Porting LibreOffice To GTK3

Caolán McNamara,
Red Hat
2015-09-25

- Demo
- Architecture
- Getting it to fully work
- Wayland tweaks

Demo

Architecture

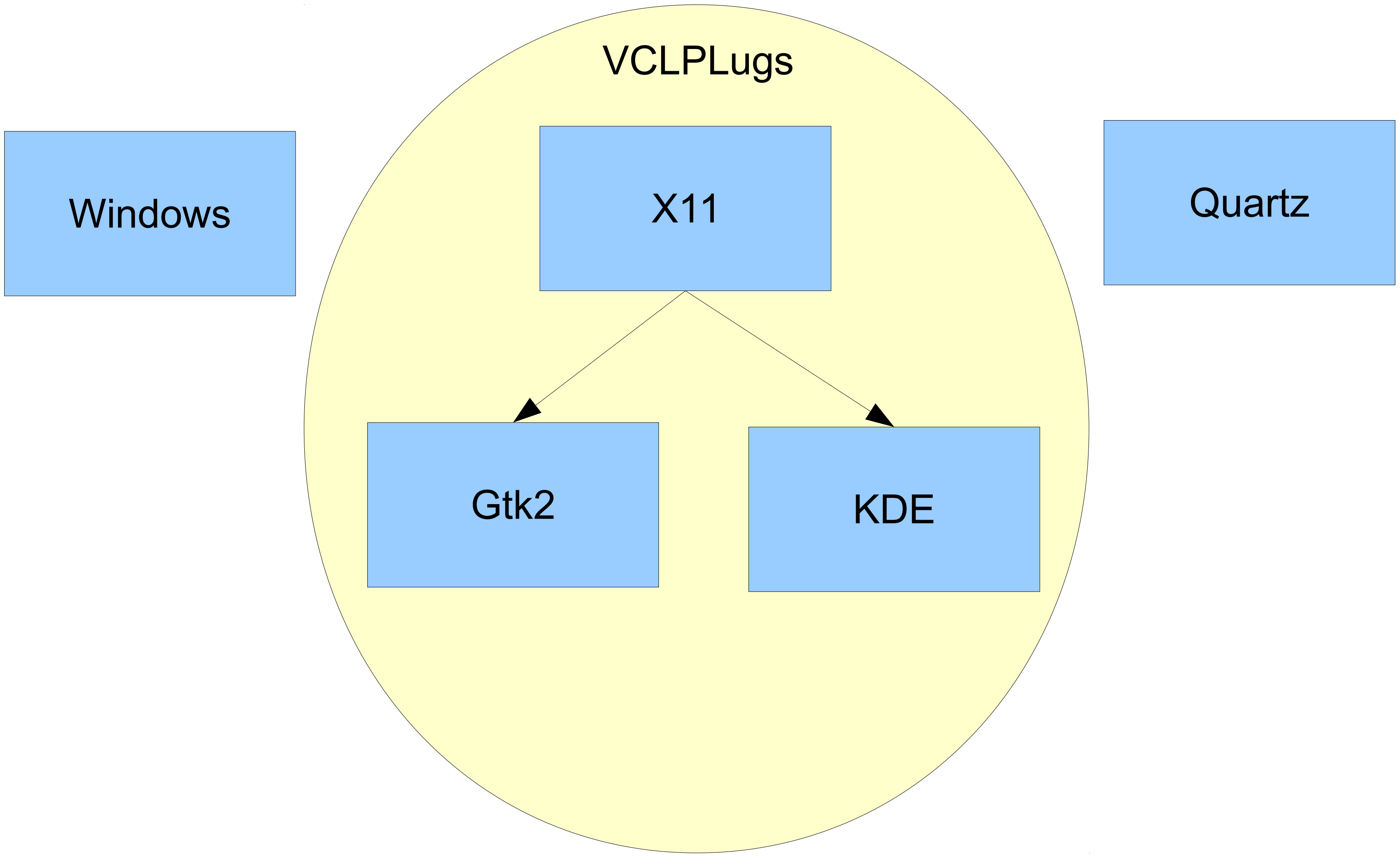
SalInstance, SalFrames

- Each platform has to implement a SalInstance
- A SalInstance mostly consists of Create/Destroy pairs for SalFrames, SalPrinters, SalVirtualDevice, etc.
- Each platform has to provide concrete implementations of SalFrames, SalPrinters and SalVirtualDevices, etc
- SalFrames are system windows (X11 Window)
- SalVirtualDevices are non visible drawables/buffers (X11 Pixmap)

SalGraphics

- SalFrames and SalVirtualDevices must implement AcquireGraphics which returns a SalGraphics
- Each port has to implement a SalGraphics which enables drawing to the SalFrame/SalVirtualDevice
- Apis like drawLine, drawRect
- Some of the drawing apis are optional
- Some of these apis are somewhat “fat”
  - DrawEPS
  - isNativeControlSupported/drawNativeControl for native widget framework

VCL Implementations



Gtk2

- GtkSalGraphics inherited from the X11SalGraphics
  - Mostly reused X11 code, except added native widget support
- GtkSalFrame inherited from X11SalFrame
  - In many places grabbed the underlying xid of the GtkWindow and tweaked it directly
- Printing inherited from generic cups backend
- Entirety of cut-and-paste and draw-and-drop inherited from X11 equivalents.

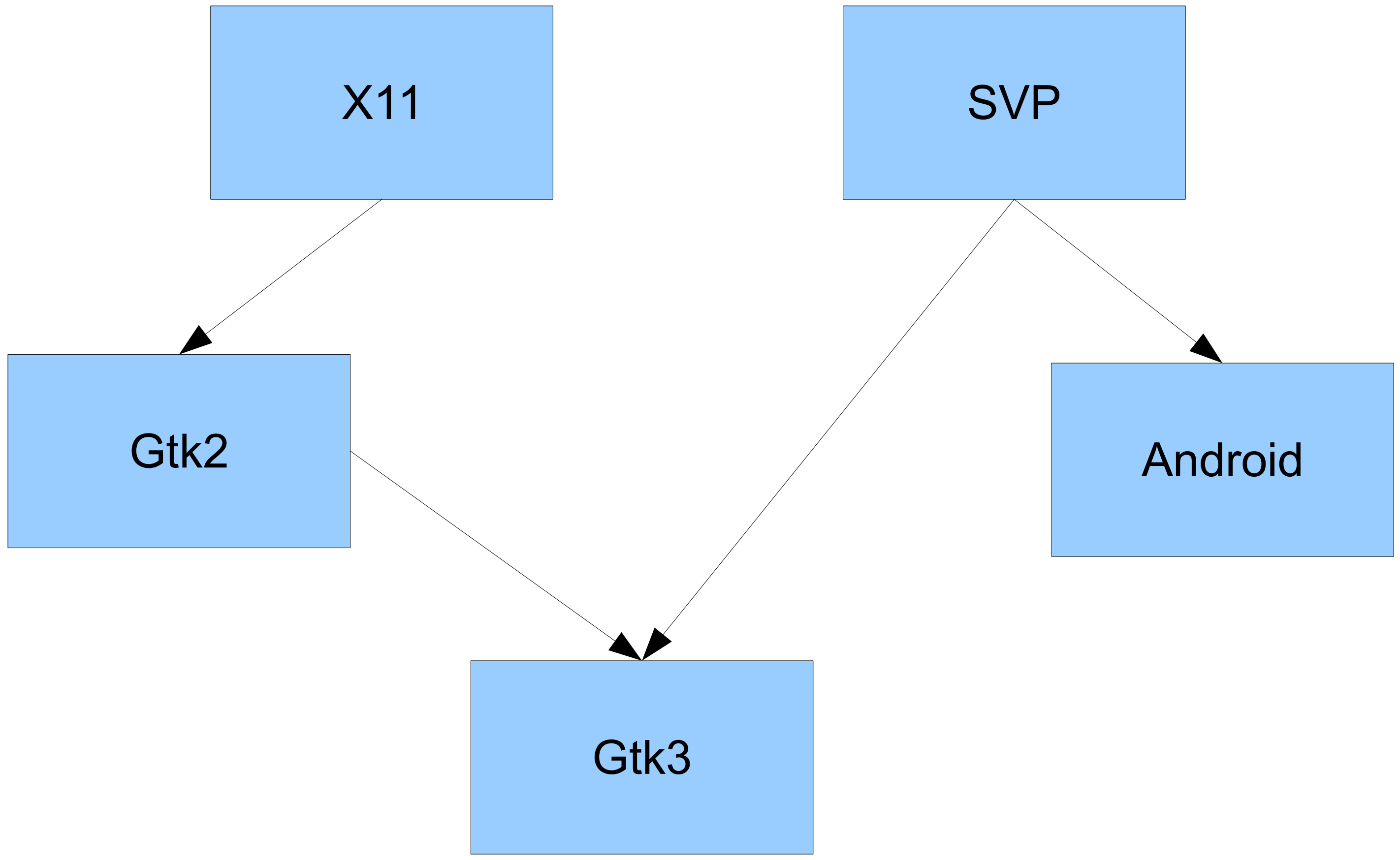
Gtk3

- Obviously lots of overlap with the Gtk2 vclplug, where we didn't just use X directly
- But we need something to back our virtual devices
- And we also can no longer draw directly to windows
- Need a SalGraphics implementation that can draw into those replacements.

Headless/svp

- We have a headless mode
- Originally intended for server applications
  - e.g. Document conversion hubs
- Forms a part of the android port and libreofficekit tiled render work
- Headless mode is implemented as a SalInstance etc
- Implements a virtual device bitmap buffer and a mostly complete SalGraphics impl to render to it

VCL Implementations



Getting it that far

- I had little input into this stage of the effort
- Thanks to:
    - Michael Meeks, Cosimo Cecchi, Lucas Baudin, Ivan Timofeev, Antonio Fernandez, etc for bootstrapping things to that stage

Getting it to work

Getting it to work right 1

- Tweaked the basebmp bitmapbuffer that backs the svp virtual devices to take the same stride as cairo
- Added cairo compatible rgbx formats to basebmp
  - Can use cairo on our basebmp surfaces
  - Can drop converting formats and creating temp buffers to draw native widgets
  - Fix preexisting emf bugs with 32bit bitmap formats!
- Refactor our cairo text rendering to be reusable for the canvas

Getting it to work right 2

- The basebmp backing surface provides damaged events when its modified
  - Route those directly to gtk_widget_queue_draw_area
  - Simplify “draw” to simple copy from backing to draw cairo context
  - Debug damage tracking to death and fix a pile of corner cases
  - Trigger redrawing on resize etc

Rework native widget drawing

- Laboriously reproduce the same sort of native widget rendering we had for gtk2 with gtk3
- Added native focus rectangles
- Lots of the required gtk3 tweaks are similar to pre-existing quartz ones
- Can be tricky to set up the right contexts here, e.g. render menu arrows with menuitem style not menu style. Have to have a peek into gtk itself to see what style and context something is rendered with

Gesture support

- Swipe
- Long Press
- Only a sample use made of these in impress slideshow for now

Auto-mnemonics in menu/menubars

- Thanks to Simon Long from raspberrypi
- Underlines appear on appropiate keystrokes
  - Looks like a real gtk3 app

Other things

- Cut and Paste
  - Gtk2 impl just delegated this to the X11 impl
  - So new impl from scratch, not as scary as feared
- Accessibility.
  - Gtk2 impl had to do some horrific hacks to hook into a11y and capture everything destined for the toplevel window and report back in terms of vcl widgets
  - Gtk3 impl can set the get_accessible member of the first level child of the toplevel window whose gtk2 purpose was just to capture a11y events

Wayland Tweaks

Toplevel Window is Toxic

- gtk_widget_set_double_buffered should really warn/complain with the wayland backend. Blank window.
  - I know the documentation is up to date.
- Connecting to “draw” on the toplevel gives offset results. Presumably there's magic to handle the now special case of a toplevel window
  - Move “draw” down to the previously-only-for-ally widget
- Connecting to mouse events on the toplevel gives unresizable toplevel
  - Stick an eventbox between toplevel and “draw” widget

To Do

- Drag and Drop
- Add gstreamer support for wayland
- More sizing tweaks wrt toplevel window, lots of hackery to undo for gtk3 and clean things up
- Selection rectangles using XOR/Stippling
- 32bit rgbx bitmaps are on the unoptimized paths
- The svp/headless backend doesn't implement various optional but desirable interfaces
  - Help about svgs looks jagged, etc.
  - Improvements to the svp/headless helps gtk3 and android

Possible future stuff

More native stuff

- Could make the menubar and their submenus native gtk menubar and menus
  - There's precedent there wrt the Mac port and the Unity support
- We converted all our dialog, tab page, etc resources over to the gtk3 builder file format
  - I've a feature branch where the message dialogs are native gtk3 dialogs loading those .ui files directly
  - Require a move to gettext probably, concerns about vast .mo files with duplicated english source strings in each one

Thanks for your time