.append("playMP3(): MediaException: " + me.getMessage());
    } catch (OutOfMemoryError oome) {
        form.append("playMP3(): OutOfMemoryError: " + oome.getMessage());
    }
}
```

Audio mixing in MIDlets is simply done by creating separate `Player`s for each sound and starting the players simultaneously. If audio mixing is not supported, commonly only the last started player is audible. For example, in games with background music and sound effects the background music is paused, while a sound effect is played.

Measuring the latency by using PlayerListener

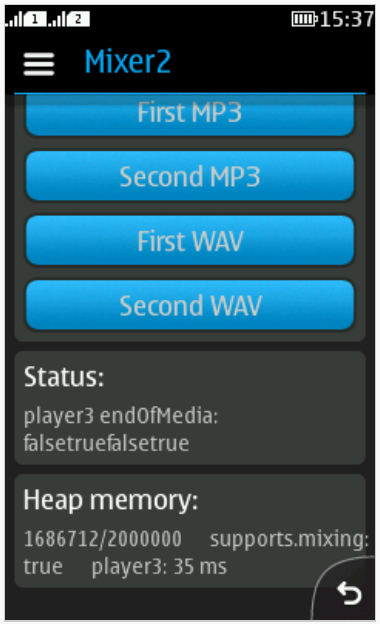
As said above, Mixer2 measures the latency between the button presses and the "started" event from `PlayerListener`. To be exact, the time counter starts just before the `player.start()` method call, as shown below in the code. The start time is measured by using `System.currentTimeMillis()`. The end time is measured in `playerUpdate()` method, when the occurred event is "started".

```
/**
 * Plays back the WAV file in the selected player.
 * @param file the WAV file to be played back
 * @param player the Player to be used for the playback
 */
private void playWav(String file, Player player) {
    try {
        InputStream is = getClass().getResourceAsStream(file);
        if (player == player3) {
            if (player3 == null) {
                player3 = Manager.createPlayer(is, "audio/wav");
                player3.prefetch();
                player3.addPlayerListener(this);
            }
            start3 = System.currentTimeMillis();
```


(C) Copyright Nokia 2014. All rights reserved.

```
            player3.start();
        }
        else if (player == player4) {
            if (player4 == null) {
                player4 = Manager.createPlayer(is, "audio/wav");
                player4.prefetch();
                player4.addPlayerListener(this);
            }
            start4 = System.currentTimeMillis();
            player4.start();
        }
    } catch (IOException ioe) {
        form.append("playWav(): IOException: " + ioe.getMessage());
    } catch (MediaException me) {
        form.append("playWav(): MediaException: " + me.getMessage());
    } catch (OutOfMemoryError oome) {
        form.append("playWav(): OutOfMemoryError: " + oome.getMessage());
    }
}

/**
 * @see javax.microedition.media.PlayerListener#playerUpdate(Player, String, Object)
 */
public void playerUpdate(Player player, String event, Object eventData) {
    if (player.equals(player3)) {
        if (event.equals("started")) {
            form.append("player3: " + (System.currentTimeMillis() - start3));
        }
...
}
```



Mixer2 MIDlet showing 35 ms latency time for a "sound effect" in the Nokia SDK 2.0 for Java emulator

Note: Testing audio mixing support should be done in real devices, not only in emulators. In general, audio (and video) support in emulators might differ from the support in real devices.

Example application

(C) Copyright Nokia 2014. All rights reserved.

File:Mixer.zip containing sources, **Mixer.jad** and **Mixer.jar**

(C) Copyright Nokia 2014. All rights reserved.