Project Wakefield: A New Wayland Desktop for Java On Linux

[BOF2627 – JavaOne 2022]

Phil Race, Alexander Zvegintsev

Java Desktop UI Client Team,

Java Platform Development Group, Oracle

Project Overview : Background : X11

- **The Linux X11 based desktop technology is getting out-dated**
  - X11 is the core display server technology for Linux and most other Unix-like OSes
  - Many design decisions of X11 are rooted in the 1980's
  - Client application connects to the Xserver process.
  - The client sends requests and receives events.
  - Rendering mostly happens in the Xserver process.
  - Lots of extensions tagged on to help keep it usable for a modern desktop stack.

 Copyright © 2022, Oracle and/or its affiliates

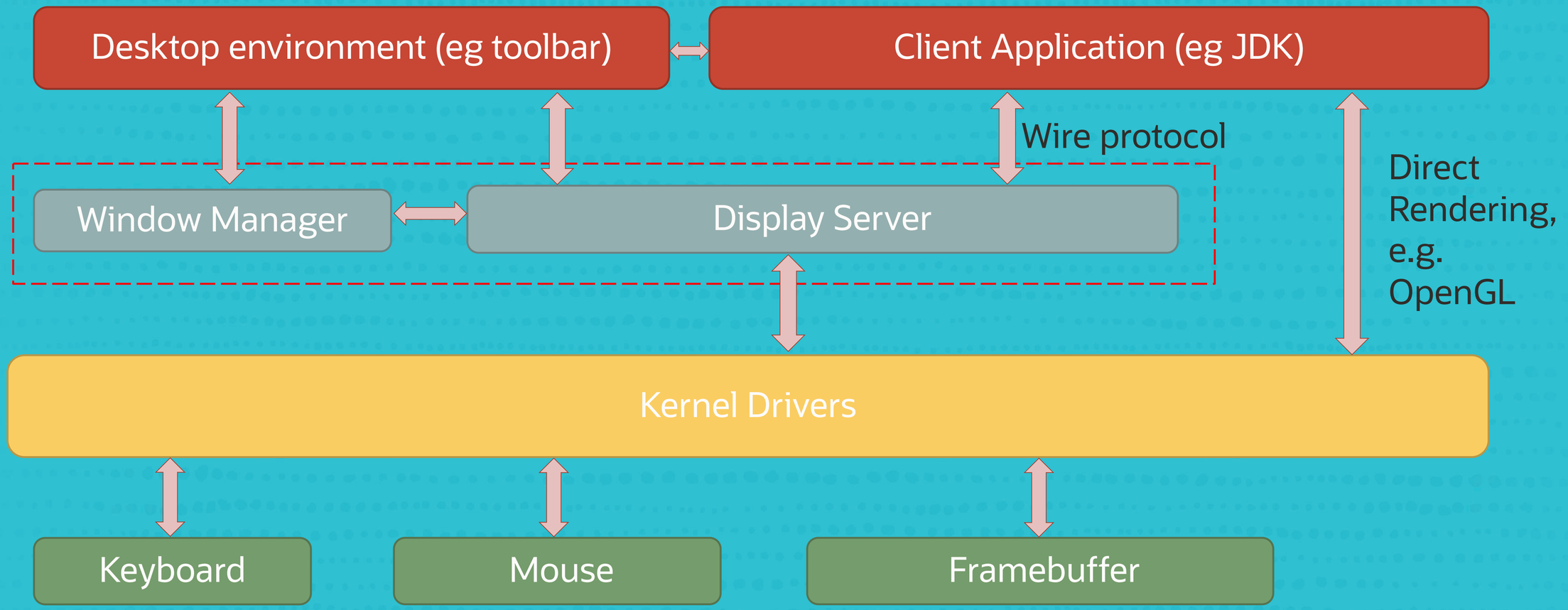
Project Overview : Background : Enter Wayland

- The Linux community has been working for a number of years to replace X11 on Linux
  - **Wayland** is the solution that has gained most traction and is now being delivered by multiple distros

- Wayland replaces X11 but not the entire desktop stack
  - Rendering has increasingly moved from Server to client and just push pixels to the Xserver
  - Wayland reflects this as it expect the client to do all rendering so is agnostic to how that is done.
  - Wayland focuses on surface creation, management, composition and event delivery
  - Window composition built-in rather than needing roundabout solutions for X11 needing a Window Manager.
  - GTK-based applications may use very little X functionality directly even on X11 backend
  - So GTK can (via its subsidiary components) use X11 or Wayland at the backend
  - Applications using GTK just need to migrate to the newer GTK 4 and they are wayland applications.
  - End result is a Wayland based desktop can look exactly like an X11 based desktop
  - But with an architecture more suited for modern client applications – less overhead, no unused baggage.

 Copyright © 2022, Oracle and/or its affiliates

Typical Desktop Architecture on Linux



Copyright © 2022, Oracle and/or its affiliates

X11 vs Wayland

- X11 and Wayland both define communications protocols for display servers

- X.org and Weston are implementations of display servers for each of these protocols
  They are not a full desktop stack - just a display server.
  This handles (at a minimum)
  - Display initialisation and management of client connections
  - Requests from a client eg for surface creation
  - Input events
  - Low level libraries are provided for client and server-side to wrap the protocol

```
wl_display_connect <-> XOpenDisplay
wl_display_disconnect <-> XCloseDisplay

X11 : connect to port 6000 or socket /tmp/.X11-unix/X0
wayland - Unix Domain socket "wayland-0"
```

 Copyright © 2022, Oracle and/or its affiliates

X11 vs Wayland (continued)

- X11 does server-side rendering
  - XDrawLine(display, window, x0, y0, x1, y1)
  - XDrawString(display, window, x, y, “hello”)
- So also provides graphics context support
  - XSetFont(display, gc, font)
- And since it has the fonts, it needs to do the text measurement for you too
  - XQueryTextExtents(display, ….)
- Wayland just gives you a surface and API to copy from a buffer to the surface
- Fonts/text obtained, measured, laid out and drawn with freetype/pango/harfbuzz
- Geometrics primitives drawn by whatever graphics library used eg Cairo

```
wl_buffer = wl_shm_pool_create_buffer(..) ; // draw to this memory buffer
wl_surface_attach(surface, buffer, 0); // attach to wayland compositor surface
wl_surface_commit(surface); // move contents of buffer to screen (front buffer)
```

 Copyright © 2022, Oracle and/or its affiliates

Wayland usage today [Wayland is on its way]

- Wayland is already the default on OL8, RHEL8, Ubuntu 22.04 LTS, etc ..
  - This means X11 clients like JDK run in compatibility mode - connect to "XWayland" which is an Xserver running in conjunction with the Wayland desktop compositor. It is a wayland client as well as an Xserver and the Wayland compositor is also a client of Xwayland.
  - This connection is completely automatic – Xwayland advertises as a standard Xserver
  - Not all X11 APIs and extensions work with wayland (or xwayland).
    - **Ecosystem still working on answers to these**
    - **Specific examples coming up on later slides**
  - User login screen provides a (somewhat hidden) option to revert back to pure X.org

 Copyright © 2022, Oracle and/or its affiliates

OpenJDK Project Wakefield

Support for the Wayland based desktop in OpenJDK

- JDK's AWT for Linux is X11-based so must run in XWayland compatibility mode
  - How ever some things do not work well (or at all) due to limitations of XWayland or Wayland design decisions
  - Native Wayland support is a lot of work – replace all of AWT+2D code for X11
- Project Wakefield initiated with Redhat and JetBrains as key contributors/partners
- Project goals
  - Support JDK's X11-based AWT/2D on XWayland
  - Identify problems and work with Linux community / project partners to find solutions to limitations - slides on these coming up.
  - Natively support Wayland : A lot of work – replace all of AWT/2D for X11 – Wayland limitations apply here too (note: *mostly* the same problems)
  - X11 mode and Wayland native need to be supported together for many years

 Copyright © 2022, Oracle and/or its affiliates

Issues which must be addressed

Pure Wayland

- Screenshots
- mouse/keyboard events generation
- Drawing window decorations on our own
- No top-level window position control
- Splash screen is not centered

X11 compatibility

- Screenshots
- mouse/keyboard events generation
- Popup dismiss
- HiDPI looks blurry

Copyright © 2022, Oracle and/or its affiliates

Wayland: no top-level window position control

- Wayland does not allow to set a position for a top level window
- Window manager decides where to place window
- Our documentation is ready for that:

Calls to setLocation, setSize, and setBounds are requests (not directives) which are forwarded to the window management system. Every effort will be made to honor such requests. However, in some cases the window management system may ignore such requests, or modify the requested geometry in order to place and size the Window in a way that more closely matches the desktop settings.

- Works fine with X11

 Copyright © 2022, Oracle and/or its affiliates

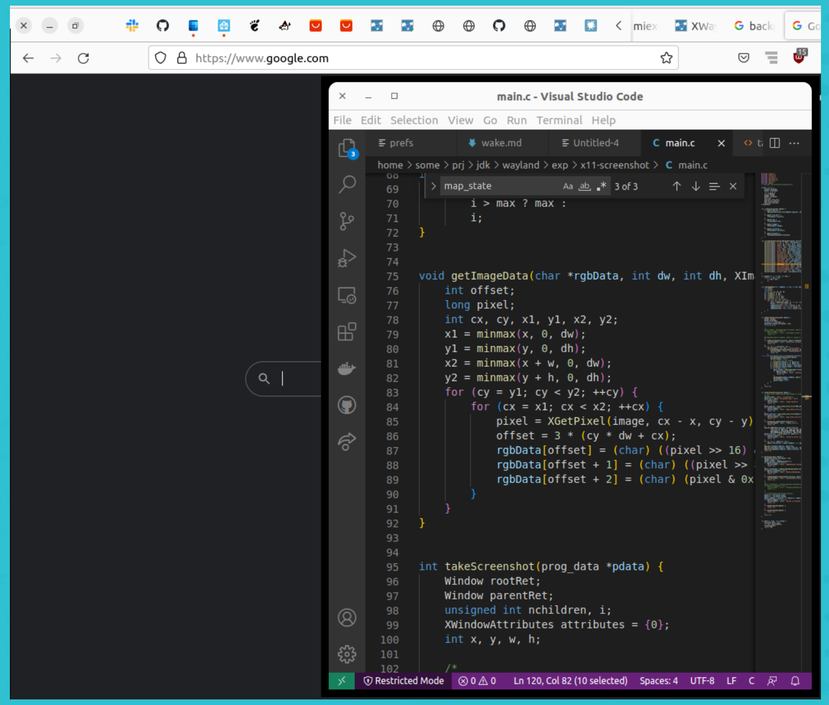
Wayland: splash screen location

- Splash screen usually appears in the top left corner
- This will need a Wayland protocol extension to add a role for an output-only, centered surface
- There is an open issue for this
- Works fine with X11

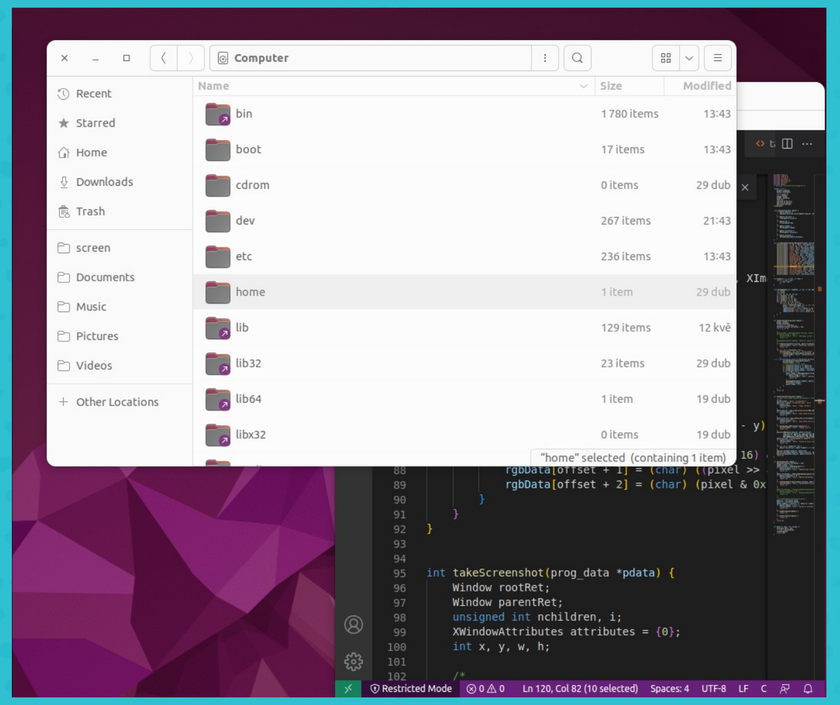
 Copyright © 2022, Oracle and/or its affiliates

Wayland Screen Capture using X11 APIs

What can we obtain using X11 API



What is actually on screen

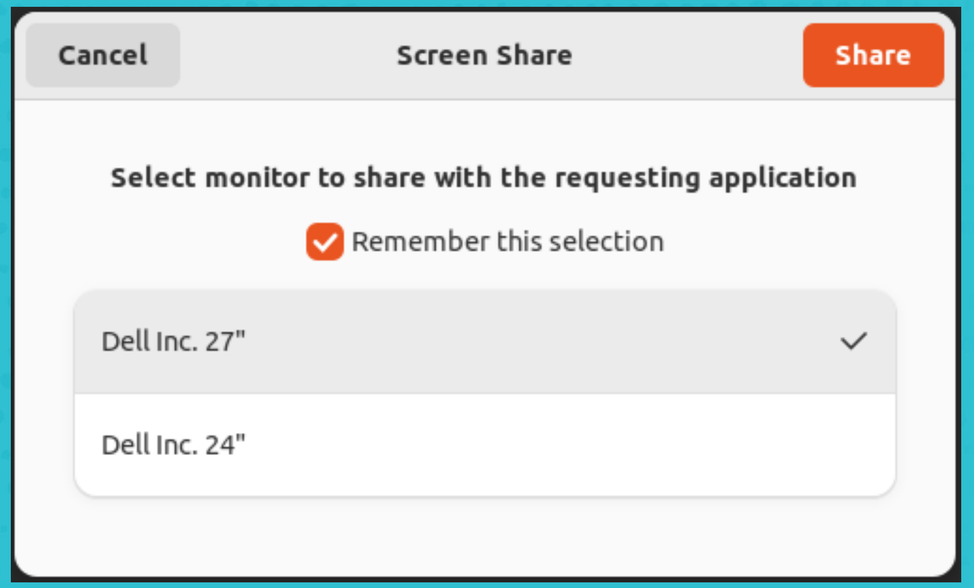


Looking at xdg-desktop-portal DBUS API options instead (next slides)

Copyright © 2022, Oracle and/or its affiliates

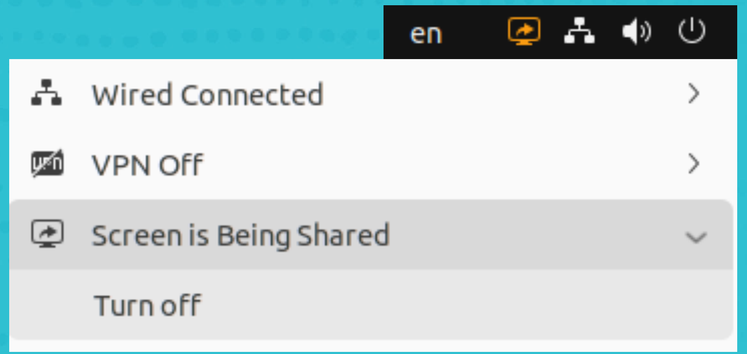
Screen Capture Option 1: org.freedesktop.portal.ScreenCast

- No intermediate file
- We can keep user permission via a restore token



Drawbacks and caveats:

- OL9, RHEL9, SLES15 don’t yet have restore_token functionality
  Ubuntu 22.04 is the only LTS version that supports it
- User can deny permission request
- “Screen is being shared” icon
- Need to reacquire user permission if the configured displays change



Copyright © 2022, Oracle and/or its affiliates

10/20/22

Screen Capture Option 2: org.freedesktop.portal.Screenshot

- saved to a file -> slow
- no way to control its path.
- a visual screen flash that cannot be turned off
- Needs Gnome 43+ to keep user permission to take screenshots without confirmation

We may need to use both options – Prefer #1 (ScreenCast), use #2 (Screenshot) as a fallback

Copyright © 2022, Oracle and/or its affiliates

Mouse/Keyboard Event Generation

Wayland:

- Generating new virtual input device with uinput.
  Root privileges required.
- org.freedesktop.portal.RemoteDesktop DBUS API
  Confirmation for every session
- Using libinput/libEI library
  release date is unknown

X11 compatibility:

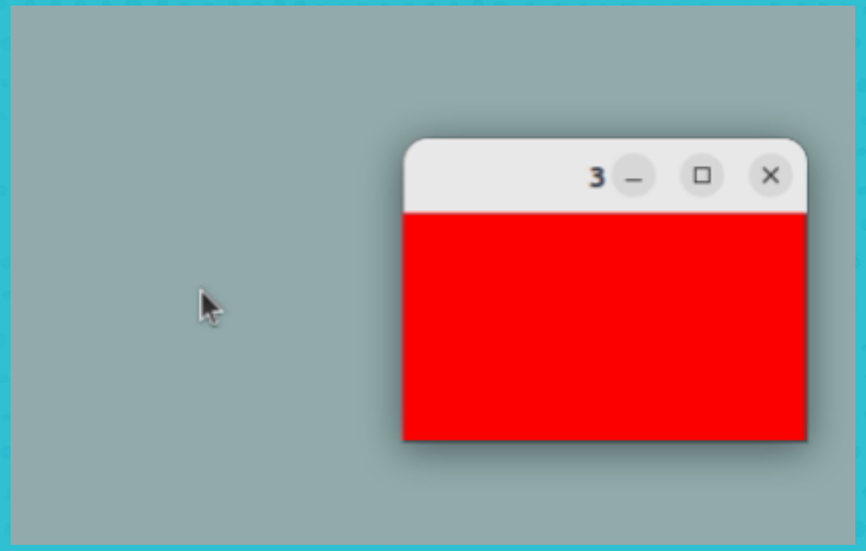
- XTest works
- There are some cases when it fails

 Copyright © 2022, Oracle and/or its affiliates

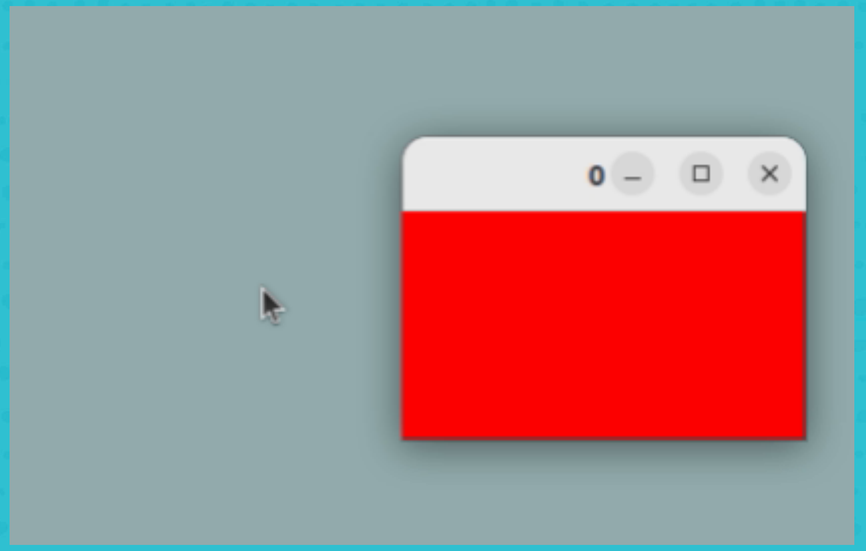
X11: mouse/keyboard events generation

- Java.awt.Robot#mouseMove does not visually move mouse cursor
- There is a JCK test for this
- Mouse events inside XWayland server delivered to correct location

X11



XWayland



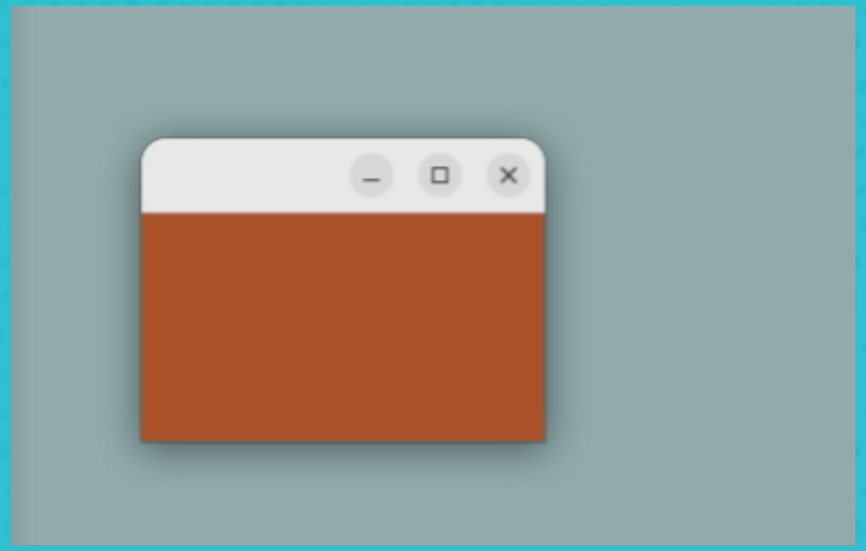
Animated content - view in slideshow mode

Copyright © 2022, Oracle and/or its affiliates

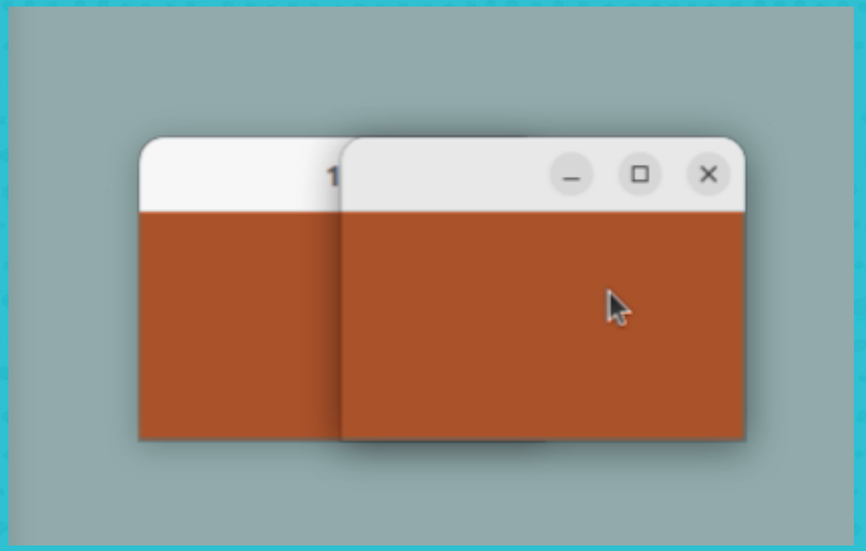
X11: mouse/keyboard events generation

- Emulated mouse click does not bring window to front
- Workaround: call toFront() on window

X11



XWayland



Animated content - view in slideshow mode

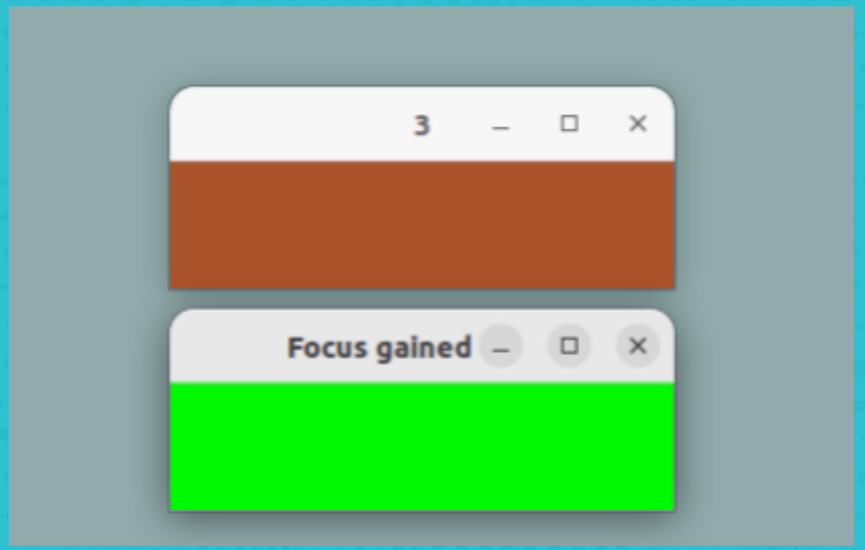
Copyright © 2022, Oracle and/or its affiliates

X11: mouse/keyboard events generation

- Unable to focus window by clicking on its title
- Workaround: click inside window

X11

XWayland



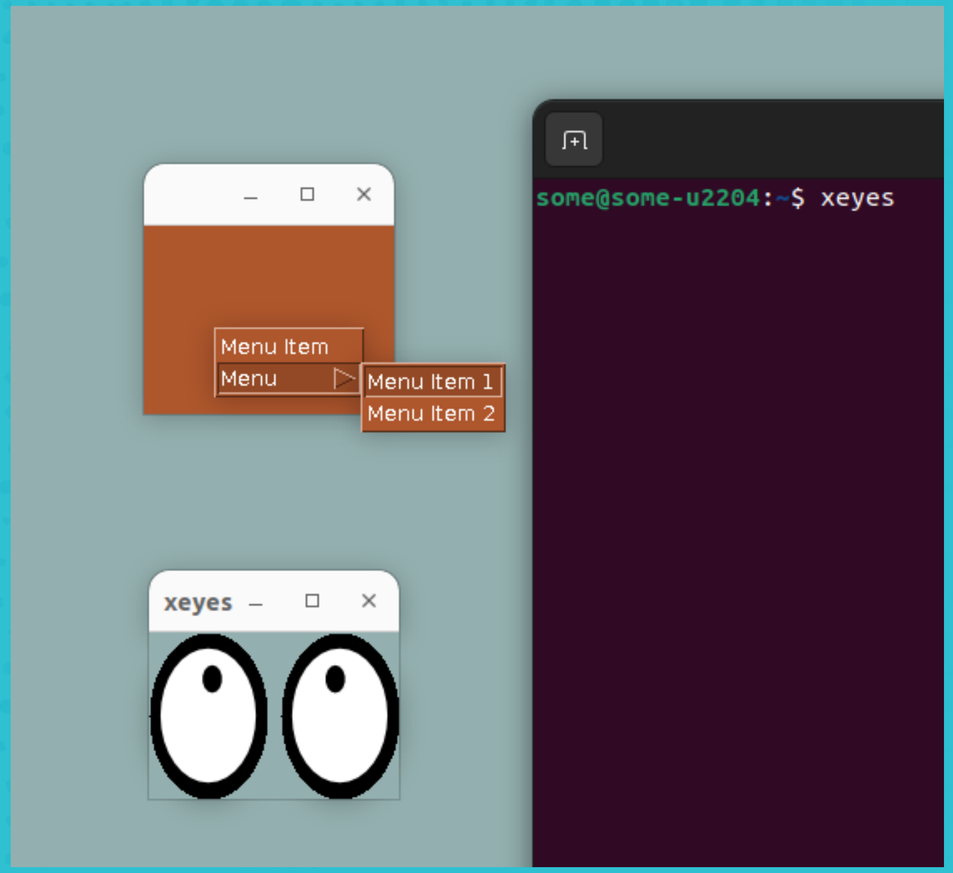
Animated content - view in slideshow mode

Copyright © 2022, Oracle and/or its affiliates

X11: Popup dismiss

Unable to dismiss popup by clicking outside of XWayland server

- X11 mouse input grab does not work outside of XWayland server
- We can dismiss popup on window focus lost
- Drawback: can’t dismiss popup if clicked on its own window title



Copyright © 2022, Oracle and/or its affiliates

X11: HiDPI looks blurry with fractional scaling

- XWayland only issue
- Native Wayland applications support HiDPI

 Copyright © 2022, Oracle and/or its affiliates

Plans

- Xwayland/X11 compatibility mode
  - Need to be able to support this
  - Must have a solution for screen capture – at least.
  - Some issues can perhaps be solved by documentation changes.
  - Backports are not intended

- Native Wayland
  - Prototyping of libwayland-client and perhaps GTK4 based alternative
  - Track work of Wayland ecosystem on blockers

 Copyright © 2022, Oracle and/or its affiliates