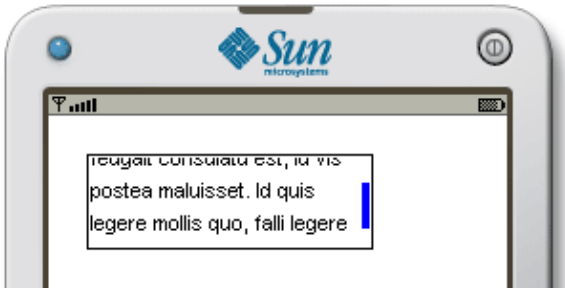




Here is a J2ME example showing how scrollable text can be implemented. (Will post full midlet source code soon)



This is a sample midlet showing the following code in action: [Media:ScrollableTextField.zip?](#), that you can try also with the [emulator on this page](#)

We start defining some customizable variables, to define the layout of our text element:

```
static final int SCROLL_STEP = 25;

int scrollbarWidth = 4;
int scrollbarHeight = 0;
int scrollbarTop = 0;
int scrollbarColor = 0x0000ff;

int borderWidth = 1;
int borderColor = 0x000000;
int bgColor = 0xffffffff;

Font textFont = Font.getDefaultFont();
int textColor = 0x000000;

int padding = 1;
int interline = 2;
```

The we define some variable that will be used internally:

```
static final String VOID_STRING = "";
static final char SPACE_CHAR = ' ';

int width = 0;
int height = 0;
int innerWidth = 0;
int innerHeight = 0;

int currentY = 0;
int textHeight = 0;

String[] textRows = null;
```

Now, let's define a simple constructor that accept a width and a height as parameters:

```
public ScrollableTextFieldExt(int width, int height)
{
    this.width = width;
    this.height = height;
```

Scrollable_Text_in_Java_ME

```
this.innerWidth = width - 2 * borderWidth - 2 * padding - scrollbarWidth;
this.innerHeight = height - 2 * borderWidth - 2 * padding;
}
```

Now, it's time to set some text into this element, don't you think? Here is the method:

```
public void setText(String text)
{
    this.textRows = getTextRows(text, textFont, innerWidth);

    this.textHeight = textRows.length * (interline + textFont.getHeight());

    scrollbarHeight = Math.min(innerHeight, innerHeight * innerHeight / textHeight);

    scrollbarTop

    currentY
}
}
```

And let's manage the scrolling of this element, with these methods:

```
public void scrollDown()
{
    (SCROLL_STEP);
}
public void scrollUp()
{
    (-SCROLL_STEP);
}
private void scroll(int delta)
{
    currentY += delta;

    if(currentY < 0)
    {
        = 0;    currentY
    }
    else if(currentY > textHeight - innerHeight)
    {
        = Math.max(0, textHeight - innerHeight);
    }

    scrollbarTop = innerHeight * currentY / textHeight;
}
}
```

The **getTextRows()** method will return the text rows, splitted accordingly to given **Font** and **width**. A possible implementation is the following (note that this implementation will not split single words, even if wider than the given maximum width):

```
public static String[] getTextRows(String text, Font font, int width) {
    char spaceChar = ' ';

    //will contain text rows
    Vector rowsVector = new Vector();

    //will contain current row text
    StringBuffer currentRowText = new StringBuffer();

    //indexes used to split text words
```

Scrollable_Text_in_Java_ME

```
int prevIndex = 0;
int currIndex = text.indexOf(spaceChar);

if(currIndex == -1)
    currIndex = text.length();

//will hold widths of current row and token
int rowWidth = 0;
int tokenWidth = 0;

//width of a single whitespace
int whitespaceWidth = font.stringWidth(" ");

//current text token
String currentToken = null;

while (currIndex != -1) {
    //get the current token
    currentToken = text.substring(prevIndex, currIndex);

    //get the width of current token..
    tokenWidth = font.stringWidth(currentToken);

    //..and update row width
    rowWidth += tokenWidth;

    //if row is not empty, add the whitespace width too
    if (currentRowText.length() > 0) {
        rowWidth += whitespaceWidth;
    }

    //if new row width is bigger than max width, and previous row is not empty
    if (currentRowText.length() > 0 && rowWidth > width) {
        //add current row text to rows Vector
        rows.addElement(currentRowText.toString());

        //reinitialize current row with current token
        currentRowText.setLength(0);
        currentRowText.append(currentToken);

        //and update current row width
        rowWidth = tokenWidth;
    } else {
        //if current row is not empty, add a whitespace
        if (currentRowText.length() > 0)
            currentRowText.append(spaceChar);

        //and then add current token
        currentRowText.append(currentToken);
    }

    //check if text is ended
    if (currIndex == text.length())
        break;

    //update indexes
    prevIndex = currIndex;
    currIndex = text.indexOf(spaceChar, prevIndex);

    if (currIndex == -1)
        currIndex = text.length();
}
```

Scrollable_Text_in_Java_ME

```
}

//finally append current row, if not empty
if (currentRowText.length() > 0) {
    addElement(currentRowText.toString());
}

//Convert our rows vector to a String array
String[] rowsArray = new String[rowsVector.size()];

rowsVector.toArray(rowsArray);

return rowsArray;
}
```

And, finally, we should paint this element :)

```
public void paint(Graphics g)
{
    setColor(borderColor);
    fillRect(0, 0, width, height);

    setColor(bgColor);
    fillRect(borderWidth, borderWidth, width - 2 * borderWidth, height - 2 * borderWidth);

    setColor(textColor);
    setFont(textFont);

    translate(borderWidth + padding, borderWidth + padding);

    setClip(0, 0, innerWidth, innerHeight);

    if(textRows != null)
    {
        for(int i = 0; i < textRows.length; i++)
        {
            drawString(textRows[i], 0, i * (textFont.getHeight() + interline) - currentY, Graphics.TOP | Graphics.LEFT);
        }
    }

    setClip(0, 0, width, height);

    setColor(scrollbarColor);
    fillRect(innerWidth, scrollbarTop, scrollbarWidth, scrollbarHeight);

    translate(- (borderWidth + padding), - (borderWidth + padding));
}
```