



Mobile Information Device Programming (13)

Lecturer: Alireza Mousavi
School of Engineering & Design
www.brunel.ac.uk/~emstaam



Graphics

- **Graphics** object is a tool for drawing onto a *Canvas*
- Contains 30 methods
- You can draw text and shapes – Colour and Font



Graphics Object Lifetime

- 1. `paint()` method:** The *Graphics* reference is valid only inside the **`paint()` method**.
- 2. `Image`:** The *Graphics* reference can only be used as long as the *Image* class and the variable referring to the *Graphics* (e.g. ***g***) object are available.

```
Image myIMAGE = Image.createImage(30, 30);  
Graphics g = image.getGraphics( );
```



Stroke Style

There are two styles for drawing arcs, lines and rectangles

1. Solid (default)
2. Dashed

Method

int `getStrokeStyle()`

void `setStrokeStyle(int style)`

Description

Get the stroke style

Set the stroke style (Solid and Dotted)



Drawing Lines & Arcs

- Each line has a starting and ending point (**x**, **y**)
- Method `drawLine()`

```
g.setColor(0, 0, 0); // black line
```

```
g.drawLine(startx, starty, endx, endy); // draw line
```

- Two methods to draw Arcs
 - Create an outline of an Arc
 - Fill an Arc

<i>Methods</i>	<i>Description</i>
<i>void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)</i>	<i>Draw an arc with specified attributes</i>
<i>void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)</i>	<i>Fill an Arc inside the specified Arc attributes</i>



Example 13-1 Drawing an Arc

- Draw an Arc inside a bounding box defined by 7, 7, 70, 70
- The start angle will be 0 and the arc angle to be 240



Result

Do a fillArc method



Drawing Rectangles



<i>Method</i>	<i>Description</i>
<code>void drawRect(int x, int y, int width, int height)</code>	Draw Rectangle
<code>void drawRoundRect(int x, int y, int width, int height, int arcWidth, int ArcHeight)</code>	Draw round cornered Rectangle
<code>void fillRect(int x, int y, int width, int height)</code>	Fill the rectangle
<code>void fillRoundRect(int x, int y, int width, int height, int arcWidth, int ArcHeight)</code>	Fill a round rectangle





Text & Fonts

- Font support within J2ME is much more restricted than J2SE
- A new **Font** is requested through *static* method *Font.getFont()* - (why?)
- Bear in mind that sometimes the requested Font may not be supported by the implementation – **the closest will be displayed**



Drawing a Text

- You can draw single characters, an array of characters, a String, or a subset of a String. (6 methods [see slide 12](#))
- Some things to consider:
 - Font : has *face, style and size*
 - You can combine *style* (**only**) with a **logical OR (|)**

Example:

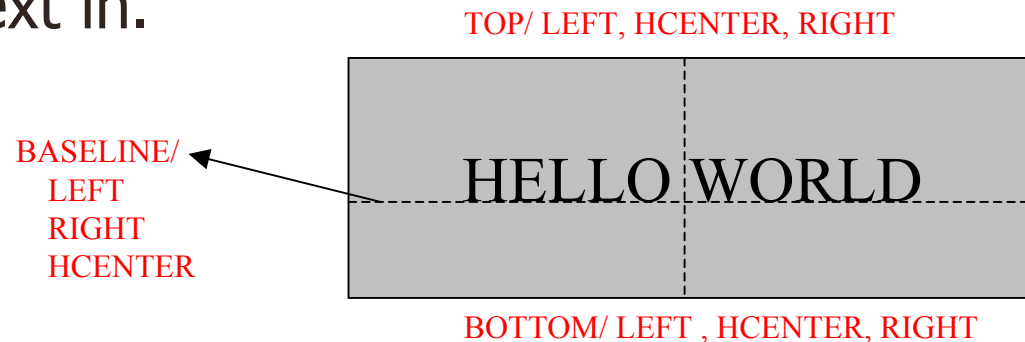
```
Font myfont = Font.getFont(Font.FACE_System, Font.STYLE_ITALIC | UNDERLINED,  
    Font.SIZE_LARGE);
```

- There are seven methods for querying font attributes



Anchor Points

- Makes text Aligning easier
- Anchor points are assigned in pairs like ***x*** and ***y*** coordinates
- The ***x*** values can be LEFT, RIGHT and HCENTER
- The ***y*** values can be TOP, BASELINE and BOTTOM
- It is like having an imaginary box where you try to place your text in.





Drawing Text Methods

<i>Method</i>	<i>Description</i>
<i>void drawChar(char character, int x, int y, int anchor)</i>	Draw a character
<i>void drawChars(char[], data, int offset, int length, int x, int y, int anchor)</i>	Draw an array of characters
<i>void drawString(String str, int x, int, y, int anchor)</i>	Draw a String
<i>void drawSubstring(String str, int offset, int length, int x, int y, int anchor)</i>	Draw a part of a String
<i>Font getFont()</i>	Get the current font
<i>void setFont (Font font)</i>	Set the current Font to font

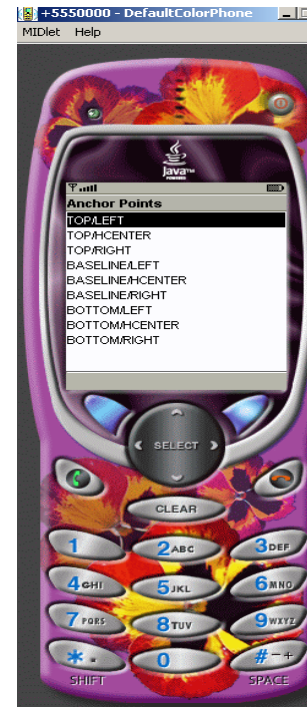


Example Moving Text E13-2

- Draw a text
- Use Anchor position to move the Text along specified alignments



A. Mousavi





Drawing Images

1. Immutable images

- Allocate the image (memory):

```
Image myImage = Image.createImage("/myimage.png");
```

- Display the Image:

```
protected void paint(Graphics g){  
g.drawImage(myImage, 15, 15, Graphics.LEFT | Graphics.TOP) }
```

2. Mutable images

- Allocate the image (memory):

```
Image myImage = Image.createImage(50, 100);
```

- Create the image using drawing tools (line, arc, rectangle and text)
- Display the image

```
g.drawImage(myImage, 20, 20, Graphics.RIGHT | Graphics.BOTTOM);
```



Translating Coordinates

A Graphics object can reference its origin (0, 0) at different places on the display.

<i>Method</i>	<i>Description</i>
<i>void translate(int x, int y)</i>	Translate the origin for a Graphics object (0, 0)
<i>int getTranslateX()</i>	Get the current translated <i>x</i> coordinate
<i>int getTranslateY</i>	Get the current translated <i>y</i> coordinate

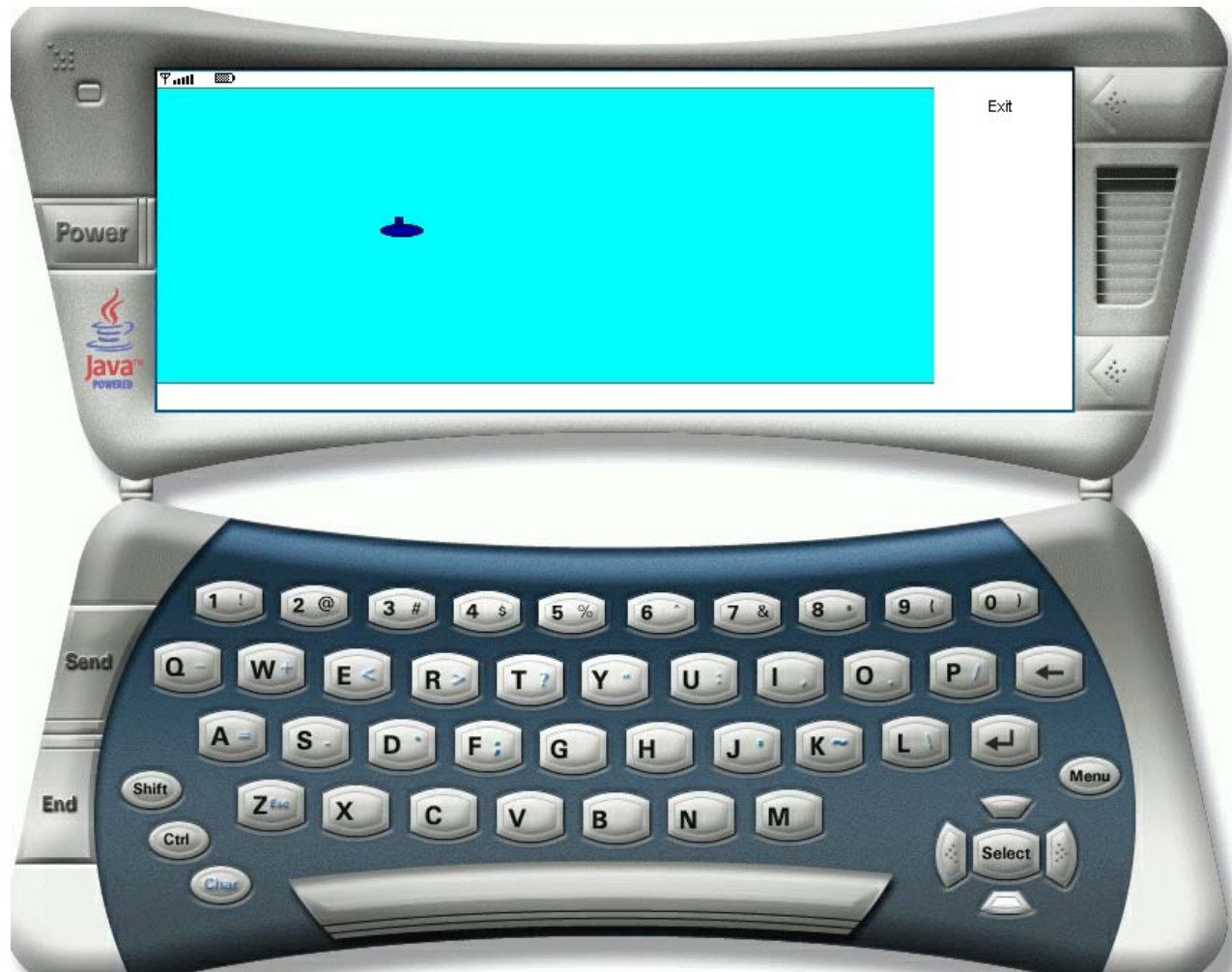


Example Translate Coordinate E13-3

- Move an image in different direction on a Canvas
- Use the game actions LEFT, RIGHT, UP and DOWN
- Note keyPressed() actions



Result





Clipping Regions

- Intended to reduce the time for refreshing the display
- Previously we redrew images on the entire display
paint() and *repaint()*



Clipping Methods

<i>Method</i>	<i>Description</i>
<i>void setClip(int x, int y, int width, int height)</i>	Set the clipping area(rectangle)
<i>void clipRect(int x, int y, int width, int height)</i>	Intersect this rectangle with the clipping one to generate a new region
<i>int getClipX() or int getClipY()</i>	Get the X or Y coordinates of current clipping region
<i>int getClipHeight() or getClipWidth()</i>	Get the current height or width of the clipping



Example E13-4

- Create an application that moves the clipping region within a Canvas
- Use game actions to generate events (LEFT, RIGHT, TOP, BOTTOM and FIRE)
- Use a random number generator to start the coordinates (a sort of a game that you will guess the image!)



Result

