Programming Mobile Devices

Low Level User Interface; Threads and Timers

University of Innsbruck

WS 2009/2010

thomas.strang@sti2.at

STI · INNSBRUCK

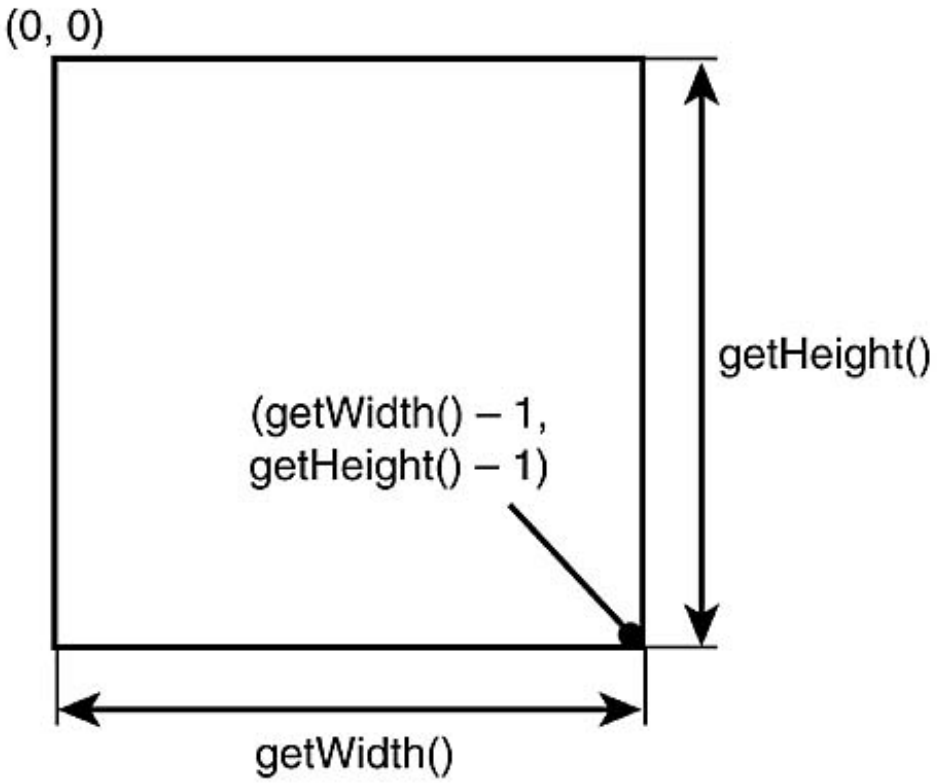
LCDUI Displayables

- LCDUI classes extending `Displayable`:
  - `Screen` ("high level GUI")
    - direct subclasses:
      - **TextBox**: A screen that allows the user to enter / edit text
      - **Alert**: A screen that allows data to the user and waits for a certain period of time before proceeding
      - **Form**: A screen that contains an arbitrary mixture of items (images, text, text fields, choice groups, etc.)
      - **List**: A screen containing a list of choices
    - common to all: `Title` and `Ticker` properties
  - `Canvas` ("low level GUI")
    - later …

Low Level User Interface

- `Canvas` is designed for applications that need precise placement and control of graphic elements and need access to low level input
- based on a redrawing mechanism – the application is responsible for redrawing the screen
  - using the `paint` / `repaint` methods
  - called automatically with `showNotify`
- Drawing area size can be determined with `Canvas.getWidth()` and `Canvas.getHeight()`

Canvas Coordinate System



the coordinates do not address the pixel itself, but the space between two pixels

-> the first pixel in the upper left corner of the display lies in the square bounded by coordinates (0,0) , (1,0) , (0,1) , (1,1)

A "Hello World" Canvas

```
import javax.microedition.lcdui.*;

public class MyCanvas extends Canvas {
  public void paint(Graphics g) {
    g.setColor(255, 0, 0);
    g.fillRect(0, 0, getWidth(), getHeight( ));
    g.setColor(255, 255, 255);
    g.drawString("Hello World!", 0, 0, g.TOP | g.LEFT);
  }
}


// … within a MIDlet

public void startApp( ) {
  Canvas canvas = new MyCanvas( );
  Display display = Display.getDisplay(this);
  display.setCurrent(canvas);
}
```

Canvas Input Events

- Events delivered to a Canvas are:
  - showNotify()
  - hideNotify()

  } visibility notifications
  - keyPressed()
  - keyRepeated()
  - keyReleased()

  } key events
  - pointerPressed()
  - pointerDragged()
  - pointerReleased()

  } pointer events
  - paint()
  - the CommandListener's commandAction() method

Canvas Output: Graphics object

- `Graphics g` is a parameter of the `paint` event
- used to draw arbitrary pixels or to apply drawing functions such as
  - drawString / drawSubstring
  - drawChar / drawChars
  - drawImage
  - drawRect
  - drawLine
  - etc.
- colors, font, style of lines etc. can be set

Exercise 4:

- centered maximum cake pie with caption

using `drawArc` / `fillArc`

Special LIBs (1): Thread

- A `Thread` is a sequence of instructions which may be executed in parallel with other threads, sharing the same address space within a process
- MIDP allows two ways to create a new thread of execution:
  1. Declare a class to be a subclass of `Thread` which overrides the `run` method
  2. Declare a class that implements the `Runnable` interface which then implements the `run` method

Threads – Method 1

- Direct subclassing `Thread`:

```
class PrimeThread extends Thread {
  long minPrime;
  PrimeThread(long minPrime) {
    this.minPrime = minPrime;
  }
  public void run() {
    // compute primes larger than minPrime …
  }
}
```

- The following code would then create a thread and start it running:

```
PrimeThread p = new PrimeThread(143);
p.start();
```

Threads – Method 2

- Implementing the `Runnable` interface:

```
class PrimeRun implements Runnable {
  long minPrime;
  PrimeRun(long minPrime) {
    this.minPrime = minPrime;
  }
  public void run() {
    // compute primes larger than minPrime …
  }
}
```

- The following code would then create a thread and start it running:

```
PrimeRun p = new PrimeRun(143);
new Thread(p).start();
```

Special LIBs (2): Timer

- `Timer`
  - A facility for Threads to schedule tasks for future execution in a background `Thread`
  - used similar to `Timer().schedule(timerTask, delay)`
  - several alternatives to schedule a task (`Date` etc.)
- `TimerTask`
  - A task that can be scheduled for one-time or repeated execution by a `Timer`
  - is a `public abstract class TimerTask implements Runnable`

Alternative Alert

```
import java.util.*;
import javax.microedition.lcdui.*;

public class MyAlert extends Canvas {
  private Display display = null;
  private Displayable next = null;
  private Timer timer = new Timer();

  public MyAlert(Display display, Displayable next) {
    this.display = display;
    this.next = next;
    display.setCurrent( this );
  }

  protected void keyPressed(int keyCode) { dismiss(); }

  protected void paint(Graphics g) { /* put drawings here */ }

  protected void pointerPressed(int x, int y) { dismiss(); }

  protected void showNotify() { timer.schedule( new CountDown(), 5000 ); }

  private void dismiss() {
    timer.cancel();
    display.setCurrent( next );
  }

  private class CountDown extends TimerTask {
    public void run () { dismiss(); }
  }
}
```

Exercise 5:

- Analog Clock

But: MIDP 1.0 + CLDC 1.0

you may use
`System.getCurrentMillis()`
to show the *real* time