FORUM NOKIA

Camera MIDlet: A Mobile Media API Example

Version 1.0; January 24, 2003

Java

NOKIA

Contents

Change History

24 January 2003	V1.0	Document published in Forum Nokia.

Disclaimer

The information in this document is provided "as is," with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

The phone UI images shown in this document are for illustrative purposes and do not represent any real device.

Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc.

Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

Camera MIDlet: A Mobile Media API Example

Version 1.0; 24 Jan. 2003

1 Introduction

Beginning with Nokia 3650, some Nokia MIDP-enabled mobile phones implement the Mobile Media API **[MMAPI]**, which allows MIDlets to have more powerful use of media types. In particular, image capture via the device's camera may be supported.

The following guide describes the Camera MIDlet, a simple example program that makes use of this image capture capability. The Camera MIDlet is deliberately simple and hence useful only as a tutorial, so this guide provides a few ideas for how it could be extended to make useful applications.

The guide assumes that you are familiar with Java programming. It also assumes that you understand the basics of MIDP programming, perhaps by having read the Forum Nokia paper *Brief Introduction to MIDP Programming* **[MIDPPROG]**.

2 Image Capture Using the Mobile Media API

The Mobile Media API specification includes a code fragment ("Scenario 11: Camera") that describes how to capture an image using the device's camera. The Camera MIDlet uses the same technique.

First, a `Player` must be created, which takes its input from the camera:

```
Player player;
…
player = Manager.createPlayer("capture://video");
```

Then the player must be initialized:

```
player.realize();
```

You must then get the visible "video control," which will show the live viewfinder image:

```
VideoControl videoControl;
…
videoControl = (VideoControl)player.getControl("VideoControl");
```

The video control has two display modes: It can draw itself in a MIDP low-level UI Canvas object, or it can be added to a MIDP high-level UI Form as an Item. In this MIDlet, you will use the Canvas approach, but Section 5 shows an alternative implementation using a Form.

```
videoControl.initDisplayMode(VideoControl.USE_DIRECT_VIDEO, canvas);
```

Start the player, so that the user can see what the camera is pointing at:

```
player.start();
```

Once the video control is active, you can take a snapshot:

```
byte[] pngData = videoControl.getSnapshot(null);
```

The parameter for getSnapshot is a String stating which format the data should be in; if it is null, then the data will be in PNG format, which is guaranteed to be supported by all implementations.

This data can then be used in many ways. For instance it can be saved in a record store for later use, send over HTTP to a server, or displayed immediately as an Image. To do so, use Image's createImage(byte[], int, int) method:

```
Image photoImg = Image.createImage(pngData, 0, pngData.length);
```

Camera images and image data occupy relatively large amounts of memory; be sure to release that memory for garbage collection (e.g., by setting your reference to `null`) when it is no longer needed.

3 Camera MIDlet Design

3.1 User Interface Design

The Camera MIDlet has two screens between which it alternates:

Figure 1: Taking a photo

The first screen shows a running video "viewfinder" image of the scene in front of the camera. When the user decides that the image is properly framed (and the subject is smiling nicely), he or she selects the option "Capture" or simply clicks the button mapped to "game action" FIRE.

Figure 2: Displaying the photo taken

The second screen then shows the photo that was taken. To keep this example MIDlet simple, nothing else can be done with the photo (it can't be saved or sent anywhere).

3.2 Software Design

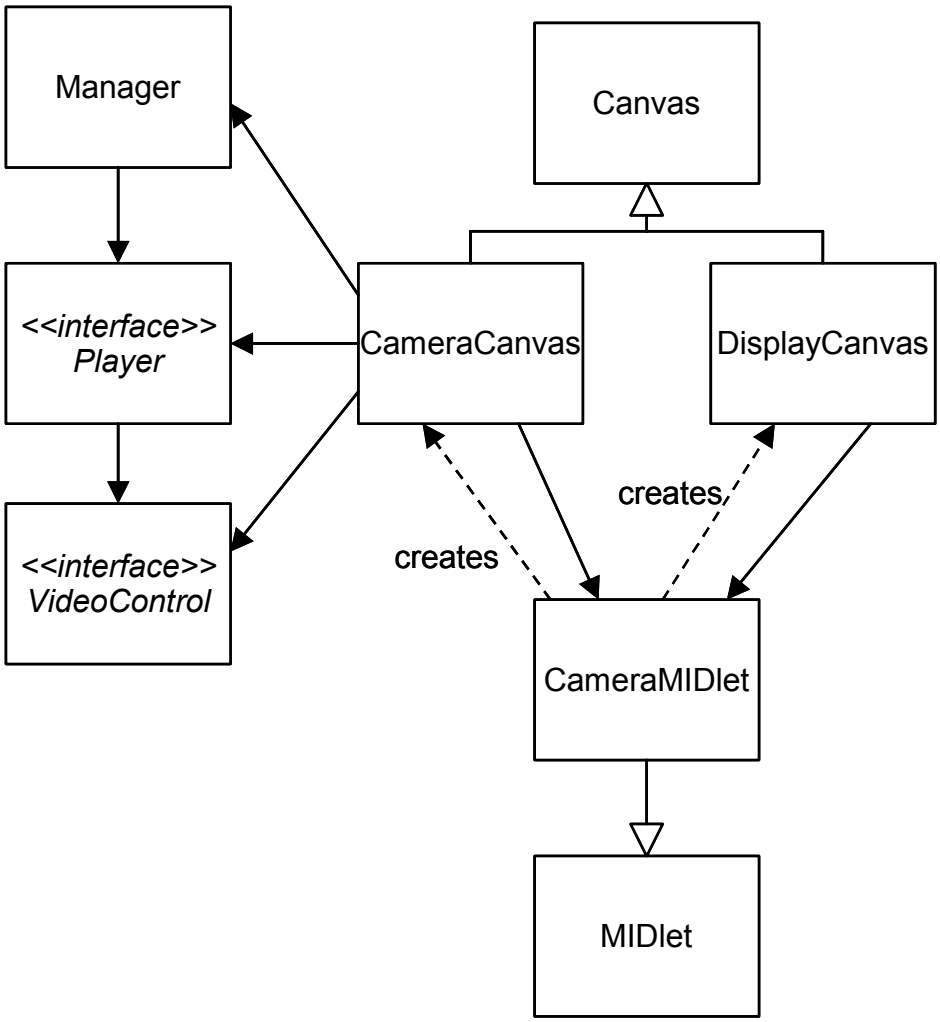

Figure 3: Camera MIDlet class diagram

The CameraMIDlet creates both the CameraCanvas and the DisplayCanvas. The CameraCanvas requests a video capture player from the MMAPI Manager class, and receives an object implementing the Player interface. From this object it requests a video control, and receives an object implementing the VideoControl interface. You won't know (or need to know) the actual class of either of these objects, as you will interact with them only through those interfaces.

4 Source Code

4.1 Class CameraMIDlet

Per usual in Forum Nokia's MIDlet examples, the MIDlet class controls which user interface screen is displayed, and the screens call back to it when they're ready for transitions to other screens – they don't contact the other screens directly.

Be careful to stop the CameraCanvas's video player when it isn't being displayed, or when the MIDlet has been told to pause.

```
// unnamed package

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;


public class CameraMIDlet
    extends MIDlet
{
    private CameraCanvas cameraCanvas = null;
    private DisplayCanvas displayCanvas = null;


    public CameraMIDlet()
    {
    }


    public void startApp()
    {
        Displayable current = Display.getDisplay(this).getCurrent();
        if (current == null)
        {
            // first call
            cameraCanvas = new CameraCanvas(this);
            displayCanvas = new DisplayCanvas(this);
            Display.getDisplay(this).setCurrent(cameraCanvas);
            cameraCanvas.start();
        }
        else
        {
            // returning from pauseApp
            if (current == cameraCanvas)
            {
                cameraCanvas.start();
            }
            Display.getDisplay(this).setCurrent(current);
        }
    }


    public void pauseApp()
    {
        if (Display.getDisplay(this).getCurrent() == cameraCanvas)
        {
            cameraCanvas.stop();
        }
    }
```

```
    public void destroyApp(boolean unconditional)
    {
        if (Display.getDisplay(this).getCurrent() == cameraCanvas)
        {
            cameraCanvas.stop();
        }
    }


    private void exitRequested()
    {
                destroyApp(false);
                notifyDestroyed();
    }


    void cameraCanvasExit()
    {
        exitRequested();
    }


    void cameraCanvasCaptured(byte[] pngData)
    {
        cameraCanvas.stop();
        displayCanvas.setImage(pngData);
        Display.getDisplay(this).setCurrent(displayCanvas);
    }


    void displayCanvasBack()
    {
        Display.getDisplay(this).setCurrent(cameraCanvas);
        cameraCanvas.start();
    }
}
```

4.2 Class CameraCanvas

The CameraCanvas class creates a video player and a video control from it, and tells the video control to display itself in the canvas. You'll notice from the `paint` method that the CameraCanvas object doesn't itself draw the video image: it just draws the screen background and the video control is automatically drawn on top. To clarify which screen is being displayed, the CameraCanvas uses a yellow background and the DisplayCanvas uses a cyan background.

The `takeSnapshot` method takes a snapshot and passes the image data, a byte array in PNG format, back to the MIDlet. The MIDlet will then pass it to the DisplayCanvas object for display.

```
// unnamed package

import javax.microedition.lcdui.*;
import javax.microedition.media.*;
import javax.microedition.media.control.*;
import java.io.IOException;


class CameraCanvas
    extends Canvas
    implements CommandListener
```

```
{
    private final CameraMIDlet midlet;
    private final Command exitCommand;
    private Command captureCommand = null;
    private Player player = null;
    private VideoControl videoControl = null;

    private boolean active = false;

    // Two strings for displaying error messages
    private String message1 = null;
    private String message2 = null;


    CameraCanvas(CameraMIDlet midlet)
    {
        this.midlet = midlet;

        exitCommand = new Command("Exit", Command.EXIT, 1);
        addCommand(exitCommand);
        setCommandListener(this);

        try
        {
            player = Manager.createPlayer("capture://video");
            player.realize();

            // Grab the video control and set it to the current display.
            videoControl = (VideoControl)(player.getControl("VideoControl"));
            if (videoControl == null)
            {
                discardPlayer();
                message1 = "Unsupported:";
                message2 = "Can't get video control";
            }
            else
            {
                videoControl.initDisplayMode(VideoControl.USE_DIRECT_VIDEO,
                                             this);
                // centre video, letting it be clipped if it's too big
                int canvasWidth = getWidth();
                int canvasHeight = getHeight();
                int displayWidth = videoControl.getDisplayWidth();
                int displayHeight = videoControl.getDisplayHeight();
                int x = (canvasWidth - displayWidth) / 2;
                int y = (canvasHeight - displayHeight) / 2;
                videoControl.setDisplayLocation(x, y);

                captureCommand = new Command("Capture", Command.SCREEN, 1);
                addCommand(captureCommand);
            }
        }
        catch (IOException ioe)
        {
            discardPlayer();
            message1 = "IOException:";
            message2 = ioe.getMessage();
        }
        catch (MediaException me)
        {
            discardPlayer();
```

```
            message1 = "MediaException:";
            message2 = me.getMessage();
        }
        catch (SecurityException se)
        {
            discardPlayer();
            message1 = "SecurityException";
            message2 = se.getMessage();
        }
    }


    // Called in case of exception to make sure invalid players are closed
    private void discardPlayer()
    {
        if (player != null)
        {
            player.close();
            player = null;
        }
        videoControl = null;
    }


    public void paint(Graphics g)
    {
        g.setColor(0x00FFFF00);   // yellow
        g.fillRect(0, 0, getWidth(), getHeight());

        if (message1 != null)
        {
            g.setColor(0x00000000);   // black
            g.drawString(message1, 1, 1, Graphics.TOP | Graphics.LEFT);
            g.drawString(message2, 1, 1 + g.getFont().getHeight(),
                                       Graphics.TOP | Graphics.LEFT);
        }
    }


    synchronized void start()
    {
        if ((player != null) && !active)
        {
            try
            {
                player.start();
                videoControl.setVisible(true);
            }
            catch (MediaException me)
            {
                message1 = "Media exception:";
                message2 = me.getMessage();
            }
            catch (SecurityException se)
            {
                message1 = "SecurityException";
                message2 = se.getMessage();
            }
            active = true;
        }
    }
```

```
    synchronized void stop()
    {
        if ((player != null) && active)
        {
            try
            {
                videoControl.setVisible(false);
                player.stop();
            }
            catch (MediaException me)
            {
                message1 = "MediaException:";
                message2 = me.getMessage();
            }
            active = false;
        }
    }


    public void commandAction(Command c, Displayable d)
    {
        if (c == exitCommand)
        {
            midlet.cameraCanvasExit();
        }
        else if (c == captureCommand)
        {
            takeSnapshot();
        }
    }


    public void keyPressed(int keyCode)
    {
        if (getGameAction(keyCode) == FIRE)
        {
            takeSnapshot();
        }
    }


    private void takeSnapshot()
    {
        if (player != null)
        {
            try
            {
                byte[] pngImage = videoControl.getSnapshot(null);
                midlet.cameraCanvasCaptured(pngImage);
            }
            catch (MediaException me)
            {
                message1 = "MediaException:";
                message2 = me.getMessage();
            }
        }
    }
}
```

4.3 Class DisplayCanvas

The DisplayCanvas class just creates an image from the PNG format byte array it is given and displays that image in the center of the screen.

```
// unnamed package

import javax.microedition.lcdui.*;


class DisplayCanvas
    extends Canvas
    implements CommandListener
{
    private final CameraMIDlet midlet;

    private Image image = null;


    DisplayCanvas(CameraMIDlet midlet)
    {
        this.midlet = midlet;

        addCommand(new Command("Back", Command.BACK, 1));
        setCommandListener(this);
    }


    public void paint(Graphics g)
    {
        g.setColor(0x0000FFFF);   // cyan
        g.fillRect(0, 0, getWidth(), getHeight());
        if (image != null)
        {
            g.drawImage(image,
                        getWidth()/2,
                        getHeight()/2,
                        Graphics.VCENTER | Graphics.HCENTER);
        }
    }


    void setImage(byte[] pngImage)
    {
        image = Image.createImage(pngImage, 0, pngImage.length);
    }


    public void commandAction(Command c, Displayable d)
    {
        midlet.displayCanvasBack();
    }
}
```

4.4 Application Descriptor (camera.jad)

The Camera MIDlet's application descriptor simply has the mandatory fields plus a description of the MIDlet.

```
MIDlet-Description: Camera MIDlet
MIDlet-Jar-Size: 4186
MIDlet-Jar-URL: camera.jar
MIDlet-Name: Camera
MIDlet-Vendor: Forum Nokia
MIDlet-Version: 0.4
```

4.5 Manifest

The manifest includes the mandatory fields plus a description and an icon (📷).

```
MIDlet-Name: Camera
MIDlet-Version: 0.4
MIDlet-Vendor: Forum Nokia
MIDlet-Description: Camera MIDlet
MIDlet-Icon: /camera.png
MicroEdition-Profile: MIDP-1.0
MicroEdition-Configuration: CLDC-1.0
MIDlet-1: Camera,/camera.png,CameraMIDlet
```

5 Using a Form with VideoControl

Class CameraCanvas is used to display the viewfinder image and take a picture. It is possible to do the same thing with a Form. An implementation of this, which can be used to replace CameraCanvas (with trivial modifications to CameraMIDlet), is shown below.

5.1 Class CameraForm

```
// unnamed package

import javax.microedition.lcdui.*;
import javax.microedition.media.*;
import javax.microedition.media.control.*;
import java.io.IOException;


class CameraForm
    extends Form
    implements CommandListener
{
    private final FormCameraMIDlet midlet;
    private final Command exitCommand;
    private Command captureCommand = null;
    private Player player;
    private VideoControl videoControl;

    private boolean active = false;
    private StringItem messageItem;


    CameraForm(FormCameraMIDlet midlet)
    {
        super("Camera");
        this.midlet = midlet;

        messageItem = new StringItem("Message", "start");
        append(messageItem);

        exitCommand = new Command("Exit", Command.EXIT, 1);
        addCommand(exitCommand);
        setCommandListener(this);

        // initialize camera
        try
        {
            player = Manager.createPlayer("capture://video");
            player.realize();

            // Grab the video control and set it to the current display.
            videoControl = (VideoControl)(player.getControl("VideoControl"));
            if (videoControl != null)
            {
                append((Item)(videoControl.initDisplayMode(
                                        VideoControl.USE_GUI_PRIMITIVE,
                                        null)));
                captureCommand = new Command("Capture", Command.SCREEN, 1);
```

```
                addCommand(captureCommand);
                messageItem.setText("OK");
            }
            else
            {
                messageItem.setText("No video control");
            }
        }
        catch (IOException ioe)
        {
            messageItem.setText("IOException: " + ioe.getMessage());
        }
        catch (MediaException me)
        {
            messageItem.setText("Media Exception" + me.getMessage());
        }
        catch (SecurityException se)
        {
            messageItem.setText("Security exception: " + se.getMessage());
        }
    }

    synchronized void start()
    {
        if (! active)
        {
            try
            {
                if (player != null)
                {
                    player.start();
                }
                if (videoControl != null)
                {
                    videoControl.setVisible(true);
                }
            }
            catch (MediaException me)
            {
                messageItem.setText("Media exception: " + me.getMessage());
            }
            catch (SecurityException se)
            {
                messageItem.setText("Security exception: " + se.getMessage());
            }
            active = true;
        }
    }

    synchronized void stop()
    {
        if (active)
        {
            try
            {
                if (videoControl != null)
                {
                    videoControl.setVisible(false);
                }
```

```
                if (player != null)
                {
                    player.stop();
                }
            }
            catch (MediaException me)
            {
                messageItem.setText("Media exception: " + me.getMessage());
            }
            active = false;
        }
    }


    public void commandAction(Command c, Displayable d)
    {
        if (c == exitCommand)
        {
            midlet.cameraFormExit();
        }
        else if (c == captureCommand)
        {
            // if we have a capture command, we know videoControl is not null
            try
            {
                byte[] pngImage = videoControl.getSnapshot(null);
                midlet.cameraFormCaptured(pngImage);
                messageItem.setText("OK");
            }
            catch (MediaException me)
            {
                messageItem.setText("Media exception: " + me.getMessage());
            }
        }
    }
}
```

6 Practical Uses of Camera Facility

By itself, this MIDlet is useful only as a programming example. In practice, here are some ideas with which to experiment.

6.1 Camera Notes

The MIDlet can be modified to save each photo in a record store. Then an extra screen can be implemented to browse the saved photos. This can be useful for taking photo notes, for example if an artist wants to record sights for drawing later.

6.2 Photo ”Blogging”

”Blogging” means maintaining a public Web diary (a web-log, or ”blog”). The MIDlet can be modified to optionally send the photo over HTTP to a Web server, which can then dynamically add the photo to the user’s Web log. The MIDlet can also allow the user to enter a text caption to be sent to the Web site with the photo.

7 References

MMAPI Mobile Media API (JSR-135) Specification
Version 1.0, 23 May 2002
Java Community Process
http://www.jcp.org

MIDPPROG Brief Introduction to MIDP Programming
Forum Nokia, 2002
http://www.forum.nokia.com