# Mobile and Wireless Systems Programming

Graphical User Interfaces
Low level API

# Canvas

- javax.microedition.lcdui.Canvas (inherits from Displayable)
- abstract class
- paint() and repaint() methods
- draw text, image, geometric shapes...
- Can be mixed with high level API (not in the same screen)

# Screen's properties

- getWidth()
- getHeight()
- setFullScreenMode(true)
- sizeChanged()
- isColor()
- numColors()
- isDoubleBuffered()

# Events

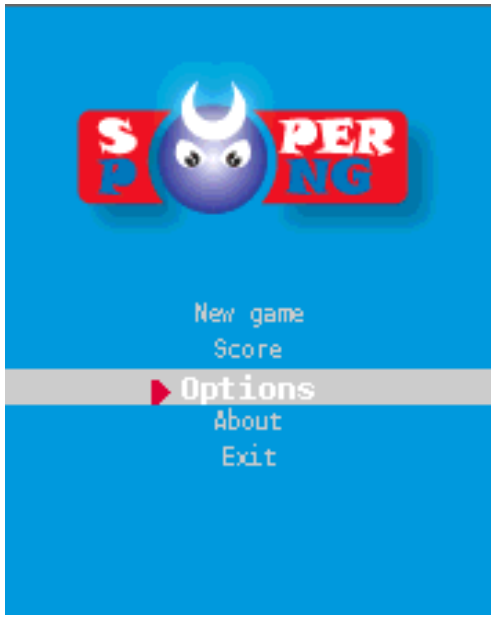
- showNotify()
- hideNotify()
- keyPressed()
- keyRepeated()
- keyReleased()
- pointerPressed()
- pointerDragged()
- pointerReleased()
- paint()

# Key codes

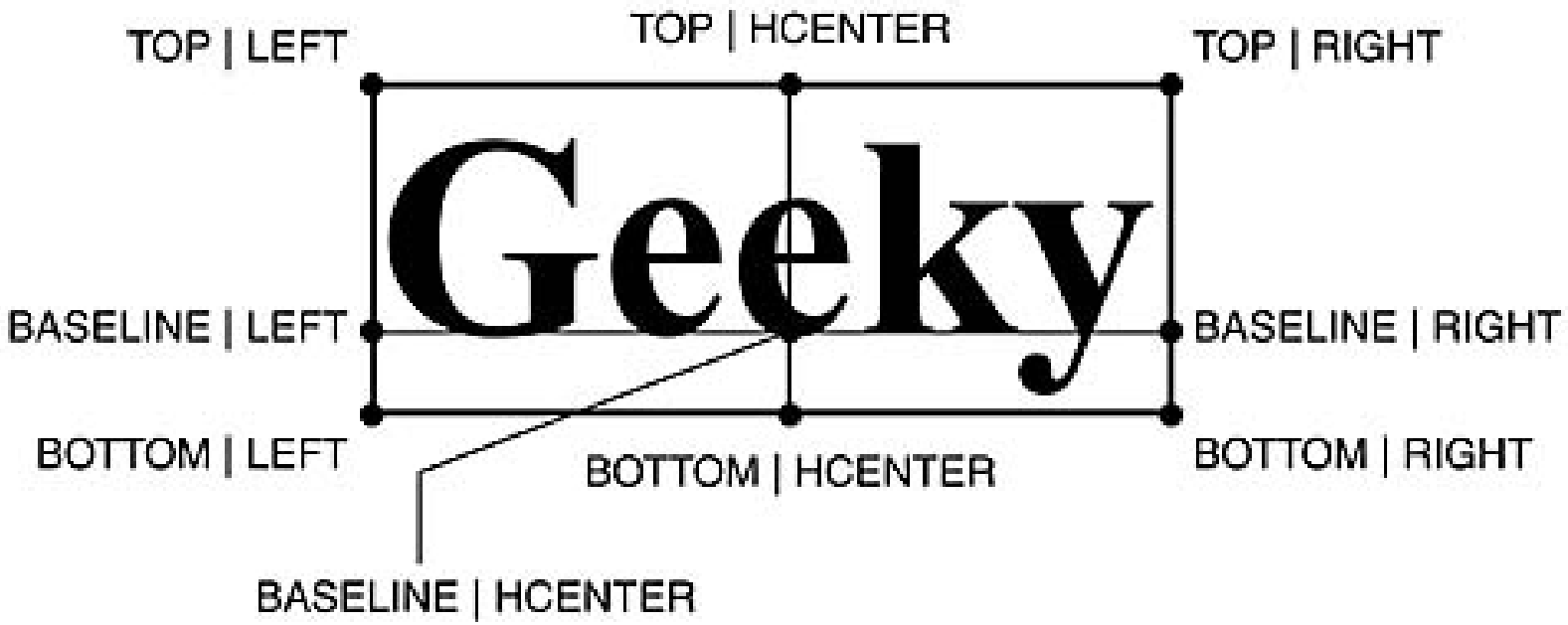
- KEY_NUM0
- KEY_NUM1
- ...
- KEY_NUM9
- KEY_POUND
- KEY_STAR
- LEFT
- RIGHT
- UP
- DOWN
- FIRE
- ...

# Drawing

- drawLine(...)
- drawString(...)
- drawRect(...)
- fillArc(...)
- setFont(...)
- setColor(...)
- drawImage(...)
- ...



# Anchor


TOP | LEFT
TOP | HCENTER
TOP | RIGHT
Geeky
BASELINE | LEFT
BASELINE | RIGHT
BOTTOM | LEFT
BOTTOM | HCENTER
BOTTOM | RIGHT
BASELINE | HCENTER

# Color

- int red = 0xFF0000 ;
- int green = 0x00FF00 ;
- int blue = 0x0000FF ;
- setColor()
- getColor()
  - COLOR_BACKGROUND
  - COLOR_FOREGROUND
  - COLOR_BORDER
  - COLOR_HIGHLIGHTED_BACKGROUND
  - COLOR_HIGHLIGHTED_FOREGROUND
  - COLOR_HIGHLIGHTED_BORDER

# Font

- Face
  - javax.microedition.lcdui.Font.FACE_MONOSPACE
  - javax.microedition.lcdui.Font.FACE_PROPORTIONAL
  - javax.microedition.lcdui.Font.FACE_SYSTEM
- Style
  - javax.microedition.lcdui.Font.STYLE_PLAIN
  - javax.microedition.lcdui.Font.STYLE_ITALIC
  - javax.microedition.lcdui.Font.STYLE_BOLD
  - javax.microedition.lcdui.Font.STYLE_UNDERLINED
- Size
  - javax.microedition.lcdui.Font.SIZE_LARGE
  - javax.microedition.lcdui.Font.SIZE_MEDIUM
  - javax.microedition.lcdui.Font.SIZE_SMALL

# Font

- getFont()
  - Font FONT_PLAIN = Font.getFont (Font.FACE_MONOSPACE,Font.STYLE_PLAIN, Font.SIZE_MEDIUM);
  - ...
- setFont()
- Combined attributes : STYLE_BOLD | STYLE_ITALIC

# Image

- MSA : PNG and JPEG

```
Image image;

try {
    image = Image.createImage("/img.png");
} catch (IOException ex) {
    ex.printStackTrace();
}

public void paint (Graphics g) {
    g.setGrayScale (255);
    g.fillRect (0, 0, getWidth (), getHeight ());
    g.drawImage (image, 0, 0, Graphics.TOP | Graphics.LEFT);
    g.drawImage (image, getWidth () / 2, getHeight () / 2,
                 Graphics.HCENTER | Graphics.VCENTER);
    g.drawImage (image, getWidth (), getHeight (),
                 Graphics.BOTTOM | Graphics.RIGHT);
}
...
```

# Animation

## Thread

```
class MyThread extends Thread{

    public MyThread() {
        ...
    }

    public void run(){
        ...
    }
}
```

---

```
    MyThread th = new MyThread();
    th.start();
```

## Runnable

```
class MyRun implements Runnable{

    public MyRun(){
        ...
    }

    public void run() {
        ...
    }
}
```

---

```
    MyRun rnb = new MyRun();
    new Thread(rnb).start;
```

# Game API

- Since MIDP 2.0
- javax.microedition.lcdui.game.*
  - GameCanvas
  - Layer
  - LayerManager
  - Sprite
  - TiledLayer

# Drawbacks

- Difficult to build custom items
- Portability issues
- Solutions : LWUIT, Kuix...

# Conclusion

- Game API become to be used widely (MIDP 2.0)
- Use Thread safely
- Professional canvas based applications
- Javadoc : http ://java.sun.com/javame/reference/apis/jsr118/

## Low level UI example

An application to draw shapes, texts and images.