Collabora

LibreOffice
The Document Foundation

SUSE

Implementing Vulkan-capable drawing using the Skia library

Collabora Office

By Luboš Luňák
Software Engineer at Collabora Productivity

OPENSUSE-LIBREOFFICE CONF'20

2020

Graphics in LibreOffice

Basic overview

Different LO libraries

- Low-level: VCL
  - Basic graphics operations, widgets
- High-level: Drawinglayer
  - Representation of graphics primitives
  - Uses “processors” to draw primitives using VCL
- Others
  - Canvas – UNO-based, more modern, meant-to-replace VCL
    - Almost unused
  - ??? (it’s complicated, competing designs, unfinished rewrites,...)

VCL (LibreOffice Visual Class Library)

- Widgets (buttons, checkboxes, …)
- Basic rendering (lines, rectangles, gradients, ...)
- Graphic backends: Windows, gen, Qt, Gtk, Quartz, headless
- Some backends themselves have several implementation backends:
  - Windows → GDI / OpenGL
  - gen → X11 / OpenGL

VCL problems

Old design

- 1bpp, 4bpp bitmaps
- Paletted bitmaps
- Transparency vs opacity
- Separate alpha channel (24+8bpp vs 32bpp bitmaps)
- Graphics operations done by explicitly writing the code
- Graphics operations done on the CPU
- API reflects these design choices

VCL problems #2

- => slow, error prone, complicated code
- We are office suite developers, not graphics library developers

OpenGL VCL backend

Attempt to improve VCL graphics

- GPU-accelerated
- Operations done by dedicated code (instead of “anywhere”)
- More modern concepts

Problems

- Requires working HW/drivers, no fallback (other than other VCL backend)
- Graphics operations still done by explicitly written code

Skia VCL backend

Skia Library

- Modern 2D graphics rendering library
- Different drawing backends
  - CPU-based (‘Raster’)
  - Vulkan
  - Others (OpenGL, D3D, Metal)
- Good performance
- Multiplatform
- Powerful yet reasonably simple C++ API

VCL backend basic parts

Bitmap representation

- Stores image information (pixels)

Graphics operations

- Performs drawing and stores the result

Instance

- Creates objects, book-keeping, …

Others …

SkiaSalBitmap

- Stores pixels, palette, color-depth, size, …
- Allows access to these bitmap data
- Converts to Skia formats for use

SkiaSalGraphicsImpl

- Stores result of drawing (not SkiaSalBitmap, but can be converted to)
- Draws the result to the screen (if window-based and not offscreen)
- Draws bitmaps
- Draws other primitives (lines, gradients, polygons, text,...)

Problems&solutions #1

Skia does not support old formats (1bpp, 4bpp, palettes)

- Need to be stored twice and converted
- Better: Outside code should accept bitmap's preferred format
- Currently: For huge bitmaps (and raster or low memory), only one format is kept and converted to other on-demand

Problems&solutions #2

Bitmaps usually give pixel access to outside code to perform an operation

- Slow, especially fetching GPU-stored pixels
- No or difficult caching
- Explicitly written external code
- No encapsulation, tedious to change (e.g. transparency vs opacity)
- Better: Common operations done directly by the bitmap
  - Scale, blend, convert, …
  - Already started by the OpenGL backend

Problems&solutions #3

Separate alpha channel

- ARGB is stored as 24bpp RGB bitmap and 8bpp bitmap
- Uses transparency (inverse of opacity, the usual alpha format)
- For drawing these need to be blended back
- SalGraphics uses additional internal alpha SalGraphics
- Slow, wastes memory, complicated
- Better: Store as 32bpp, do not separate (unless needed)

Problems&solutions #4

Higher-level LibreOffice code performs graphics operations itself

- Gradients converted to polygons
- Line dashing converted to polygons
- Bitmap blending done in loops pixel by pixel
- => slow, error-prone, complicated, have to maintain the code
- Better: Implement in SalBitmap and SalGraphics, improve VCL API
  - Already started by the OpenGL backend

Present and Future

Current status

- Implemented for Windows and Linux (the ‘gen’ backend)
- Default on Windows.
- Vulkan is default, if available, with a fallback to Raster if there are problems.
- So far no big problems (it seems, hopefully :)).
- Passes all VCL unittests (unlike all other VCL backends except for headless).
- Already outperforms other backends in many cases, even in Raster mode.

Future

- More bugfixing, including performance fixes/improvements.
- ?
  - Depends also on others. Contributions are welcome.
- Improvements mentioned in this talk.
- Support for more platforms (Qt5/KF5?)
- ...

More Information

- vcl/skia/README
- http://skia.org

Thank you.

- More information:
  - vcl/skia/README
  - https://skia.org
  - l.lunak@collabora.com

Work sponsored by