CMER

Centre for Mobile Education and Research

Graphics and Sound

Overview

- Images
- Raw Images
- Encoded Images
- Drawing and Rendering Images
- Creating a Media Player
- Working with Blackberry Media Player
- Rich Media
- Playing Rich Media Content

Images

- 2 types of images
  - RAW Images
  - Encoded Images

Raw Images

- Allow applications to use raw image data.
  - To retrieve raw image data from a specified region of a bitmap and store the data in an integer array, invoke Bitmap.getARGB().

    void getARGB(int[] argbData, int offset, int scanLength, int x, int y, int width, int height);
- Retrieve image data.
  - Initialize an integer array.
  - To store the raw image data of the new or predefined bitmap in the integer array, invoke Bitmap.getARGB().

Raw Images (Cont.)

- Retrieve image data.

```
Bitmap original =
    Bitmap.getPredefinedBitmap(Bitmap.INFORMATION);
int[] argb = new int[original.getWidth() * original.getHeight()];
original.getARGB(argb, 0, original.getWidth(), 0, 0,
    original.getWidth(), original.getHeight());
```

- Compare two images to see if they are identical.
  - Invoke Bitmap.equals().

```
if(restored.equals(original)) {
    System.out.println("Success! Bitmap renders correctly with
    RGB data.");
} else if(!restored.equals(original)) {
    System.out.println("Bitmap rendered incorrectly with RGB
    data.");
}
```

Encoded Images

- Access an image.
  - Save an image to the project folder or sub-folder.
  - Add the image to the project in the BlackBerry® Integrated Development Environment.
  - Invoke Class.getResourceAsStream() to retrieve the image as an input stream of bytes.

```
private InputStream input;
...
try {
      input =
         Class.forName("com.rim.samples.docs.imagedemo.ImageDemo").
      getResourceAsStream("/images/example.png");
} catch (ClassNotFoundException e) {
   System.out.println("Class not found");
}
```

Encoded Images (Cont.)

- Encode an image.
  - Invoke EncodedImage.createEncodedImage(). This method creates an instance of EncodedImage using the raw image data in the byte array.
  - Check for an IllegalArgumentException, which EncodedImage.createEncodedImage() throws if the byte array that you provide as a parameter does not contain a recognized image format.

```
private byte[] data = new byte[2430]; // Store the contents of the
    image file.
try {
    input.read(data);           // Read the image data into the byte array.
} catch (IOException e) {   // Handle exception.
}
try {
    EncodedImage image = EncodedImage.createEncodedImage(data,
    0, data.length);
} catch (IllegalArgumentException iae) {
    System.out.println("Image format not recognized.");
}
```

Encoded Images (Cont.)

- Display an encoded image.
  - To assign the encoded image to a BitmapField, invoke BitmapField.setImage().
  - To add the BitmapField to the screen, invoke add().

```
BitmapField field = new BitmapField();
field.setImage(image);
add(field);
```

- Set the decoding mode.
  - Invoke EncodedImage.setDecodeMode().
  - Provide one of the following modes as a parameter to the method:
    - DECODE_ALPHA: decodes an alpha channel, if one exists (this is the default mode)
    - DECODE_NATIVE: forces the BlackBerry® Java® Application to decode the bitmap to the native bitmap type of the handheld software application
    - DECODE_READONLY: marks the decoded bitmap as read-only
- Set the image display size.
  - Invoke EncodedImage.setScale().
  - The inverse of the integer specified by the scale parameter scales the image. For example, if you set the scaling factor to 2, the image decodes at 50% of its original size.

Drawing and Rendering Images

- Position an image
  - Use an individual field.
    - Invoke the Graphics() constructor.

```
Bitmap surface = new Bitmap(100, 100);
BitmapField surfaceField = new BitmapField(surface);
Graphics graphics = new Graphics(surface);
```

    - Add the BitmapField to the screen.

```
mainScreen.add(surfaceField);
```

  - Use the whole screen.
    - Invoke Screen.getGraphics().

```
Graphics graphics = Screen.getGraphics();
```

    - Make sure your methods perform their drawing functions within the boundaries of the screen.

```
graphics.fillRect(10, 10, 30, 30);
graphics.drawRect(15, 15, 30, 30);
```

Drawing and Rendering Images (Cont.)

- Draw an image in color
  - Determine whether device supports color display.
    - Invoke Graphics.isColor().
  - Determine the number of colors that the handheld supports.
    - Invoke Graphics.getNumColors().
  - Set the pixel transparency in the drawing area.
    - Invoke one of the following methods:
      - Graphics.setGlobalApha()
      - Graphics.getGlobalAlpha()
    - Define a global alpha value within the following range:
      - 0 (0x0000): completely transparent
      - 255 (0x00FF): fully opaque

Drawing and Rendering Images (Cont.)

- Draw an image in color
  - Determine raster operations that the Application supports.
    - Invoke Graphics.isRopSupported(int).
    - Provide one of the following constants as a parameter:
      - ROP_CONST_GLOBALALPHA: blends the constant foreground color using a constant global alpha value with destination pixels
      - ROP_SRC_GLOBALALPHA: blends a source bitmap using a constant global alpha value with destination pixels

Drawing and Rendering Images (Cont.)

- Draw an image in color
  - Draw a set of shaded, filled paths.
    - Invoke Graphics.drawShadedFilledPath():

```
public void drawShadedFilledPath(int[] xPts, int[] yPts,
    byte[] pointTypes, int[] colors, int[] offsets);
```

  - The following example draws a path that blends from blue to red:

```
Bitmap surface = new Bitmap(240, 160);
BitmapField surfaceField = new BitmapField(surface);
add(surfaceField);
Graphics graphics = new Graphics(surface);
int[] X_PTS = { 0, 0, 240, 240 };
int[] Y_PTS = { 20, 50, 50, 20 };
int[] drawColors = { 0x0000CC, 0x0000CC, 0xCC0000, 0xCC0000 };
try {
    graphics.drawShadedFilledPath(X_PTS, Y_PTS, null, drawColors,
    null);
} catch (IllegalArgumentException iae) {
    System.out.println("Bad arguments.");
}
```

Drawing and Rendering Images (Cont.)

- Draw an image in color
  - Turn a drawing style on.
    - Invoke Graphics.setDrawingStyle(int drawStyle, boolean on).
  - Turn a drawing style off.
    - Invoke Graphics.setDrawingStyle(int drawStyle, boolean off).
  - Determine if a drawing style is set.
    - Invoke Graphics.isDrawingStyleSet(int drawStyle).
  - Use a monochrome bitmap as a stamp.
    - The STAMP_MONOCHROME option enables the Applications to use monochrome bitmaps as stamps by rendering the nontransparent region in color. This option applies to bitmaps that are 1 bit and have alpha defined.

```
BitmapField field = new BitmapField(original,
  BitmapField.STAMP_MONOCHROME);
```

Drawing and Rendering Images (Cont.)

- Draw an image in color
  - Draw an image on an empty bitmap.
    - Create an empty bitmap. The example below copies the type and size from an existing bitmap.
    - Create a Graphics object using the empty bitmap as the drawing surface.
    - To draw a new image using raw data retrieved from the original, invoke Graphics.rawRGB().

```
Bitmap restored = new Bitmap(original.getType(),
   original.getWidth(), original.getHeight());
Graphics graphics = new Graphics(restored);
try {
   graphics.drawRGB(argb, 0, restored.getWidth(), 0, 0,
   restored.getWidth(), restored.getHeight());
} catch(Exception e) {
   System.out.println("Error occurred during drawing:
   " + e);
}
```

Creating a Media Player

- To play audio on a device, use the API items in the javax.microedition.media package (JSR 135) to create a media player and then add functionality to it.
- Create a player for a sequence of tones.
  - Use the ToneControl to permit playback of a BlackBerry® device user-defined sequence of tones in an unvarying pitch. See “Access media player functionality” on page 89 for more information on media player controls.
  - Tempo is the beats per minute with 1 beat equal to 1/4 note. You determine the tempo by multiplying the tempo modifier by 4 to keep it within the byte range of 1 to 127. Tempos in the range of 20 bpm to 508 bpm equate to a tempo modifier range of 5 to 127.
- Create a player for media from a URL.
  - Invoke Manager.createPlayer(String locator). The string parameter must use URI syntax that describes the media content.

Creating a Media Player (Cont.)

- Create a player for media from an input stream.
  - Invoke Manager.createPlayer(InputStream stream, String type).
    - The type parameter represents the input media content type.
  - Check for a MediaException if null is the content type.

```
RecordStore recSt;
int recId;
try {
      InputStream inpStr = new
      ByteArrayInputStream((store.getRecord(recId));
      Player p = Manager.createPlayer(inpStr,
         "audio/mpeg");
      p.start();
} catch (IOException ioEx) {
} catch (MediaException meEx) {}
```

Creating a Media Player (Cont.)

- Create a player for streaming media.
  - For BlackBerry devices that operate on EDGE networks, Real Time Streaming Protocol (RTSP) functionality is available only over a WiFi connection.
  - Invoke Manager.createPlayer(String locator), passing an RTSP locator as a parameter.

```
Manager.createPlayer("rtsp://streaming.rim.com/streamin
  g_video.3gp");
```

- Create a player that displays a video in a field.
  - Create Player, VideoControl, and Field variables.

```
Player _videoPlayer;
VideoControl _videoControl;
Field videoField;
```

  - Start a try block.

```
try {
```

Creating a Media Player (Cont.)

- Create a player that displays a video in a field.
  - Invoke Manager.createPlayer(String locator), where locator is a string in URI syntax that describes the video content. Store a reference to the Player object that the call to createPlayer(String locator)returns.

```
_videoPlayer =
  Manager.createPlayer("file:///SDCard/BlackBerry/videos/
  soccer1.avi");
```

  - To enable a Player to get the information it requires to acquire media resources, invoke Player.realize().

```
_videoPlayer.realize();
```

  - Invoke Player.getControl(), using as a parameter a string representation of the VideoControl class. Cast the returned object as a VideoControl object.

```
_videoControl = (VideoControl)_videoPlayer.getControl(
  "javax.microedition.media.control.VideoControl");
```

Creating a Media Player (Cont.)

- Create a player that displays a video in a field.
  - To initialize the mode that a videoField uses to display the video, invoke VideoControl.initDisplayMode(int mode, Object arg). Use the arg parameter to specify the UI primitive that will display the video. For example, in a BlackBerry Application, use "net.rim.device.api.ui.Field" as the arg parameter, casting the object that this method returns as a Field object. See the API reference for the BlackBerry Java Development Environment for more information.

```
videoField = (Field)_videoControl.initDisplayMode(
    VideoControl.USE_GUI_PRIMITIVE,
    "net.rim.device.api.ui.Field" );
```

  - Check for any exceptions that may have occurred within the try block.

```
} catch ( Exception e ) {
System.out.println( "Exception: " + e.toString() );
}
```

Working with the BlackBerry Media Player

- Start the media player from the BlackBerry Browser
  - Invoke Browser.getDefaultSession().

```
BrowserSession soundclip =
  Browser.getDefaultSession();
```

  - Invoke BrowserSession.displaypage().

```
soundclip.displayPage(
  "file:///SDCard/BlackBerry/music/TarzanYell.mp3"
  );
```

Working with the BlackBerry Media Player (Cont.)

- Start the media player with no content
  - Import the javax.microedition.content package.
    - Import javax.microedition.content;
  - Invoke Registry.getRegistry(), storing a reference to the returned object in a Registry object. The classname parameter is the name of the class in the application that extends
    - javax.microedition.midlet.MIDlet, net.rim.device.api.system.Application or net.rim.device.api.ui.UiApplication.
    - Registry reg = Registry.getRegistry(String classname);
  - Create a new instance of an Invocation object, storing a reference to the object in an Invocation object.
    - Invocation invocation = new Invocation(null, null, BlackBerryInvocation.CONTENT_HANDLER_MEDIA_PLAYER);
  - Invoke Registry.invoke(Invocation invocation) using the new Invocation object as a parameter.
    - reg.invoke(invocation);

Working with the BlackBerry Media Player (Cont.)

- You can use the API items in the javax.microedition.media package (JSR 135) to create a BlackBerry application that can play media.
- Prepare the media player.
  - Invoke Player.realize().
  - Invoke Player.prefetch().
- Start the media player.
  - Invoke Player.start(). The Player returns to the Prefetched state when you invoke Player.stop() or when it reaches the end of the media file.

```
try {
    Player p =
      Manager.createPlayer("http://www.test.rim.net/abc.w
      av");
    p.start();
} catch (MediaException pe) {
} catch (IOException ioe) {
}
```

Working with the BlackBerry Media Player (Cont.)

- Determine the controls that a media player supports.
  - Invoke Player.getControls().
  - To provide additional functionality for a media player, use one or more of the controls that the media player supports.
  - You can use the same object to access multiple controls: for example, one object can be both a VolumeControl and a ToneControl. The javax.microedition.media package contains a number of Control interfaces. See the API Reference in the BlackBerry® Java® Development Environment for more information about the javax.microedition.media package.

Working with the BlackBerry Media Player (Cont.)

- Enable video playback support
  - Invoke Player.getControls() to retrieve a VideoControl object.
  - Implement the methods of the VideoControl interface to give a BlackBerry® Java® Application a variety of video support features, including the following:
    - control over the mode of video display (one of USE_GUI_PRIMITIVE or USE_DIRECT_VIDEO)
    - control over the location of the video with respect to the canvas that displays the video
    - access to the X-coordinate and the y-coordinate of the video with respect to the GUI object that displays the video

Working with the BlackBerry Media Player (Cont.)

- Enable video playback support
  - displaying or hiding video
  - resizing the video image
- Adjust the volume of the media player.
  - Invoke VolumeControl().
  - Define a volume value in the following range:
    - 0: no volume
    - 100: maximum volume level
  - The PlayerListener sends a VOLUME_CHANGED event when its state changes.
- Close the media player.
  Invoke Player.stop().

Listening for Media Player Events

- You can use the API items in the javax.microedition.media package (JSR 135) to create a BlackBerry application that can listen for and send media player events.
- Listen for changes to the media player state.
  - Implement PlayerListener.
  - To register the player listener, invoke addPlayerListener.

```
private void doPlay() throws IOException, MediaException {
    Player p = Manager.createPlayer("http://www.rim.com/rim.mp3");
    p.addPlayerListener(this);
    p.realize();
    p.prefetch();
    p.start();
}
```

- Send a media player event to a registered player listener.
  - Invoke playerUpdate(Player player, String event, Object eventData).

```
public void playerUpdate(Player player,String event, Object eventData) {      //
    Release resources
        player.close();
        if ( event == PlayerListener.END_OF_MEDIA ) // Add code for
            actions if the end of media is reached.
}
```

Rich Media

- Playing rich media content
  - To play rich media content, use the following classes:
    - To retrieve PME content on BlackBerry® devices or networks, use methods from the MediaManager class.
    - To play PME content that exists on BlackBerry devices, use methods from the MediaPlayer class.
- Download rich media content
  - Create a MediaManager object.
  - Invoke MediaManager.createMedia().

Rich Media (Cont.)

- Download rich media content
  - The first time that you invoke MediaManager.createMedia(), the URI must be absolute, unless you first invoke MediaManager.setProperty("URI_BASE", base_url) to set a base URL. When you invoke createMedia() subsequently, the URL that the method used previously is the base.

```
MediaManager manager = new MediaManager();
try {
    Object media =
      manager.createMedia("http://webserver/sample.pme
      ");
} catch (IOException ioe) {
    System.out.println("Error: requested content was not
      downloaded.");
} catch (MediaException me) {
   System.out.println("Error: " + me.getCode()); }
```

Playing Rich Media Content

- Set the PME object for playback.
  - Invoke MediaPlayer.setMedia().

```
MediaPlayer player = new MediaPlayer();
try {
    player.setMedia(media);
} catch (MediaException me) {
    System.out.println("Error: requested content type is not
    supported.”);
}
```

- Allow an application’s screen to display content created in Plazmic Media Engine Version 4.2.
  - To display content created in Plazmic Media Engine Version 4.2, the screen must not support scrolling.
  - Create an instance of a screen object using the NO_VERTICAL_SCROLL and NO_HORIZONTAL_SCROLL fields (inherited from the Manager class).

```
Screen screen = new Screen(Screen.NO_VERTICAL_SCROLL |
Screen.NO_HORIZONTAL_SCROLL);
```

Playing Rich Media Content (Cont.)

- Allow an application's screen to display content created in Plazmic Media Engine Version 4.2.
- Retrieve a UI object that displays rich media content.
  - Invoke MediaPlayer.getUI().
  - Cast the object that getUI() returns as a Field, and add it to a Screen for display.

```
screen.add((Field)player.getUI());
```

- Play rich media content.
  - Check the media player state.
  - Invoke MediaPlayer.start().

```
if(player.getState() == MediaPlayer.REALIZED) {
    try {
        player.start();
    } catch(MediaException me) {
        System.out.println("Error occurred during media
        playback: " + me.getCode() + me.getMessage());
    }
}
```

Listening for Rich Media Events

- **Listen for media engine events.**
  - **Implement the MediaListener interface to let your BlackBerry® Java® Application listen for media engine events.**
  - **Implement mediaEvent() to handle all possible media events.**

```
public final class MediaListenerImpl implements MediaListener {
    public void mediaEvent(Object sender, int event, int eventParam, Object
      data) {
        switch(event) {
            case MEDIA_REQUESTED: // Perform action.
            break;
            case MEDIA_COMPLETE: // Perform action.
            break;
            case MEDIA_REALIZED: // Perform action.
            break;
            case MEDIA_IO: // Perform action.
            break;
        }
    }
}
```

Listening for Rich Media Events (Cont.)

- Register the listener.
  - Invoke addMediaListener() on the MediaPlayer and MediaManager objects.

```
private MediaListenerImpl _listener = new MediaListenerImpl();
private MediaPlayer player = new MediaPlayer();
private MediaManager manager = new MediaManager();
player.addMediaListener(_listener);
manager.addMediaListener(_listener);
```

- Load content in the background, and play it when the download is complete.
  - To download content for future playback, invoke MediaManager.createMediaLater().
  - In MediaListener.mediaEvent(), add code to manage the MEDIA_REALIZED event that occurs when the content the application downloads finishes loading on the BlackBerry® device.

Listening for Rich Media Events (Cont.)

- Load content in the background, and play it when the download is complete.
  - To register the content that the data parameter specifies, invoke MediaPlayer.setMedia(data).
  - To start playback, invoke MediaPlayer.start().

```
manager.createMediaLater("http://webserver/sample.pme");
public void mediaEvent(Object sender, int event, int eventParam, Object data) {
    switch(event) {
        ...
        case MEDIA_REALIZED:
        try {
            player.setMedia(data);
            player.start();
        } catch(MediaException me) {
            System.out.println("Error playing media" + me.getCode() +
            me.getMessage());
        }
        break;
    }
}
```

Listening for Rich Media Events (Cont.)

- Track the progress of a download.
  - Extend the net.rim.plazmic.mediaengine.io.LoadingStatus class.
  - In your implementation of mediaEvent(), when the MEDIA_IO event occurs, cast the Object in the data parameter to a LoadingStatus object.
  - To retrieve the download status, and manage each status, invoke LoadingStatus.getStatus().
  - For each normal status, print a message to the console.
- Manage a failed download.
  - For the LOADING_FAILED status, perform the following actions:
  - To retrieve the error code, invoke LoadingStatus.getCode().
  - To retrieve the detailed message, invoke LoadingStatus.getMessage().
  - To retrieve the URL string of the content, invoke LoadingStatus.getSource().