Unifying LibreOffice Low Level Text Layout

- Khaled Hosny
- Akash Jain (autoit)
How text is laid out?

- Handled by VCL, other parts of LibreOffice just talk to it.
- Interface is provided by OutputDevice class:
  - `DrawText()`, `DrawTextArray()`, `GetTextWidth()`, `GetTextHeight()`, etc.
- Which in turn uses `SalLayout` class to do the actual layout:
- `SalLayout`, however, doesn’t do much by itself, actual layout is done by its platform-specific, font technology specific subclasses.
Problem

Unifying LibreOffice Low Level Text Layout

SalLayout

GraphiteLayout

ServerFontLayout

GraphiteServerFontLayout

GenericSalLayout

PspFontLayout

PspServerFontLayout

WinLayout

SimpleWinLayout

UniscribeLayout

GraphiteWinLayout

CTLayout
Problem: complexity

- 3 classes on Windows, simple, OpenType and Graphite.
- 4 classes on Unix, OpenType, Graphite, and 2 just for PostScript printing.
- 1 class on macOS, but no Graphite support.
- Now imagine how much work is needed to add support for something as simple as user controllable font features.
Problem: inconsistency

- Different code paths on different platform, or even for different writing systems, with little to no code sharing.
- Language/writing systems support varies between platform APIs.
- It varies even between different versions of the same platform.
- Different platform APIs allow for different levels of integration.
Solution

How standards proliferate:
(see: A/C chargers, character encodings, instant messaging, etc)

**Situation:**
There are 14 competing standards.

14?! Ridiculous!
We need to develop one universal standard that covers everyone's use cases.
Yeah!

**Soon:**

**Situation:**
There are 15 competing standards.
GSoC project

- Unify text layout across platforms.
- By Akash Jain, all the credit goes to him, and the blame too.
- Proposed and mentored by Khaled Hosny, me, but I deny everything.
- Also part of TDF-funded project for reliable cross-platform layout testing.
GSoC project: idea

- One class to rule them all.
- Based mostly on the existing Unix implementation.
- Uses HarfBuzz for everything; OpenType, Graphite and even AAT on macOS.
- Font loading is still done by platform APIs, for better integration.
GSoC project: rationale

Why HarfBuzz:
- Well tested, actively maintained, feature-complete, cross-platform, and free software library.
- Tracks latest versions of Unicode and OpenType standards.
- We get the same high quality output regardless of the platform supports natively.
- Used by Firefox and Chrome in all platforms, so we are in good company.
A new class, CommonSalLayout. Naming is really hard!

Code integrated with font and graphics libraries on all supported platforms. Can be switched on/off at run time.

Some issues remain:
- There seem to be some performance regressions.
- Font fallback does not work on macOS.
- Some Graphite fonts are broken (e.g. Awami font), GenericSalLayout::ApplyDXArray() I’m looking at you!
- Non-SFNT (e.g. Type 1) fonts are not currently supported.
- Controlling font features is not yet supported.
- Windows XP is not supported.
- Code should land on master soonish.
Soon

**Unifying LibreOffice Low Level Text Layout**

- GraphiteLayout
  - ServerFontLayout
    - GraphiteServerFontLayout
  - PspServerFontLayout
- GenericSalLayout
- WinLayout
  - SimpleWinLayout
  - UniscribeLayout
    - GraphiteWinLayout
- CTLayout
- CommonSalLayout
- SalLayout
  - MultiSalLayout
Wishful thinking

CommonSalLayout → SalLayout → MultiSalLayout

GenericSalLayout
Even more wishful thinking
Daydreaming

SalLayout
Thank you!

⚠ Special thanks to TDF for sponsoring my work on this project and allowing me to attend this conference.