How to create custom fonts in Java ME

This tutorial shows how to create a custom font and use it to draw text in Java ME

Note that all code and comments have been taken from information in this discussion board topic by Graham Hughes.

Draw the font

Lay the characters out in a grid. Don't put them in one long, thin strip, as some devices have problems with images that have one, very large dimension.

To simplify things, each cell of the grid is the same size.

Your image should look something like:

```
1234567890!" '
*()-.,:/ABCDEF
GHIJKLMNOPQRST
UVWXYZabcdefgh
ijklmnopqrstuv
wxyzАБВГДЕЁЖЗИ
ЙКЛМНОПРСТУФХЦ
ЧШЩЪЫЬЭЮЯабвгд
еёжзийклмнопрс
туфхцчшщъыьэюя
```

Make sure the background is transparent. Use 8-bit PNG format for best compatibility across devices.

Note that I've left a space (between the " and ' on the top row). You can have separate code later to handle the space as a special condition, but this is easier.

Defining the font parameters

A string holds a list of characters, in the order that they appear in the image. Use unicode escapes for non-ASCII characters, or someone in another country (with a different character encoding) will get different results.

This information should really be loaded from some kind of font definition file, but for this example, they'll go in the code.

```
private static int CELL_WIDTH = 12;
private static int CELL_HEIGHT = 16;

private static String sequence = "1234567890!\" '*()-.,:/"
        + "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        + "abcdefghijklmnopqrstuvwxyz"
        + "\u0410\u0411\u0412\u0413\u0414\u0415\u0401\u0416"
        + "\u0417\u0418\u0419\u041A\u041B\u041C\u041D\u041E"
        + "\u041F\u0420\u0421\u0422\u0423\u0424\u0425\u0426"
        + "\u0427\u0428\u0429\u042A\u042B\u042C\u042D\u042E"
        + "\u042F\u0430\u0431\u0432\u0433\u0434\u0435\u0451"
        + "\u0436\u0437\u0438\u0439\u043A\u043B\u043C\u043D"
        + "\u043E\u043F\u0440\u0441\u0442\u0443\u0444\u0445"
        + "\u0446\u0447\u0448\u0449\u044A\u044B\u044C\u044D\u044E\u044F";

private static Image fontImage;
private static int width;
private static int height;
private static int charsPerRow;

public static void inialise(String fontname) throws IOException {
    fontImage = Image.createImage(fontname);
    width = fontimage.getWidth();
    height = fontimage.getHeight();
    charsPerRow = width / CELL_WIDTH;
}
```

Drawing a character

Using `setClip()`, you can draw any character you like.

I've made this code throw an exception if the characters is not in the font, so you know quickly if you have a missing character. For a production build, you might prefer to draw some place-holder character instead.

```
/** @return width of the character */
private static int drawChar(Graphics g, char ch, int x, int y) {
    // find the position in the font
    int i = sequence.indexOf(ch);
    if (i == -1) {
        throw new IllegalArgumentException("unsupported character");
    }

    // find that character in the image
    int cx = (i % charsPerRow) * CELL_WIDTH;
    int cy = (i / charsPerRow) * CELL_HEIGHT;

    // draw it
    g.setClip(x, y, CELL_WIDTH, CELL_HEIGHT);
```


(C) Copyright Nokia 2013. All rights reserved.

```
    g.drawImage(fontimage, x - cx, y - cy, Graphics.TOP | Graphics.LEFT);

    return CELL_WIDTH;
}
```

Drawing a string

Once you can draw a character, you can draw a string.

```
private static void drawString(Graphics g, String s, int x, int y) {
    // this is faster than using s.charAt()
    char[] chs = s.toCharArray();
    for (int i = 0; i < chs.length; i++) {
        x += drawChar(g, chs[i], x, y);
    }
}
```

Improvements

This, of course, will only handle a fixed-width font. Support a proportional-width font is just a matter of adding an array of character widths. The last line of `drawChar()` then becomes:

```
return characterWidth[i];
```

This implementation also changes the current clip region, which is different behaviour from the API implementation of `Graphics.drawString()`.

Saving the clip region is a simple matter of:

```
int clipX = g.getClipX();
int clipY = g.getClipY();
int clipW = g.getClipWidth();
int clipH = g.getClipHeight();
```

And restoring it:

```
g.setClip(clipX, clipY, clipW, clipH);
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.