MIDlet volume control in Series 40 and S60 devices

Overview

In general, in MIDlets volume is controlled by using MMAPI's VolumeControl and its `getLevel()` and `setLevel()` methods.

For example:

```
VolumeControl vc = (VolumeControl)player.getControl("VolumeControl");
if (vc != null) vc.setLevel(100);
```

Devices might have their own volume keys for controlling the volume level. Java implementations in older Nokia phones don't support changing volume by using these keys, but latest models do. This is supported in S60 3rd Edition FP2 (and newer) and Series 40 5th Edition FP1 devices (and newer).

Details about controlling volume

Some more details about volume control support in S60 3rd Edition FP2 and in Series 40 5th Ed. FP1:

- If the phone model has external volume keys present, then those keys can be used to control the volume of MMAPI Players.
- The keys work in this way when a MIDlet that uses MMAPI is on foreground. They change the volume level for MMAPI Players globally, that is, the volume level of every Player in every MIDlet is altered.
- In Series 40 5th Ed. FP1 devices a visible indicator is shown to the user, when volume is adjusted by using external volume keys.
- In general the volume changes made by using `VolumeControl.setLevel()` method is indicated by using `VOLUME_CHANGED` event
- In S60 3rd Edition FP2 devices (or newer) the volume level change made by using volume keys is not visible to the user, but there is a Nokia proprietary event is indicating the volume change. More information can be found from here: http://library.developer.nokia.com/topic/Java_Developers_Library/GUID-7332888D-1C7E-4FF5-973E-1EE0ACF5F95B.html.
- In S60 3rd Edition FP2 devices the volume changes made by external volume keys are not indicated by using PlayerListener.VOLUME_CHANGED event, but by using "com.nokia.external.volume.Event" event instead.
- It is not possible to get any key events from these external volume keys.
- In S60 3rd Edition FP2 (and newer) the volume level set using external volume keys stays effective even if the phone is restarted.
- In S60 3rd Edition FP1 (and older) devices VolumeControl.getLevel() initially returns value 100 if the volume level of the player has not been set using VolumeControl.setLevel() method.
- In S60 3rd Edition FP2 devices VolumeControl.getLevel() initially returns the default volume level (decided by the underlying implementation), if player volume level has not been set using VolumeControl.setLevel().
- Note that in S60 devices sounds played by a MIDlet are not heard, if profile setting "Warning sounds" is set off. This setting can be found (in N95 8GB) from: Menu -> Tools -> Profiles -> General -> Personalise -> Warning tones.
- Note, that volume keys control global volume and MMAPI VolumeControl controls MIDlet's own volume. The actual volume that is heard by the user is the global volume in proportion to the Volume Control. For example:

  If the global volume scale is 1-10 and it is set to value 6. If the level of VolumeControl is 100 (=max) then the output is 6. If the level of VolumeControl is 50 then the output is 3. So there can be situations, where MIDlet volume is set to maximum level (100), but still nothing is heard. That situation happens when the global volume value is 0.

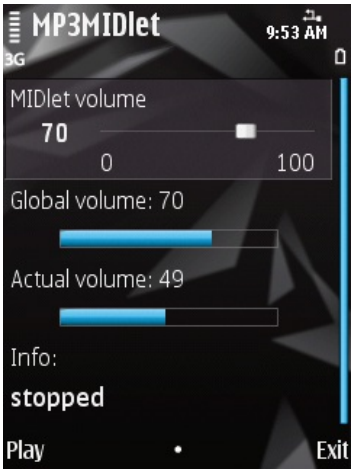
MP3MIDlet

The MP3MIDlet demonstrates, how VolumeControl and changing volume in S60 3rd Edition FP2 (and newer) devices by using external volume keys work. It uses LCDUI Gauges for showing the volume levels. There are three gauges on the MIDlet's Form:

- midletVolumeGauge: showing VolumeControl's volume (values between 0 and 100), controlled by using VolumeControl.setLevel() method
  - this Gauge is interactive and it can be used for changing the midlet volume level
- globalVolumeGauge: showing device's volume (values between 0 and 100), controlled by using the external volume keys
- actualVolumeGauge: showing "actual" volume, which is calculated by using the following formula: globalVolume/100 * midletVolume

The image below shows the MIDlet screen in Nokia N78 device.

(C) Copyright Nokia 2013. All rights reserved.



Source code: MP3MIDlet.java

```
import javax.microedition.lcdui.Item;
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Gauge;
import javax.microedition.lcdui.ItemStateListener;
import javax.microedition.lcdui.StringItem;
import java.io.IOException;
import java.io.InputStream;
import javax.microedition.media.Manager;
import javax.microedition.media.MediaException;
import javax.microedition.media.Player;
import javax.microedition.media.PlayerListener;
import javax.microedition.media.control.VolumeControl;

public class MP3MIDlet extends MIDlet implements CommandListener, ItemStateListener , PlayerListener {
    private Form form;
    private Command playCommand;
    private Command stopCommand;
    private Command exitCommand;
    private Player p;
    private VolumeControl vc;
    private Gauge midletVolumeGauge;
    private Gauge globalVolumeGauge;
    private Gauge actualVolumeGauge;
    private String midletVolumeGaugeLabel = "MIDlet volume";
    private String globalVolumeGaugeLabel = "Global volume";
    private String actualVolumeGaugeLabel = "Actual volume";
    private int actualVolume = 0;
    private int globalVolume = 0;
    private int midletVolume = 50;
    private StringItem stringItem;

    public void startApp() {
        if (form == null) { // Create the Form and Command only once
            form = new Form("MP3MIDlet");
            playCommand = new Command("Play", Command.SCREEN, 1);
            stopCommand = new Command("Stop", Command.STOP, 1);
            exitCommand = new Command("Exit", Command.EXIT, 1);
            form.addCommand(playCommand);
            form.addCommand(exitCommand);
            form.setCommandListener(this);
            midletVolumeGauge = new Gauge(midletVolumeGaugeLabel, true, 100, midletVolume);
            globalVolumeGauge = new Gauge(globalVolumeGaugeLabel, false, 100, globalVolume);
            actualVolumeGauge = new Gauge(actualVolumeGaugeLabel, false, 100, actualVolume);
            form.setItemStateListener(this);
            form.append(midletVolumeGauge);
            form.append(globalVolumeGauge);
            form.append(actualVolumeGauge);
            stringItem = new StringItem("Info:", "");
            form.append(stringItem);
        }
        Display.getDisplay(this).setCurrent(form);
    }

    public void pauseApp() { }

    public void destroyApp(boolean unconditional) { }

    public void commandAction(Command c, Displayable d) {
        if (c == playCommand) {
            while (form.size() > 4) form.delete(form.size()-1);
            form.removeCommand(playCommand);
            form.addCommand(stopCommand);
            playMP3();
        }
        else if (c == stopCommand) {
            form.removeCommand(stopCommand);
            form.addCommand(playCommand);
            stop();
        }
        else if (c == exitCommand) {
            stop();
            p = null;
            this.notifyDestroyed();
        }
    }
```


(C) Copyright Nokia 2013. All rights reserved.

```
    private void playMP3() {
        try {
            InputStream is = getClass().getResourceAsStream("/piano_32k.mp3");
            if (p == null) p = Manager.createPlayer(is, "audio/mp3");
            p.addPlayerListener(this);
            p.realize();
            vc = (VolumeControl)p.getControl("VolumeControl");
            p.start();
        } catch (IOException ioe) {
            this.showError("IOException", ioe.getMessage());
        } catch (MediaException me) {
            this.showError("MediaException", me.getMessage());
        }
    }

    private void stop() {
        if (p != null) {
            try {
                p.stop();
            } catch (MediaException me) {
                this.showError("MediaException", me.getMessage());
            }
        }
        form.removeCommand(stopCommand);
        form.addCommand(playCommand);
    }

    protected void showError(String title, String text) {
        System.out.println("showError: " + title);
        System.out.println("showError: " + text);
        Alert alert = new Alert(title, text, null, AlertType.ERROR);
        alert.setTimeout(Alert.FOREVER);
        alert.getType().playSound(Display.getDisplay(this));
        Displayable current = Display.getDisplay(this).getCurrent();
        if (current instanceof Alert) {}
        else Display.getDisplay(this).setCurrent(alert);
    }

    public void itemStateChanged(Item item) {
        if (item.equals(midletVolumeGauge)) {
            midletVolume = midletVolumeGauge.getValue();
            if (vc != null) vc.setLevel(midletVolume);
        }
    }

    public void playerUpdate(Player player, String event, Object eventData) {
        if (player.equals(p)) {
            stringItem.setText(event);
            if (event.equals("volumeChanged")) { // Gauge is adjusted
                actualVolume = (int)(((float)globalVolume/100) * (float)midletVolume);
            }
            else if (event.equals("com.nokia.external.volume.event")) {
                // External volumes keys are pressed
                globalVolume = Integer.parseInt(eventData.toString());
                globalVolumeGauge.setValue(globalVolume);
                globalVolumeGauge.setLabel(globalVolumeGaugeLabel + ": " + globalVolume);
  actualVolume = (int)(((float)globalVolume/100) * (float)midletVolume);
            }
            else if (event.equals("endOfMedia")) { // End of song reached
                form.removeCommand(stopCommand);
                form.addCommand(playCommand);
            }
            else if (event.equals("started")) { // Playback started

            }
            else if (event.equals("stopped")) { // Playback stopped

            }
            actualVolumeGauge.setValue(actualVolume);
            actualVolumeGauge.setLabel(actualVolumeGaugeLabel + ": " + actualVolume);
        }
    }
}
```

Example application

- MP3MIDlet.zip containing MP3MIDlet.jad and MP3MIDlet.jar

See also

- External volume buttons

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.