Wireless Online Game Development for Mobile Device

Lesson 4

What have we learnt last week?

- High-Level User Interface Object in Form: ChoiceGroup, DateField and Gauge
- How to display image in High-Level User Interface: Form, List, Alert and ChoiceGroup
- The first game: Guess Number

Why need Low-level API?

- Although the high-level API provides a complete set of components for building an application user interface, the high-level components do not provide a means to draw directly onto the device display.
- Without this flexibility, we are greatly limited as to the types of applications we can create.
- If the high-level API is the workhorse of MIDP, then the low-level API is the one we turn to when we need to let our imaginations run wild.

Components of the Low-level API

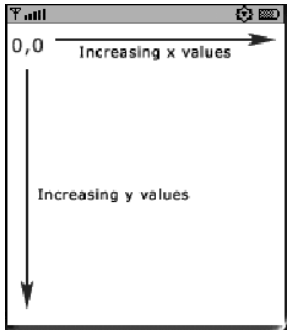
- **Canvas** and **Graphics** are the two classes at the heart of the low-level API.
  - **Canvas** forms the backdrop for writing onto the device display as well as managing related events.
  - **Graphics** provides the actual tools for drawing, such as drawRoundRect() and drawString().

The Canvas Class

- The Canvas class provides the backdrop for creating a custom user interface.
- The bulk of the methods in this class are used to handle events and draw images and text onto the device display.

The Coordinate System

- The coordinate system for the Canvas class originates in the upper-left corner of the display.
- x values increase moving to the right; y values increase going down.
- When drawing, the pen thickness is one pixel in width and height.



Height and Width

- The height and width of the Canvas represent the entire drawing area available to the device display.
- If you do not specify the size of the canvas you would like; the software on an MIDP device will always return the maximum drawing area available for a given device.
- To determine the width and height of the canvas:
  - int getWidth( ) // Canvas width
  - int getHeight( ) // Canvas height

Creating a Canvas

```
Class ImageCanvas extends Canvas {
    ...
    MIDletDisplay = Display.getDisplay(this);
    ...
    protected void paint(Graphics g) {
          // Draw onto the canvas
          ...
    }
}

ImageCanvas canvas = new ImageCanvas(this);
```

Painting on the Canvas

- The Canvas class method paint( ) lets you draw shapes, display images, write text, and otherwise create what will be shown on the device display.

How to Display Image?

- Use Canvas for display
- In J2ME Wireless Toolkit 2.2
  - The maximum screen size is 240 x 290 pixels for “Default Color Device”
  - The maximum screen size is 180 x 180 pixels for “Media Control Skin”
- Can only display image in PNG format
- All local image must stored in RES folder

Create Image

- To create an image, the basic syntax is:
  - ***createImage (Image source)***
- Parameters:
  - “Source” is the source image file to be copied (must place in Resource folder)

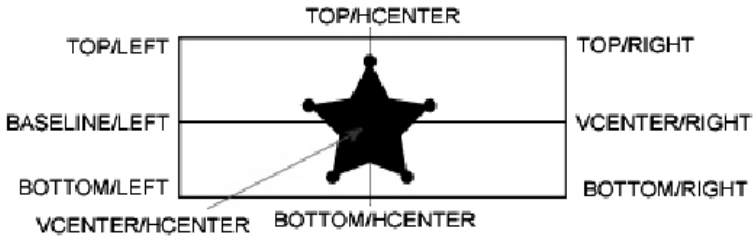
Draw Image

- To draw the image on screen, the basic syntax is:
  - ***drawImage (Image Picture, int x, int y, int anchor)***
- Parameters
  - “Picture” is the name for the image file
  - “x” is the x coordinate of the anchor point
  - “y” is the y coordinate of the anchor point
  - “anchor” is the anchor point for positioning the image.

Anchor Points

- The drawing of text is based on "anchor points".
- Anchor points are used to minimize the amount of computation required when placing text.
- The definition of the anchor point must be one of the horizontal constants (LEFT, HCENTER, RIGHT) combined with one of the vertical constants (TOP, BASELINE, BOTTOM) using the bit-wise OR operator.
- Zero may also be used as the value of an anchor point. Using zero for the anchor point value gives results identical to using TOP | LEFT.

Image Anchor Points



Rectangle

- Rectangles can be either outlined or filled.
- You can specify that a rectangle have either square or rounded corners.
- When drawing rounded corners, you specify the horizontal diameter (arcWidth) and the vertical diameter (arcHeight) to define the sharpness of the arc in each direction (Larger values represent a more gradual arc; smaller ones tighten the curve).

Draw a Regular Rectangle

- The syntax for drawing a regular rectangle is:
  - ***drawRect (int x, int y, int width, int height)***
- Parameters
  - "x" is the x coordinate of the rectangle
  - "y" is the y coordinate of the rectangle
  - "width" is the width of the rectangle
  - "height" is the height of the rectangle

Fill a Regular Rectangle

- The syntax for filling a regular rectangle is:
  - ***fillRect (int x, int y, int width, int height)***
- Parameters
  - "x" is the x coordinate of the rectangle
  - "y" is the y coordinate of the rectangle
  - "width" is the width of the rectangle
  - "height" is the height of the rectangle

Color Support

- To sets the current color to the specified RGB values.
- All subsequent rendering operations will use this specified color.
- You can specify the color in one of two ways: you can assign an integer to represent all three colors values (red, green, and blue, with eight bits allocated to each color), or you can use separate parameters for each color.
- When using one value to hold the color, red occupies the uppermost eight bits, green the middle, and blue the lower.

Set Color

- The syntax for setting color is:
  - ***setColor (int red, int green, int blue)***
- Parameters:
  - "red" is the red component of the color being set in range 0-255
  - "green" is the green component of the color being set in range 0-255
  - "blue" is the blue component of the color being set in range 0-255

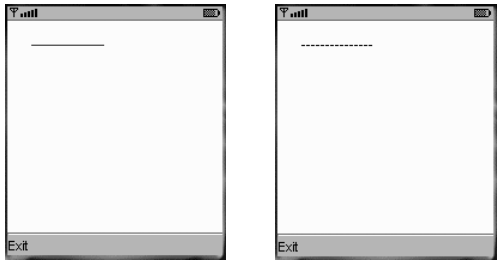
Clear Screen

- There is no direct method to clear the screen in MIDlet.
- However, you can use a full screen rectangle to clear the screen.

```
protected void paint(Graphics g) {
        // Set background color to white
        g.setColor (255, 255, 255);
        // Fill the entire canvas
        g.fillRect (0, 0, getWidth(), getHeight());
}
```

Drawing Line

- Draws a line between the coordinates (x1, y1) and (x2, y2) using the current color and stroke style.



Drawing Line

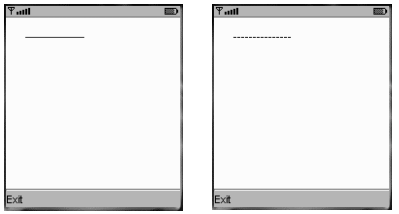
- To draw a line on screen, the basic syntax is:
  - ***drawLine (int x1, int y1, int x2, int y2)***
- Parameter
  - "x1" is the x coordinate of the start of the line
  - "y1" is the y coordinate of the start of the line
  - "x2" is the x coordinate of the end of the line
  - "y2" is the y coordinate of the end of the line

Set the Stroke Style

- Sets the stroke style used for drawing lines, arcs, rectangles, and rounded rectangles.
- This does not affect fill, text, and image operations.

Set the Stroke Style

- To set the line style, the basic syntax is:
  - ***setStrokeStyle(int style)***
- Parameters
  - style can be SOLID (0) or DOTTED (1)



Drawing Arc

- When drawing an arc, you can draw it as an outline or request that it be filled.
- You start by specifying the outside dimensions of an imaginary box (also known as the bounding box) in which it will be drawn.
- The start angle determines where the drawing of the arc will begin, with 0 being at 3 o'clock (Positive values go counter-clockwise).
- The arc angle specifies how many degrees to draw from the start angle, going in a counter-clockwise direction.

Drawing Arc

- The syntax for drawing a arc is:
  - ***drawArc (int x, int y, int width, int height, int startAngle, int arcAngle)***
- Parameters
  - "x" is the x coordinate of the upper-left corner of arc
  - "y" is the y coordinate of the upper-left corner of arc
  - "width" is the width of the arc to be drawn
  - "height" is the height of the arc to be drawn
  - "startAngle" is the beginning angle
  - "arcAngle" the angular extent of the arc, relative to the start angle

Filling Arc

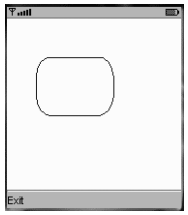
- Fills a circular or elliptical arc covering the specified rectangle.
- The resulting arc begins at startAngle and extends for arcAngle degrees. Angles are interpreted such that 0 degrees is at the 3 o'clock position. A positive value indicates a counter-clockwise rotation while a negative value indicates a clockwise rotation.
- The center of the arc is the center of the rectangle whose origin is (x, y) and whose size is specified by the width and height arguments.
- If either width or height is zero or less, nothing is drawn.

Filling Arc

- The syntax for filling a arc is:
  - ***fillArc (int x, int y, int width, int height, int startAngle, int arcAngle)***
- Parameters
  - "x" is the x coordinate of the upper-left corner of arc
  - "y" is the y coordinate of the upper-left corner of arc
  - "width" is the width of the arc
  - "height" is the height of the arc
  - "startAngle" is the beginning angle
  - "arcAngle" the angular extent of the arc, relative to the start angle

Draw a Rounded Rectangle

- Draws the outline of the specified rounded corner rectangle using the current color and stroke style.
- The resulting rectangle will cover an area (width + 1) pixels wide by (height + 1) pixels tall.
- If either width or height is less than zero, nothing is drawn.

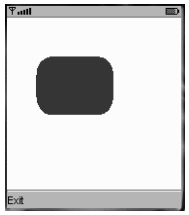


Draw a Rounded Rectangle

- The syntax for drawing a rounded rectangle is:
  - ***drawRoundRect (int x, int y, int width, int height, int arcWidth, int arcHeight)***
- Parameters
  - "x" is the x coordinate of the rectangle
  - "y" is the y coordinate of the rectangle
  - "width" is the width of the rectangle
  - "height" is the height of the rectangle
  - "arcWidth" is the horizontal diameter of the arc at the four corners
  - "arcHeight" is the vertical diameter of the arc at corners

Fill a Rounded Rectangle

- Fills the specified rounded corner rectangle with the current color.
- If either width or height is zero or less, nothing is drawn

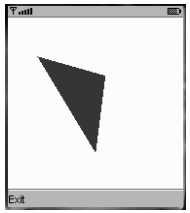


Fill a Rounded Rectangle

- The syntax for filling a rounded rectangle is:
  - ***fillRoundRect (int x, int y, int width, int height, int arcWidth, int arcHeight)***
- Parameters
  - "x" is the x coordinate of the rectangle
  - "y" is the y coordinate of the rectangle
  - "width" is the width of the rectangle
  - "height" is the height of the rectangle
  - "arcWidth" is the horizontal diameter of the arc at the four corners
  - "arcHeight" is the vertical diameter of the arc at corners

Fill a Triangle

- Fills the specified triangle will the current color. The lines connecting each pair of points are included in the filled triangle.



Fill a Triangle

- The syntax for filling a triangle is:
  - ***fillTriangle (int x1, int y1, int x2, int y2, int x3, int y3)***
- Parameters
  - "x1" is the x coordinate of the first vertex
  - "y1" is the y coordinate of the first vertex
  - "x2" is the x coordinate of the second vertex
  - "y2" is the y coordinate of the second vertex
  - "x3" is the x coordinate of the third vertex
  - "y3" is the y coordinate of the third vertex

Set Font

- Sets the font for all subsequent text rendering operations.
- If font is null, it is equivalent to setFont(Font.getDefaultFont()).
- Style parameters can be combined using a logical (|) operator,
- The attributes for the Font class are listed below:

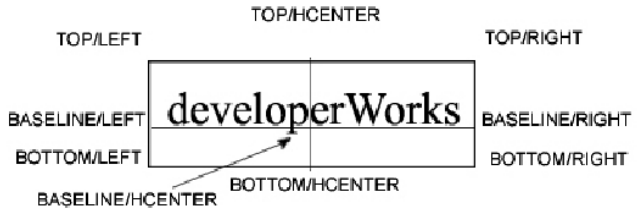
FACE_SYSTEM	STYLE_PLAIN	SIZE_SMALL
FACE_MONOSPACE	STYLE_BOLD	SIZE_MEDIUM
FACE_PROPORTIONAL	STYLE_ITALIC	SIZE_LARGE
	STYLE_UNDERLINED	

Set Font

- The syntax for set font style is:
  - ***setFont (Font font)***
- Parameters
  - "font" is the specified font

Display String

- Draws the specified String using the current font and color.
- The x,y position is the position of the anchor point.

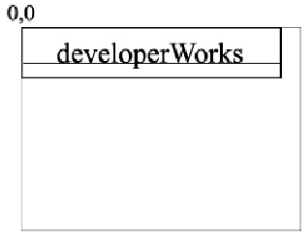


Display String

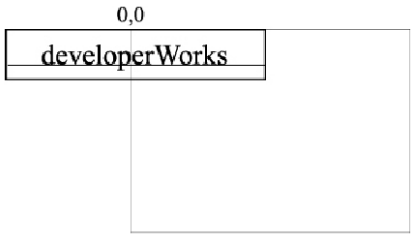
- To write an string on screen, the basic syntax is:
  - ***drawString (String str, int x, int y, int anchor)***
- Parameters
  - "str" is the String to be drawn
  - "x" is the x coordinate of the anchor point
  - "y" is the y coordinate of the anchor point
  - "anchor" is the anchor point for positioning text

Working with anchor points

- When using anchor points, you are stating which location on the bounding box is located at the specified x and y coordinates.



Output of a TOP/LEFT anchor point specification



Output of a TOP/HCENTER anchor point specification

Deployment of MIDlets

- The key component in the deployment of MIDlets is the Java Application Descriptor (JAD) file that describes all the deployment-related details for a MIDP executable.

```
MIDlet-1: MyProject, MyProject.png, MyClass
MIDlet-Jar-Size: 934
MIDlet-Jar-URL: MyProject.jar
MIDlet-Name: MyProject
MIDlet-Vendor: Unknown
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
```

Attributes in a JAD file

- Attributes in a JAD file can be divided into two main groups, depending on whether they are required or not.
- Attributes beginning with MIDLet- are thought to be "system" attributes, and are reserved.
- Application-specific attributes cannot begin with the MIDlet- prefix

Required attributes in JAD file

- MIDlet-Name
  - Specifies the name of the application that will be shown to the user.
- MIDlet-Jar-Size
  - The size of the JAR file
- MIDlet-Jar-URL
  - The URL from where the JAR file can be downloaded.
- MIDlet-Vendor
  - The MIDlet vendor.
- MIDlet-Version
  - The MIDlet version.

Deployment from Web

- Before deploy your application from web, you must configure your web server.
- Create new MIME file type associations:
  - JAD: text/vnd.sun.j2me.app-descriptor
  - JAR: application/java-archive