WSLG

X11 and Wayland Applications in WSL
What is WSL?

- **Windows Subsystem for Linux**
  - Infrastructure to run Linux applications inside of Windows
- For more context on WSL see our other XDC talk
  - *WSL – Graphics Architecture*
WSL Today – Terminal Experience
Most requested WSL features

• Access to the GPU from within WSL
  • Mostly for compute
  • Most requested is access to NVIDIA CUDA API
  • This is the subject of our other XDC talk
    • *WSL – Graphics Architecture*

• Ability to run GUI applications
  • Going beyond a terminal only experience and the ability to run X11 and Wayland applications
  • *Subject of this talk*
But wait…

• You can run X11 applications in WSL 2 today!
• You can run an X Server in WSL 2
  • Run an X Client on Windows and connect to WSL 2
  • Lots of instructions online and great videos on YouTube on how to set this up
• It works… but experience is not great
  • Fundamentally a desktop in a window experience
  • Performance is not great
  • No Wayland app support
  • Doesn’t work out of the box
Linux GUI App Integration (WSLG)
What’s our approach?

- Building on Weston
  - Reference/sample compositor for Wayland
- Extending Weston existing RDP backend and teach it about remoting application
  - RAIL – Remote application integrated locally
  - VAIL – Virtualized application integrated locally
- Leveraging Pulse Audio for audio in & out
  - Bridging over to RDP
- Building with intent of contributing back
WSLG Overview

- **Windows Host**
  - RDP Client
    - (mstsc.exe)

- **Linux Guest**
  - Pulse Audio Server
    - RDP Sink
    - RDP Source
  - Xwayland
    - X Server
  - X11 App
  - Native Wayland App
  - Weston
    - X Window Manager
    - RAIL-Shell
    - libweston
    - backend-rdp
  - FreeRDP
    - RDP RAIL/VAIL Over HVSocket
Why based this on RDP RAIL/VAIL?

• Mature technology and host side client
  • Been remoting applications through RDP for years
  • Have mature business built on this technology (Azure WVD)
  • Host side client is same irrespective of guest type
• Built-in network transparency and performance scaling
  • Over the network -> RAIL -> Pixels copied over the transport
  • Over VM boundary -> VAIL -> Shared guest <-> host memory.
• Enable remoting of both X11 and Wayland applications
• Documented & Open Source implementation available
  • Continue to use FreeRDP with Weston
Built for remoting transparency

- RDP VAIL
- WSL
- RDP VAIL
- HyperV VM

- RDP RAIL
- Linux PC
- RDP RAIL
- App Remoting Services
RAIL & VAIL
Overview
Mirrored Desktop

• X11 applications have knowledge and can have direct control over their placement on the X desktop

• In RAIL mode, the guest and the host have the same desktop configuration:
  • Same number of monitors, at same location, resolution, orientation, etc...

• Initial desktop configuration sent by RDP Client upon connecting
  • New configuration information sent by RDP Client when desktop configuration changes
    • Monitor added/removed/repositioned/rotated/resolution change/...
On RDP Connection, RDP guest sends initial desktop configuration to the RDP server.
RAIL – Mirrored Desktop – Desktop Change

Wayland Clients

wl_output0
wl_output1

linux_guest

Event

wl_output.geometry
wl_output.mode

Windows Host

Monitor0
Monitor1

Monitor0

RDP client Send configuration change to RDP server.
RAIL – Proxying top level Visual

• RDP client on the host create a proxy visual for every top level visual created by the server in the guest
  • Proxy window is borderless. No non-client area. All pixels are owned by the RDP Server.
• Damage pixel in the guest/server are copied to the host
RAIL – Proxying top level Visual - Create

Linux Guest

Windows Host

wl_output0

wl_output1

wl_surface

Monitor0

Monitor1

window

RDP Server send new window creation to the RDP Client
RAIL – Proxying top level Visual - Update

RDP Server send pixels for the damage area of the visual to the RDP Client
RAIL – Proxying top level Visual - Destroy

Linux Guest

Monitor0

wl_output0

wl_surface

Monitor1

wl_output1

window

Windows Host

RDP Server send window destruction to RDP guest
RAIL – Proxying top level Visual – Move/Resize

Linux Guest

Windows Host

wl_output0

wl_surface

wl_output1

Monitor0

Monitor1

window

Raw input

RDP Client sends raw input to RDP Server

RDP Server send new window position and/or size and content
RAIL & Composition Beat

App

Shared memory buffer

Composition thread
Processing update @60hz (timer)

Pixels copied over the network transport

Received from network transport

Linux

Windows
VAIL + Software Rendering

**Linux**

- App
- Compositor
- Shared memory buffer
- Buffer shared with the host
- Commit notification processed without delay.
- No more 1 extra frame latency (extra ~100us introduced by transport)

**Windows**

- DX11
  - 2 copies
  - GPU copy to swapchain
- DX12
  - 1 copy
  - Need vGPU for zero copy

**DX11 & DX12**

- Create Staging around shared memory
- CPU copy to staging
- GPU copy to swapchain
- DCOMP Swapchain
VAIL + vGPU

- **Linux**
  - App
  - Compositor
  - Shared memory buffer
  - Buffer shared with the host
  - Commit notification processed without delay.
  - No more 1 extra frame latency (extra ~100us introduced by transport)

- **Windows**
  - DCOMP visual

**Diagram Notes**
- Zero copy
- Minimal latency overhead
Packaging and Servicing

wsl --update
System Distro

- WSLG ships side by side in a system distro
  - Compositor has a clean and isolated user space and project socket back into user distro
    - Sort of like a container, isolated mount and PID space.
  - Each distro has a unique instance of the system distro
  - Each system distro has RDP connection back to the host
Servicing

• Shipped separately from Windows
• Always kept up to date
  • Let Window Update handle this for you if you want
  • Or take control of update
    • `wsl --update`
• System distro is read-only, replaced on update
  • Same model & technology used to update the WSL 2 Kernel
  • Simple rollback mechanism in case something goes wrong
• Custom system distro
  • User can use `wsl config` options to replace system distro with one of their own
  • Everything needed to build a system distro open source
  • Updates only changes the default system distro, don’t impact user custom
Works out of the box

• Automatically setup on WSL install or update
  • User distro pre-configured during WSL custom init
  • Install GUI app & launch

• Linux app appears in the Windows Start Menu
  • Any applications with a .desktop file

• Don’t want it or prefer to use an X-Server & Client?
  • Can disable WSLG in wsl config
  • Can keep today’s behavior

• Preview coming soon to Windows Insider