Meeting Notes

The Wakefield committers may hold off-line discussions from time-to-time. Since these meetings are just for the committers and invited wayland experts, summary minutes will be recorded here as time permits

Online Zoom Meeting 9am PST 13th Jan 2022

Attendees : Olivier, Phil, Kevin, Aleksandr, Dalibor, Victor, Niels De Graaf, Alexey Ushakov, Jonas, Mario

he topic of the day was a discussion around Alexey's email to the mailing list ..

https://mail.openjdk.java.net/pipermail/wakefield-dev/2021-December/000025.html

proposing what we might do for the "full" wayland native port and the various points it raised

Note: Alexey will add a fleshed out version on the project wiki.

The proposal suggests we'd code to the low level wayland APIs rather than using a high level toolkit (ie GTK4)

As previously observed we should explore both. Probably we could get bootstrapped a lot faster with GTK4
and it would give flexibility for backends so we'd probably not need to create an OGL and Vulkan backend ..
if Vulkan was an option on that distro then we'd expect the underlying Cairo (?) based renderer to be
using it or have it as an option.

Likely there are pros and cons to both options. GTK4 might make assumptions or so things in a way
that isn't compatible with some other functionality. A low-level port might mean a lot of re-inventing
the wheel and having to use functionality that is distor-specific but would have been hidden / abstracted
by the platform GTK 4.

So due diligence on both options, and not expecting completely smooth sailing for either.
Ease of implementation and maintenance and portability are strong arguments, but so are flexibility and
internal consistency in behaviour and behavioural compatibility with the existing X11 implementation.
So if doing it "right" is more work, we'll have to absorb that, and we want a stable base so we aren't
in a perpetual maintenance game.

Mario observed that we need to not get ahead of ourselves because we still have the list
of technical challenges that need to be resolved that apply to X compat mode and to a native port too.

We'll work on those but whilst they work through the eco-system over 2 years (?) we could be
doing something useful ..

One element of the proposal not directly related to wayland is we could merge EDT and Toolkit thread
since we no longer need these separate for applets. Possibly, but we kick off a new thread for modal
dialogs too .. could this be special cased ? Might depend on whether you can limit what happens
in such a case but I'm not sure you can.

We do have these documented on the wiki but it needs another pass to have a consitent format
- what the problem is
- what JDK can do
- what we need from the wayland ecosystem
- how much of a blocker it is
- can we "tweak" the spec - at the cost of making backports harder

We should look at this soon so we can get started on spec tweaks we think may be unavoidable but acceptable ..

Eg What's acceptable for screenshots ? We can't deprecate but can we make something optional ?

Alex Z: Is the SplashScreen API supportable ?

Apparently splash screens display in top lefe (the window positioning problem)

A splash screen (output only) surface role could probably be proposed upstream.
it'd just need a fallback implementation for compositors that doesn't support it, e.g. a dialog

Maybe similar to this PiP request : https://gitlab.freedesktop.org/wayland/wayland-protocols/-/merge_requests/132
For more info on roles: https://wayland-book.com/surfaces/roles.html
Add input to this existing issue : https://gitlab.freedesktop.org/wayland/wayland-protocols/-/issues/67

Accessibility ?

Is there a library ? Like ATK ?
There are existing DBUS based A11Y APIs ..
For the accessibility protocol Jonas is talking about: https://www.freedesktop.org/wiki/Accessibility/AT-SPI2/
For a quick overview of what Emmanuele did for accessibility in GTK4: https://blog.gtk.org/2020/10/21/accessibility-in-gtk-4/

Online Zoom Meeting 9am PST 16th Dec 2021

Attendees : Olivier, Phil, Sergey B., Kevin, Aleksandr, Dalibor, Victor, Niels De Graaf, Alexey Ushakov

Previously discussed upstream DnD bug is fixed upstream - maybe backported to gnome 41 branch ?

On Ubuntu 21.04 using JDK in Xwayland compat. mode DnD from one java app to another (eg) JEdit
gets dropped on native terminal window instead.
DnD is handled by the wayland compositor acting as proxy - Olivier has seen similar issues if you
have wayland native under X11 then it hits the window underneath this should be fixed if you try
Fedora 35 - Ubuntu 21.04 slightly older and presumably doesn't have this fix.

Sergey asked Alexsandr if his DnD testing was using the Robot API or by hand and whether
it is even theoretically possible if wayland doesn't report global screen coords for a window so you
can't move the mouse to a screen position of another windows.
i.e can't move mouse from window 1 -> windows 2 because don't know where window 2 is ...

In other words we have wiggle room in the APIs - and code in the implementation - such that SetBounds calls for a top-level
might be ignored or adjusted, but we rely on correct answers from GetBounds
Olivier : this should actually be OK because in compat mode the apps are talking to an X.org server albeit one acting as a bridge (ie Xwayland)

Some similar discussion where you just want to move the focus from Button A in WIndow 1, to Button B in Window 2
and you need to know the position
In wayland you could set the position by saying loc (x1,y1) in WIndow 2 even if you don't know the real position of Window 2
Not clear (to me) what this would look like for DnD as you want to move the mouse and see it cross the screen outside window boundaries ..
(some of the above needs verifying - outcome of the discussion was not 100% clear to me)
Screencast API is used for automated testing by wayland devs

Would we need to provide new APIs ? And respecify so that existing APIs are "optional".
Could get confusing because other platforms will not have the same issue or model.
Assumes that xwayland compat mode would not need this respecifying to give us backport options.
All too early to know.

Noted that GTK4 was re-specified in a similar way so that things that don't work on wayland are no longer part of the API

How is multi-mon handled in this scenario where you can't position/get position ? Full screen too ?
Full screen is a special case, and you may be able to specify the monitor for a window but that's about it.

Sergey : How does it work for custom decoration on pure wayland
Olivier: Some history here expected client-side decorations was always the answer - so custom decorations would naturally fall from that.
But KDE folks not happy - wanted server side decorations for consistency on the desktop so there is an optional protocol for a wayland
compositor but the client needs to be able to fall back to client-side in case that protocol is not supported on a compositor.

Is there a deadline to ship xwayland compat JDK ?
None yet - still working through these issues - a deadline might be when (say) RHEL shipped without x.org .. which would be a real problem if
we didn't have all the answers ready.

Online Zoom Meeting 9am PST 2nd Dec 2021

Some progress on issues previously seen - some upstream bugs filed and/or fixed.

As reported on the mailing list the DnD issue previously discussed has been fixed upstream

Mapping a single large window : 2 bugs : 1 in wayland 1 in xwayland

Problems with creating a large number of windows is because wayland has fixed buffer sizes for the number of mapped windows

Four, 4K buffers for file descriptors, other requests .. they can fill up
Variable buffer sizes is tricky and a proposed patch is under discussion but not yet accepted.

Some of these problems do not affect running native directly on the hardware with Glimmer and DRM available
but only affect running in a VM where shared memory is used instead.

Online Zoom Meeting 9am PST 18th Nov 2021

We went through some of the issues raised by individuals on the call

Wayland does not provide access to the absolute coordinates of a window on a desktop
or allow specifying it. Whereas X window managers don't make guarantees about positioning but
generally do honour a request and do allow querying. We may have some work to do in making
things work as well as possible with the different approach of wayland
There's (some) discussion here : https://gitlab.freedesktop.org/wayland/wayland-protocols/-/issues/72

There was discussion of ideas such as creating an invisible window that spanned the entire screen
and creating other windows as children of that but this may be fighting another battle where child
windows are assumed to be transient windows.

Focus issues .. and bringing a window to the front or sending it to the back ?
Maybe a way to bring a window to the front with focus but not sending to the back
XDG activation protocol can ask for attention and that usually means being raised to front
https://gitlab.freedesktop.org/wayland/wayland-protocols/-/blob/main/staging/xdg-activation/xdg-activation-v1.xml

Drag and Drop bugs - Alexander having some trouble on Fedora 35

Solutions to some of the challenges are still evolving and different distros may provide different solutions.
There was a discussion and agreement about the importance of implementing to a level that abstracts away
platform-specific code. Coding to something that depends on a particular library that a platform
may not choose to deliver makes our task harder and nearly impossible to deliver one OpenJDK binary
that could run on multiple distros.

All (or at least most) of this is still applicable to both the xwayland case and the wayland case.
Focus for now continues to be on the former. Too early to say if GTK4 or lower-level approach will
ultimately win out for OpenJDK needs.
It was observed that GTK and its print dialog drive you to xprint, which may not be what we want.
Not yet looked into what printing will look like in a native wayland port.

Online Zoom Meeting 9am PDT 16th Sept 2021

Attending : Wakefield committers from Oracle, Redhat, JetBrains and Amazon as well as Wayland developers from Redhat who were invited by Mario.

The intent of this meeting was to have a productive dialog between the OpenJDK developers and Wayland community developers to come to a better shared understanding of the technical challenges and current state of Wayland and identify avenues of investigation.

Some opening remarks were about the Wakefield infrastructure, including this wiki, the Wakefield mailing list and the Wakefield repo.

Now they are up and running we expect to use the mailing list for most communications and document progress etc on this wiki page.

Branches should be created in the Wakefield repo for all work / experiments etc. Don't use the master branch.

In other words, we should use the project infrastructure for everything and publish contributions and ideas there.

The major topic of conversation was around the options for JDK for capturing screenshots and synthesising input events - both for running in X11 compatibility mode or as a native Wayland client

Screen capture is done today by calling a GTK API which is no longer exposed in GTK 4 because core wayland doesn't support it.

We need an alternative, (JDK has a pure X11 fall back based on xwd but we didn't touch on that - it has the same issue)

A few things were thrown out that might merit investigation

https://flatpak.github.io/xdg-desktop-portal/portal-docs.html#gdbus-org.freedesktop.portal.Screenshot

https://docs.flatpak.org/en/latest/portals-gtk.html

https://github.com/flatpak/xdg-desktop-portal#using-portals

https://github.com/flatpak/libportal

The libportal abstraction / helper layer is probably what we'd want.

It is unclear (to me the note taker) if this is readily available on distros alongside wayland - we can't ship it ourselves and it ilikely needs more than a client-side library anyway

JDK for X11 today needs the functionality of the XTEST X11 extension protocol to support the input event side of things.

The wayland desktop answer for this is probably going to be https://gitlab.freedesktop.org/libinput/libei but it is still in development.

It is very unlikely that distros will be ready to ship all the pieces we need in the next 12 months. So the "short term" goal may actually need to wait for somewhat longer than that.

What is the API the clients used for rendering to the wayland client off-screen surface ? GTK defaults to OpenGL (EGL) and falls back to shared memory (ie some software rendering and a copy)

There is also an experimental Vulkan backend.

We use Xrender as the default Linux rendering pipeline today. Maybe we could use OpenGL for the X.org desktop too ? Yes, it is possible that we end up making improvements to the XGL pipeline as a consqeunece of adding EGL support for Wayland so that we could make OpenGL the default there (X.org) too. But it would be a side-benefit not the main goal.

What are people using for window decoration since the wayland model is that the client does the decoration ?I didn't catch all the options here seems there are a few

There was also a question about hidpi causing fuzziness of X11 apps especially for fractional scaling.

The wayland devs confirmed it is a known issue but difficult to solve without making X11 apps far too small to be legible.

The wayland devs input seems to be essential to us navigating a very complex and evolving landscape.

We agreed we need to go off and study some of these APIs and come back with follow up questions when ready.