Como criar uma entrada de texto personalizada, em Java ME

Aqui está um simples MIDlet: Media:CustomInputMidlet.zip

Aqui está um básico exemplo mostrando como criar uma entrada de texto personalizada, usando Canvas, em Java ME, coisas que muitas das vezes é necessario quando se está trabalhando com gráficos de baixo nível(por exemplo jogos).

Neste código você pode:

- definir qual mapa de caracteres para cada tecla
- definir o itervalo de piscagem
- definir o intevalo máximo entre subsequentes teclas pessionadas
- mover esquerda/direita, dentro dos limites o texto
- apagar o texto

Muitas das características não estão presentes, logo se você quiser implementá-las agradeço :)

```
package com.jappit.custominput.screen;

import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;

public class CustomInputCanvas extends Canvas implements Runnable
{
 static final char[] KEY_NUM1_CHARS = {'.', '?', '!'};
 static final char[] KEY_NUM2_CHARS = {'a', 'b', 'c'};
 static final char[] KEY_NUM3_CHARS = {'d', 'e', 'f'};
 static final char[] KEY_NUM4_CHARS = {'g', 'h', 'i'};
 static final char[] KEY_NUM5_CHARS = {'j', 'k', 'l'};
 static final char[] KEY_NUM6_CHARS = {'m', 'n', 'o'};
 static final char[] KEY_NUM7_CHARS = {'p', 'q', 'r', 's'};
 static final char[] KEY_NUM8_CHARS = {'t', 'u', 'v'};
 static final char[] KEY_NUM9_CHARS = {'w', 'x', 'y', 'z'};
 static final char[] KEY_NUM0_CHARS = {' '};

 int clearKeyCode = Integer.MIN_VALUE;

 StringBuffer currentText = new StringBuffer();
 String currentString = new String();

 int lastPressedKey = Integer.MIN_VALUE;
 int currentKeyStep = 0;

 Font inputFont = null;
 int inputWidth = 0;
 int inputHeight = 0;
 int inputTranslationX = 0;

 long lastKeyTimestamp = 0;
 long maxKeyDelay = 500L;

 int caretIndex = 0;
 int caretLeft = 0;
 boolean caretBlinkOn = true;
 long caretBlinkDelay = 500L;
 long lastCaretBlink = 0;

 boolean goToNextChar = true;

 public CustomInputCanvas()
 {
  new Thread(this).start();

  inputFont = Font.getDefaultFont();

  inputWidth = getWidth();

  inputHeight = inputFont.getHeight();
 }

 public char[] getChars(int key)
 {
  switch(key)
  {
  case Canvas.KEY_NUM1: return KEY_NUM1_CHARS;
  case Canvas.KEY_NUM2: return KEY_NUM2_CHARS;
  case Canvas.KEY_NUM3: return KEY_NUM3_CHARS;
  case Canvas.KEY_NUM4: return KEY_NUM4_CHARS;
  case Canvas.KEY_NUM5: return KEY_NUM5_CHARS;
  case Canvas.KEY_NUM6: return KEY_NUM6_CHARS;
  case Canvas.KEY_NUM7: return KEY_NUM7_CHARS;
  case Canvas.KEY_NUM8: return KEY_NUM8_CHARS;
  case Canvas.KEY_NUM9: return KEY_NUM9_CHARS;
  case Canvas.KEY_NUM0: return KEY_NUM0_CHARS;
  }
  return null;
 }
 boolean isClearKey(int key)
 {
  return key == -8;
 }

 void clearChar()
```


(C) Copyright Nokia 2013. All rights reserved.

```
{
 if(currentText.length() > 0 && caretIndex > 0)
 {
  caretIndex--;

  currentText.deleteCharAt(caretIndex);

  currentString = currentText.toString();
 }
}
void updateCaretPosition()
{
 caretLeft = inputFont.substringWidth(currentString, 0, caretIndex);

 if(caretLeft + inputTranslationX < 0)
 {
  inputTranslationX = - caretLeft;
 }
 else if(caretLeft + inputTranslationX > inputWidth)
 {
  inputTranslationX = inputWidth - caretLeft;
 }
}
public void keyPressed(int key)
{
 int gameAction = getGameAction(key);

 System.out.println("KEY: " + key + ", " + gameAction);

 if(isClearKey(key))
 {
  clearChar();

  updateCaretPosition();

  goToNextChar = true;
 }
 else if(key >= KEY_NUM0 && key <= KEY_NUM9)
 {
  writeKeyPressed(key);
 }
 else if(gameAction == Canvas.LEFT)
 {
  if(caretIndex > 0)
  {
   caretIndex--;

   updateCaretPosition();

   goToNextChar = true;
  }
 }
 else if(gameAction == Canvas.RIGHT)
 {
  if(caretIndex < currentText.length())
  {
   if(goToNextChar)
    caretIndex++;

   updateCaretPosition();

   goToNextChar = true;
  }
 }
}
public void writeKeyPressed(int key)
{
 if(goToNextChar || key != lastPressedKey)
 {
  goToNextChar = true;

  lastPressedKey = key;

  currentKeyStep = 0;
 }
 else
 {
  currentKeyStep++;
 }

 char[] chars = getChars(key);

 if(chars != null)
 {
  if(currentKeyStep >= chars.length)
  {
   currentKeyStep -= chars.length;
  }
  if(goToNextChar)
  {
   currentText.insert(caretIndex, chars[currentKeyStep]);

   caretIndex++;
  }
  else
  {
   currentText.setCharAt(caretIndex - 1, chars[currentKeyStep]);
  }
  currentString = currentText.toString();

  updateCaretPosition();

  lastKeyTimestamp = System.currentTimeMillis();

  goToNextChar = false;
 }
}

public void checkTimestamps()
{
 long currentTime = System.currentTimeMillis();

 if(lastCaretBlink + caretBlinkDelay < currentTime)
 {
  caretBlinkOn = !caretBlinkOn;

  lastCaretBlink = currentTime;
 }

 if(!goToNextChar && lastKeyTimestamp + maxKeyDelay < currentTime)
```


(C) Copyright Nokia 2013. All rights reserved.

```
    {
     goToNextChar = true;
    }
   }

   public void run()
   {
    while(true)
    {
     checkTimestamps();

     repaint();

     try
     {
      synchronized(this)
      {
       wait(50L);
      }
     }
     catch(Exception e)
     {
      e.printStackTrace();
     }
    }
   }

   public void paint(Graphics g)
   {
    g.setFont(inputFont);

    g.setColor(0xffffff);

    g.fillRect(0, 0, getWidth(), getHeight());

    g.setColor(0x000000);

    g.translate(inputTranslationX, 0);

    g.drawString(currentString, 0, 0, Graphics.LEFT | Graphics.TOP);

    if(caretBlinkOn && goToNextChar)
    {
     g.drawLine(caretLeft, 0, caretLeft, inputHeight);
    }
    g.translate(- inputTranslationX, 0);
   }
  }
```


(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.