REPLACEMENT OF LIBREOFFICE SVG FILTER IN FAVOR OF SVGIO

CIB SOFTWARE GMBH
TIRANA, FRIDAY 27. SEPTEMBER 2018

ARMIN LE GRAND
CONTENT

1. WELCOME!
2. MOTIVATION
3. PLANNING
4. GET RID OF THE DOCUMENT-SVG IMPORTER
5. REPLACE WITH SVGIO-BASED IMPORTER
6. CAVEATS
7. ADDITIONAL BENEFITS
8. TODO
WELCOME!

> This talk describes the actions taken to replace the existing LibreOffice SVG filter in favour of SVGIO.
> It will describe the motivation and reasons behind it, the pros and cons and the technical steps taken to do so.
> It will explain the advantages and the achieved progress in doing this.
> It will also contain a live experimental part to present the now existing turn-arounds and quality achievements when using the newly implemented Filter based on SVGIO.
WELCOME!

Thanks go to TDF & their donors

...for sponsoring this work!
**MOTIVATION**

- There is a SVG Import, why change it at all?

- **State of SVG Import(s) before the change**
  - There were two different SVG ‘Imports’
    - Opening a SVG as Document → OpenAsDocument
    - Inserting a SVG Graphic into a Document → Insert
  - Handling different tasks (Document/inserted Graphic)
  - Using different methods to Import (for historical reasons)
  - Leading to different results

- **Problems**
  - Inconsistencies (User View, different quality)
  - Two Importers to maintain (Developer View)
**MOTIVATION**

- **OpenAsDocument Import technique used:**
  - Based on regular ODF importer
  - Creates DOM-Tree in ASCII on demand
  - Works like a Unix-Pipe

- **Import technique used when inserting as Graphic:**
  - Uses separated SVG DOM-Tree import
  - Creates Sequence of Primitives
    - e.g. adding special 'SVG-Gradients'
  - Keeps and holds original data (exports, re-interpret)
  - On-demand interpretation/decomposition
MOTIVATION

• Do we really need two Importers for SVG?
  – Users do not understand the difference
  – Double work all the time (Developer View)
  – Risk of ‘influences’ to regular ODF importer

• How to get to a possible SVG turn-around?
  – Needs a SVG Exporter
  – There is a SVG exporter for Draw and Impress
  – Too much work for one change to also change that
  – Even Multi-Page support – somewhat SVG1.2
  – Combined with massive added JavaScript stuff for creating a Presentation-like Export
**PLANNING**

- To solve...
  - Get rid of the Document-SVG Importer
  - Replace with SVGIO-based Importer
  - Maintain the Multi-Page setup
  - Do not touch the existing Exporter (anyone...?)
  - Capability to create Draw or Impress Documents
  - As-good-as-possible turn-arounds
    - Page-size
    - Quality
    - Keep original Data (?)
  - Doable in a given Time-Frame
GET RID OF THE DOCUMENT-SVG IMPORTER

• Systematically strip code
  – Lots of experience doing this (AW080)
  – Let the compiler help you :-)
  – Try to identify all unused stuff (not easy)

• Keep Import filter

• Do not hurt basic starting points in code
• Re-Use Import filter, but do different things
• Create SVG-Primitive
  − Contains the SVG as ’data-blob’
  − Decomposes to Sequence of Primitives
• Read SVG as single SVG-Primitive
• Get the Size
  − Primitives already support getting the B2DRange
  − Needed some squeezing to speedup
    • Can directly create B2DRange Info from SVG Header
• Based on Size, create a Document
  − Maybe create a Draw or Impress Document
REPLACE WITH SVGIO-BASED IMPORTER

- Insert a SdrGraphicObject to the 1st Page of that Document
- Adapt to Size, Position it
- Adapt PageSize, take PageBorders into account
- Set the Imported SVG Graphic as content at the SdrGraphicObject

- At this point, the SVG is still not interpreted – not necessary yet :-)
CAVEATS

• How to detect from the currently written SVG if it is Draw or Impress?
  - Don’t ask, but it’s possible by identifying some nodes in DOM-Tree
  - Lead to abstract/unify the SVG FilterDetector to also do this job if needed → Output adapted to more than just detected Type

• How to then create the correct document type?
  - Due to detecting in Filter possible now at the right time
  - Needs technically two different filter entries
  - Using the same TypeDetector, but triggering different Filters (which use the same implementation)
CAVEATS

• How to detect Multi-Page?
  - Needs a PrimitiveProcessor deep-diving to the imported SVG – in most cases, the 1st 500 bytes are enough
  - Thanks to Primitives, encapsulation to MultiPage-Parts is possible
  - BTW: A problem that needs unification, also for Bitmap-Graphics (GIF, Multi-Page TIFFs – FAX, ...)

• How to create Multi-Page?
  - Need to ‘split’ imported SVG
  - On SVG-Level or Primitive-Level?
  - Missing part – ran out of time, but Idea developed
  - Current Import is in one Page only
ADDITIONAL BENEFITS

• Unify Vector-Based Importers
  - SVG already was isolated in an own module
  - Why not do the same for GdiMetafile/GDI+
    • Just did that, moved and isolated all that old code
  - SVG already used UNO API Isolation and original Data Buffering
  - Why not do the same for GdiMetafile/GDI+
    • Did that, too

• Vector-Based Input Formats are now all unified

• Result is always a Sequence of Primitives
ADDITIONAL BENEFITS

• Old code adaption: A PrimitiveProcessor extracting the GdiMetaFile (if needed – not for paint :-)

• Done using a GdiMetafilePrimitive → Decomposition is the Sequence of Primitives representation

• This is the base for a paradigm change for future GraphicData Importers
  - Always Buffer original Data (wherever, Mem-File, ...)
  - Decompose on-demand
  - Offer access to Primitive representation
  - Be accessible using UNO API
ADDITIONAL BENEFITS

• UNO API concrete:
  - Class BasePrimitive2D is based on css::graphic::Xprimitive2D
  - A Primitive/sequence<Primitive> can be handed over the UNO API
  - XprimitiveFactory2D allows to get sequence<Primitive> from
    - drawing::Xshape
    - drawing::XdrawPage
    - You can import SVG and EMF/WMF/WMF+ using
      - XsvgParser
      - XEmfParser
ADDITIONAL BENEFITS

• UNO API concrete:
  - You can use Xprimitive2DRenderer to get a rasterized version of your sequence<Primitive> based on given parameters
  - With this UNO API tooling you can already write a SVG-to-Bitmap converter or get the containing rectangle of any Xshape/XdrawPage
  - Theoretically you could also implement own Primitives and use, but you can not use existing basic primitives to implement the decomposition
• Add missing support to get real Multi-Page document Turn-Around

• Maybe enhance and unify Import Formats including Bitmap-Data
  – Preserve Original Data
    • Allow save the unchanged GraphicData
    • Allow export (Context Menu)
  – Use as Base for Swapping/TempFiles
  – Use as Base for always having a fast Thumbnail

Lots of Tasks to do – further Love and Support needed :-)
THANK YOU!

OUR PRODUCTS:

HTTP://LIBREOFFICE.CIB.DE/

WE CAN HELP:

HTTP://LIBREOFFICE.CIB.DE/SUPPORT