# How the Text in Writer Gets on the Screen?

Jan Holesovsky <kendy@collabora.com>

## It all starts with a draw request

- Window::ImplCallPaint(
  - const Region* pRegion,
    - (can be more rectangles etc.)
  - sal_uInt16 nPaintFlags )
    - (whether to paint children etc.)

## It gets to the Writer's edit window

- SwEditWin = Window class for the Writer edit area
  - handling mouse and keyboard events and doing the final painting of the document from the buffered layout.
- SwEditWin::Paint(
  - const Rectangle& rRect)
    - (rectangle to repaint)

## SwCrsrShell – ancestor of SwWrtShell

- SwWrtShell is used by the UI to modify the document model
- SwCrsrShell::Paint(
    - const Rectangle &rRect)

## SwViewShell – ancestor of SwCrsrShell

- SwViewShell::Paint(
  - const Rectangle &rRect)
- The “real” drawing starts here
  - Toplevel – draws the shadows around the document etc.
  - Very ugly, actually – part of the code in the class, part is global in the .cxx
  - Many OutputDevices out there, etc.

# Now we are getting to the document model

- SwRootFrm – the root element of a Writer document layout
- SwRootFrm::Paint(
  - SwRect const& rRect,
    - Rectangle to paint
  - SwPrintData const*const pPrintData) const
    - Gets NULL here

# Getting deeper into the model

- Calling hierarchically (more times in the backtrace)
- SwLayoutFrm::Paint(
  - SwRect const& rRect,
  - SwPrintData const*const) const

## Finally we got to the text frame

- SwTxtFrm::Paint(
  - SwRect const& rRect,
  - SwPrintData const*const) const
- We split the frame to lines

## And then split the line to portions

- SwTxtPainter::DrawTextLine(
  - const SwRect &rPaint
    - (rectangle to paint)
  - SwSaveClip &rClip,
    - (clipping)
  - const sal_Bool bUnderSz )
    - (paint the entire line, or by portions?)

## And now “only” draw the portions

- SwTxtPortion::Paint(
  - const SwTxtPaintInfo &rInf ) const
- SwTxtPaintInfo::DrawText(
  - const SwLinePortion &rPor,
  - const sal_Int32 nLength,
  - const sal_Bool bKern ) const
  - (just a wrapper for the next one)

## “Just do it” kind of method

- SwTxtPaintInfo::_DrawText(
  - const OUString &rText,
  - const SwLinePortion &rPor,
  - const sal_Int32 nStart,
  - const sal_Int32 nLength,
  - const sal_Bool bKern,
  - const sal_Bool bWrong,
  - const sal_Bool bSmartTag,
  - const sal_Bool bGrammarCheck )

## Getting closer to actual drawing

- SwFont::_DrawText(
  - SwDrawTextInfo &rInf)
  - (just a wrapper)
- SwSubFont::_DrawText(
  - SwDrawTextInfo &rInf,
  - const sal_Bool bGrey )
  - (takes care of the underlining, etc.)

## Compute the positions of the glyphs

- SwFntObj::DrawText(
  - SwDrawTextInfo &rInf )

## And finally – draw the text!

- OutputDevice::DrawTextArray(
  - const Point& rStartPt,
  - const OUString& rStr,
  - const sal_Int32* pDXAry,
    - (offsets of the letters)
  - sal_Int32 nIndex,
  - sal_Int32 nLen )