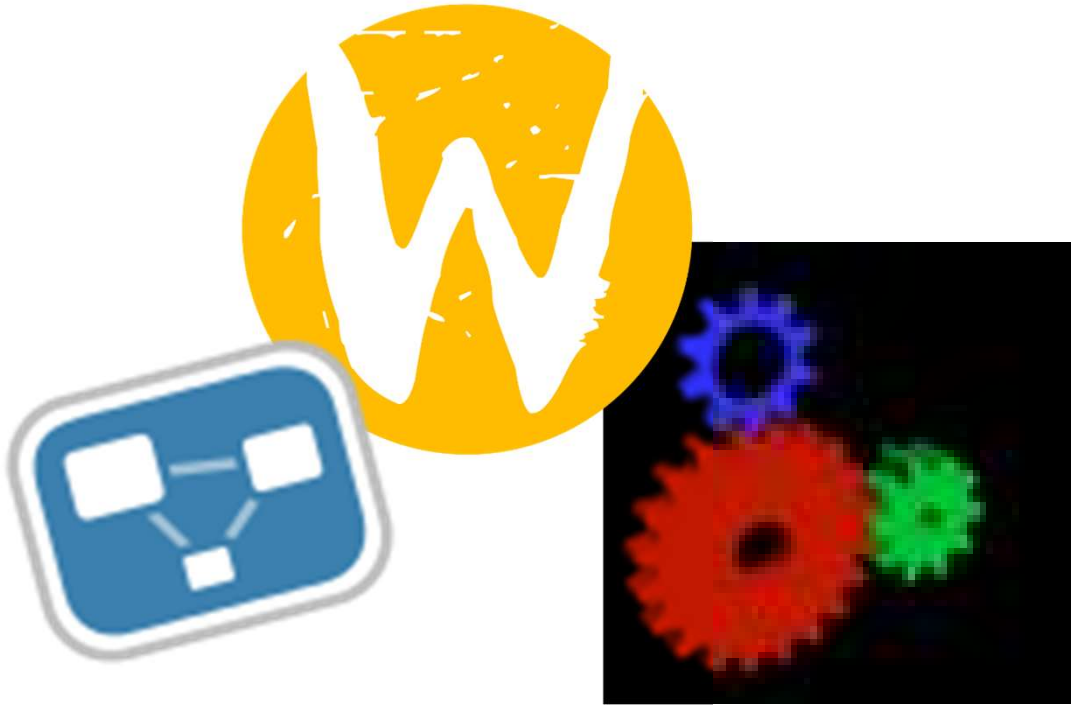
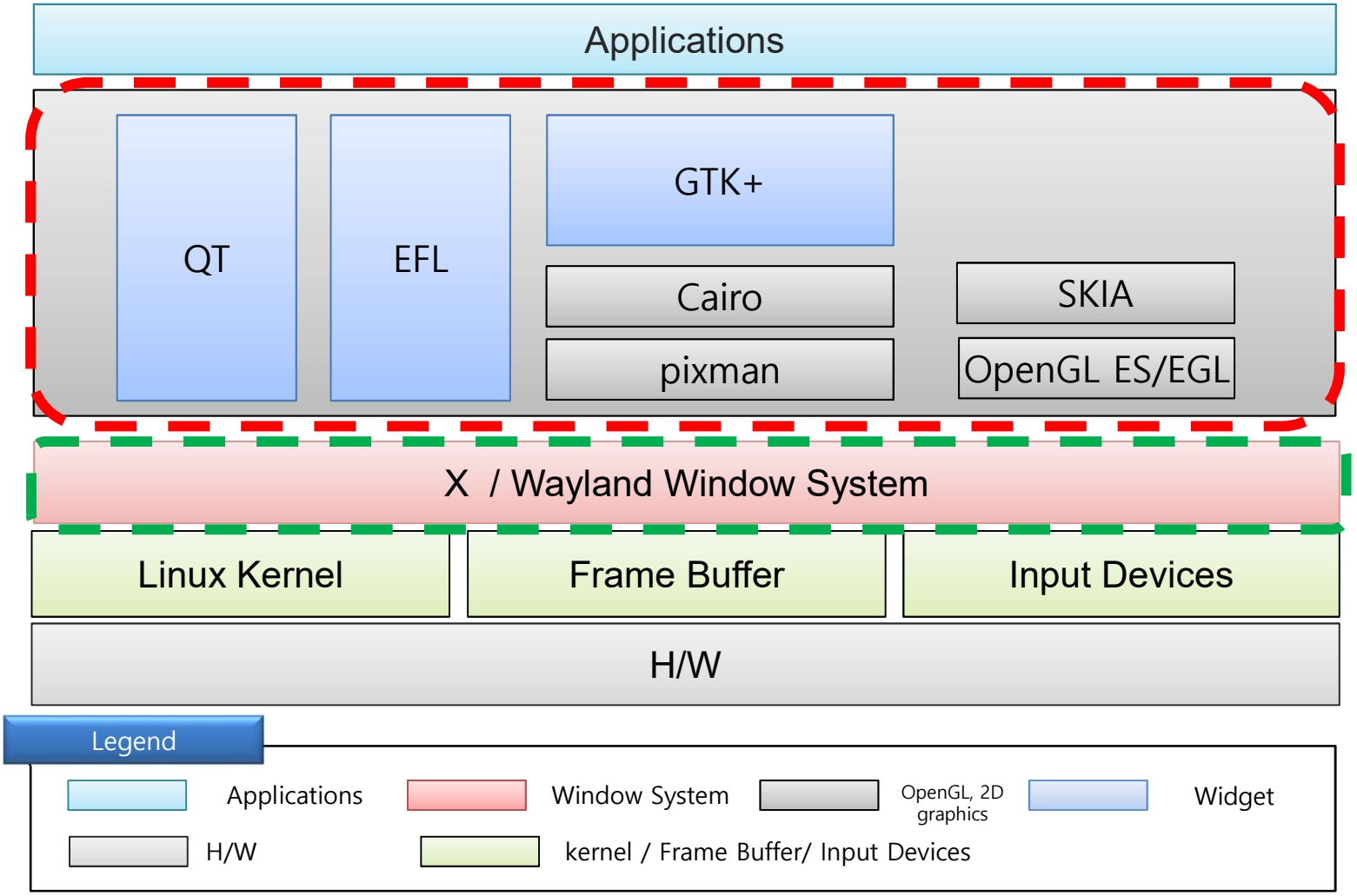
Linux Graphics Stack 2017+

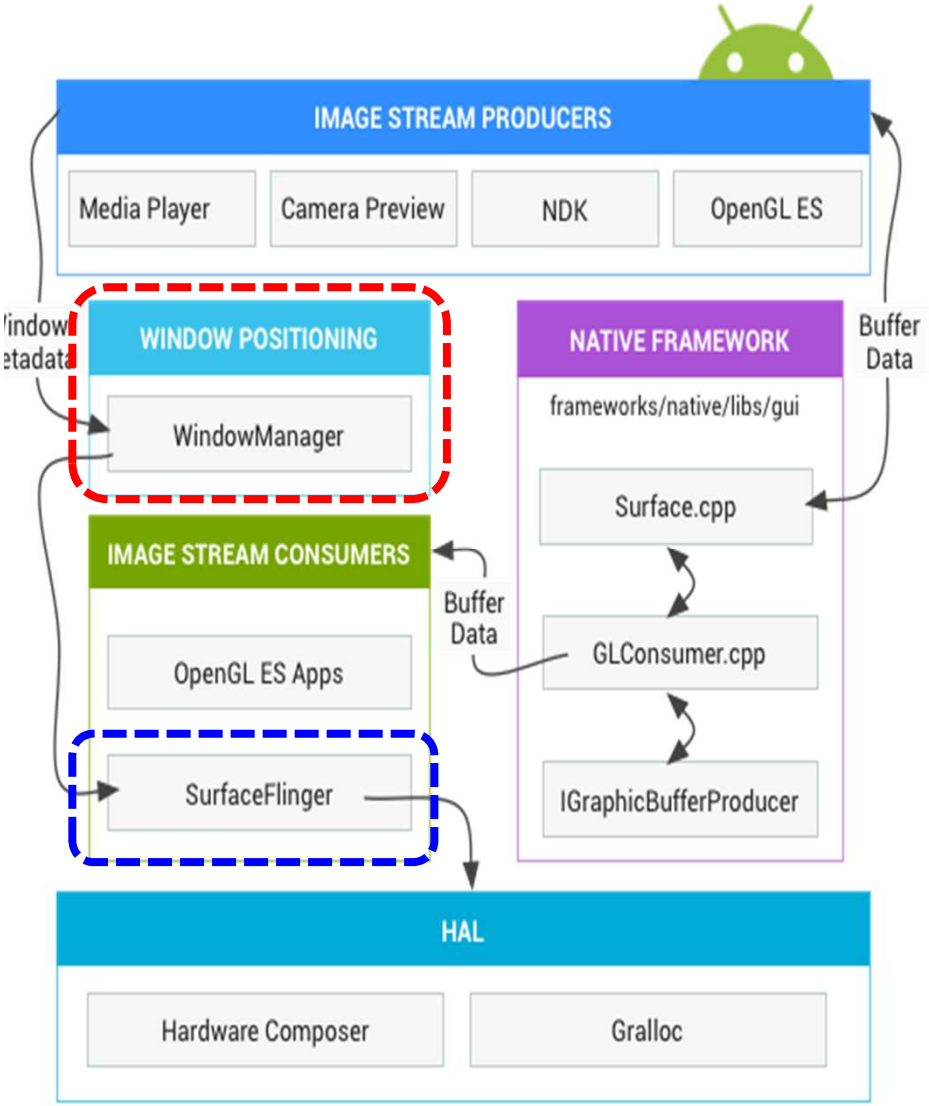
Mun Gwan-gyeong

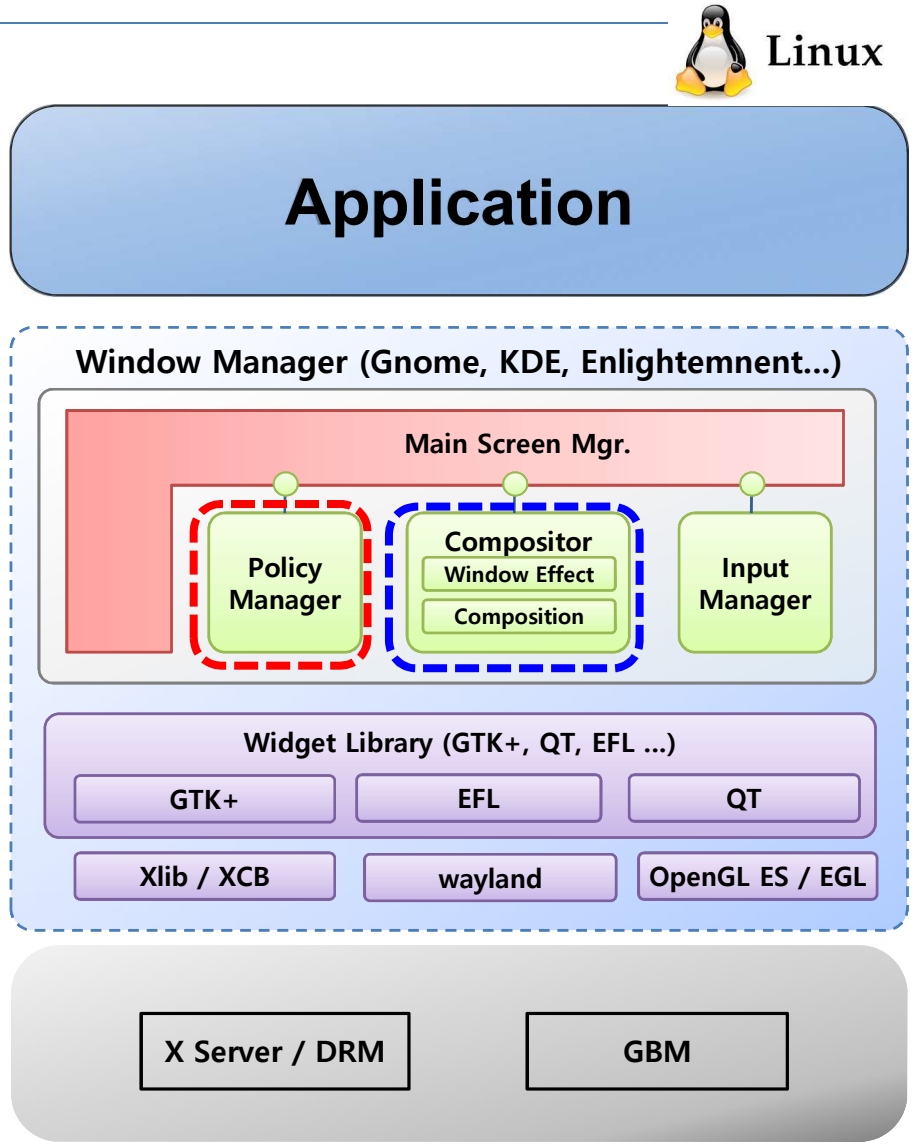


Window System / Widget



Android? Linux?





출처: https://source.android.com/devices/graphics/index.html

Kernel Side

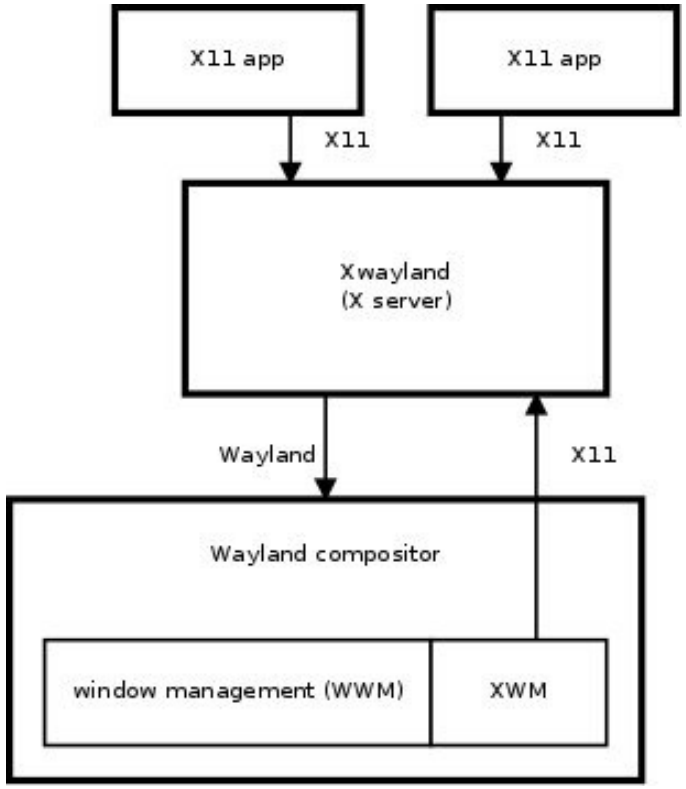
- FBDEV → DRM
- FBDEV (Linux framebuffer)
  - The Linux framebuffer (fbdev) is a graphic hardware-independent abstraction layer to show graphics on a computer monitor, typically on the console
- Direct Rendering Manager (DRM)
  - subsystem of the Linux kernel
  - interfaces with the GPUs of modern video cards.
  - DRM exposes an API that user-space programs can use to send commands and data to the GPU, and to perform operations such as configuring the mode setting of the display

Kernel Side (cont.)

	DRM	FB
Dynamic Allocation	YES	NO
Import	dmabuf	NO
Export	dmabuf, mmap	mmap
Atomicity	YES (atomic mode setting)	NO
Sync (needed for Vulkan)	DRM Sync Object	NO

Window System (Display Server) Side

- X Server → Wayland Server (+ Xwayland)
  - X Server + Window Manager
  - Wayland Display Server
    - No tearing
    - lightweight protocol (low communication overhead)
    - Direct rendering
    - easy to extend protocol



Display Server / Window Manager

- Display Server
  - Control Input / Output

- Window Manager
  - Window Policy
    - Window Size, Position, Stack, Visibility, Rotation
  - Window Compositing
    - Window Effect, HW Compositor
  - Etc
    - ANR ...
- X org
  - Display Server : X Server
  - Window Manager: Gnome, KDE, Enlightenment, works as X Client.

- Wayland
  - Display Server + Window Manager = Enlightenment, Gnome, KDE

Why Wayland? (Limitation of X)

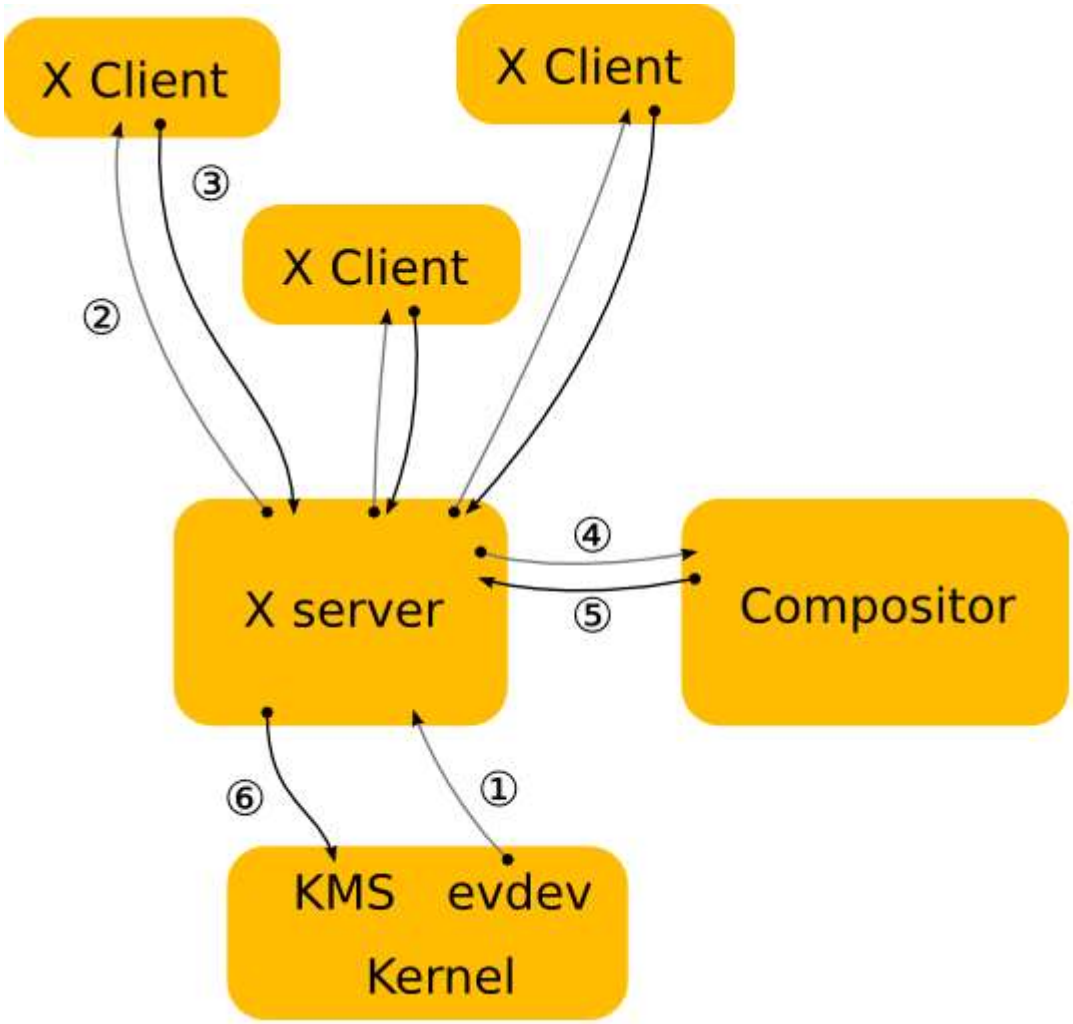
- X is impossible to secure by design
  - The X11 security model assumes that attackers are rejected at connection time.
  - There is no isolation between different apps.
  - Attacker can do below thing with access to a X11 server
    - Getting screen dumps
    - Reading keystrokes
    - Injecting keystrokes as if they were typed by the user

- Designed more than about 30 years ago
  - 1984, ,MIT Athena Widget Project
  - X11R7.7
  - Not just implements Drawing parts (Events Deliverer)
  - Many extension (XRender, XComposite, XFixes … more than 28)

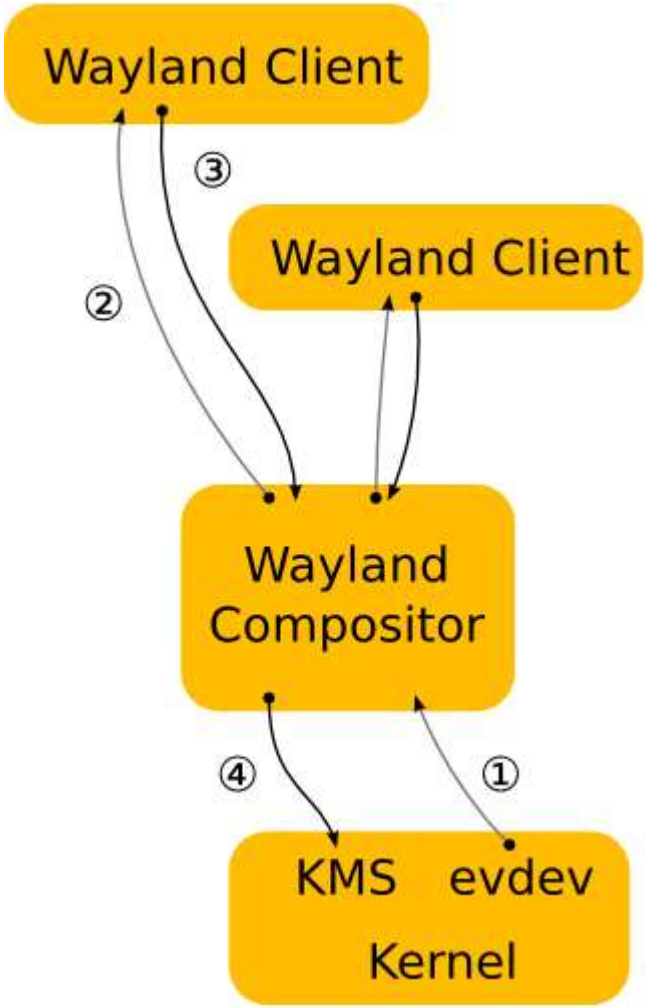
- Trend-shifting from desktop to mobile devices

- Needed to be competitive

Architecture (X Display Server)



Architecture (Wayland Display Server)



Widgets (GTK+, QT, EFL)

- Moving to client side rendering from server side
  - Use dri
- GPU rendering API (GL → Vulkan)
  - GTK 4.0 : GNOME's GTK Vulkan Renderer Faster Than OpenGL, Now Working On Windows (https://www.phoronix.com/scan.php?page=news_item&px=GTK-Vulkan-Speed-Windows)
  - QT 5.10: Vulkan Support in Qt 5.10 (https://www.phoronix.com/scan.php?page=news_item&px=Qt-5.10-Vulkan-Support
- Vector Rendering API
  - Cairo, Skia (Supports GL Renderer)
- Pixman

GPU Driver (Mesa)

- Mesa
  - Mesa3D or Mesa 3D Graphics Library
  - An **open-source device driver** and **software implementation** of the **OpenGL, Vulkan** and other specifications.
  - Supports major 2 vendor (**Intel / AMD**), 1 community (nvidia)driver.
  - And others (braodcom vc4/5, vivante, qualcomm adreno) drivers

Mesa 3D

- **Implementations of rendering / Window System APIs**
  - **OpenGL** (~4.6)
  - **OpenGLES** (~3.2)
  - **Vulkan** (1.0, anv, radv)
  - **Direct3D** (9.0c, gallium)
  - **OpenCL** (gallium)
  - **GLX / EGL** (~1.5)
  - **X11, Wayland, DRM**

OpenGL to Vulkan

- Why a new 3D API?
- OpenGL is old - Released in 1992
- Mesa Started in August 1993
- 24 years of development, GL has done well.

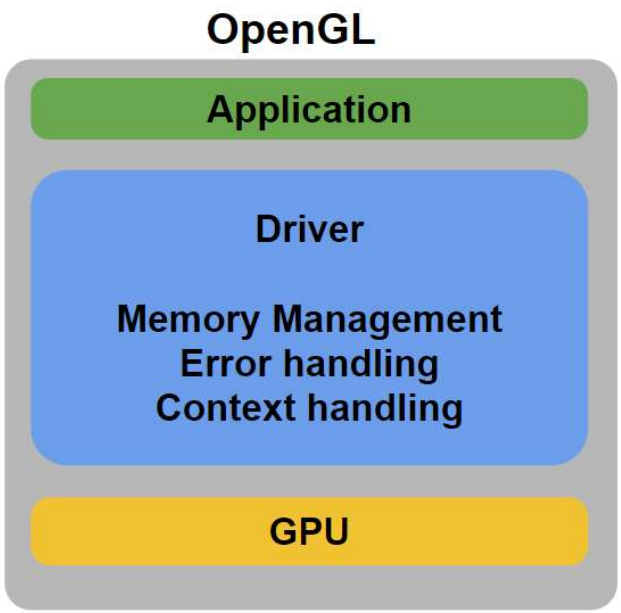
OpenGL vs Vulkan

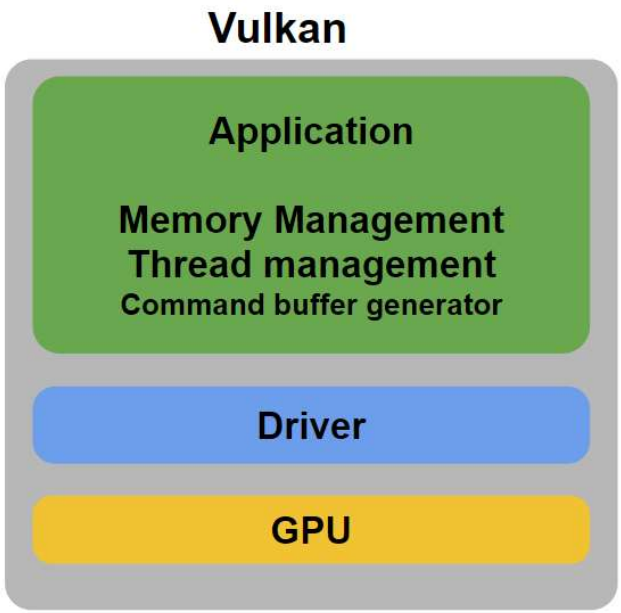
Complex drivers cause overhead and inconsistent behaviour across vendors

Always active error handling

Full preprocessor and compiler for shading language

OpenGL vs OpenGLES





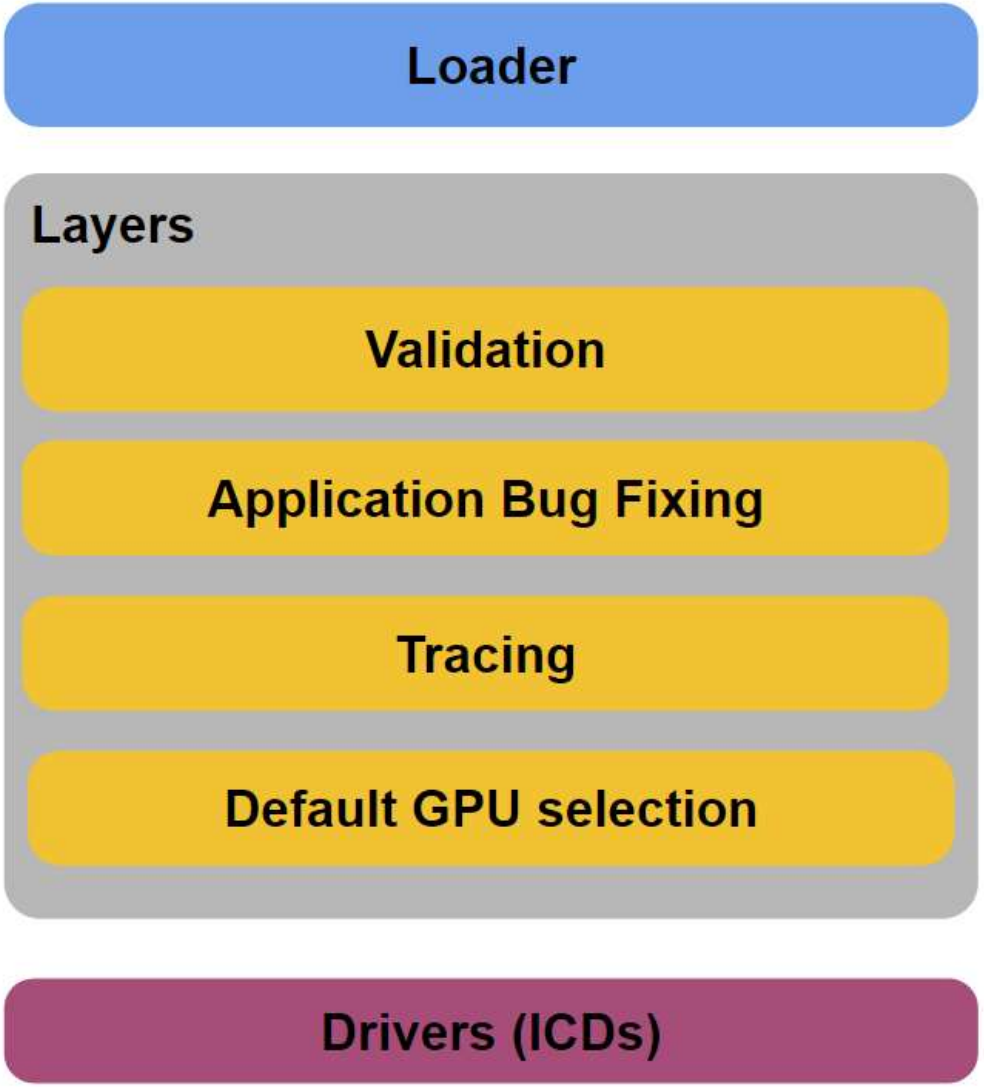
Low overhead driver
Consistent behaviour

Validate and debug layers can be loaded only when needed

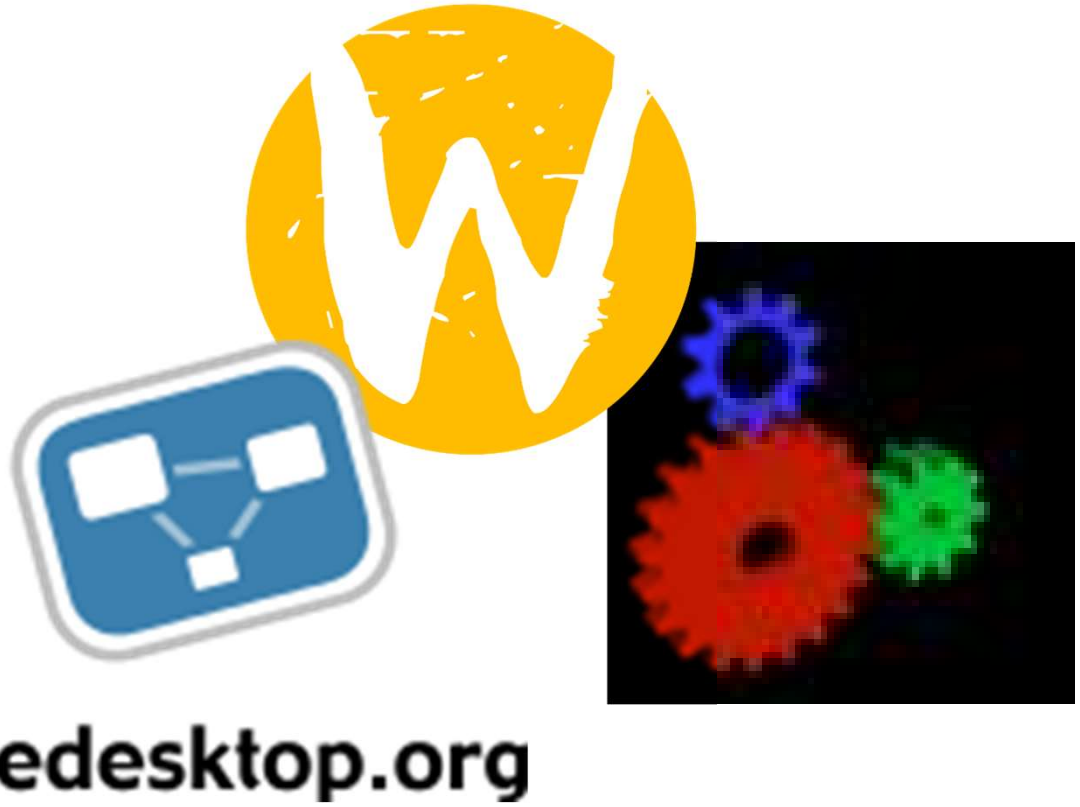
SPIR-V intermediate language

Unifued API

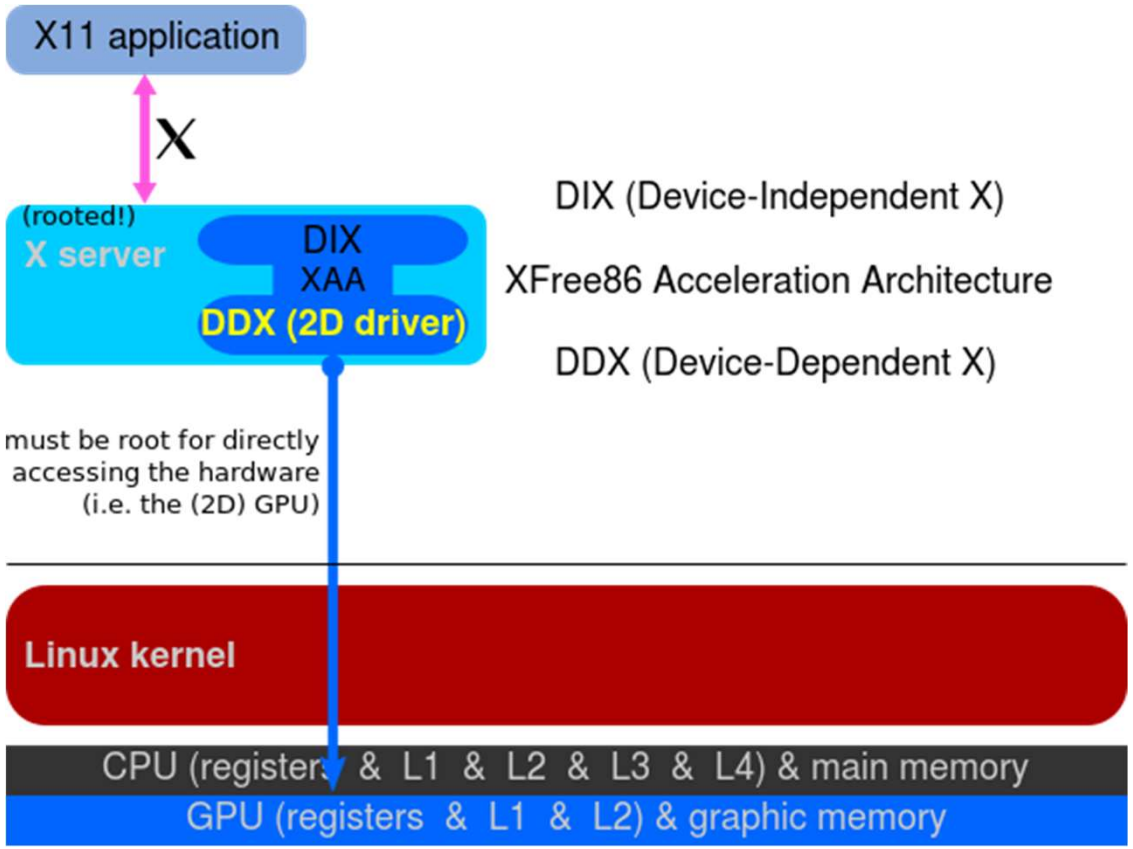
Vulkan Stack



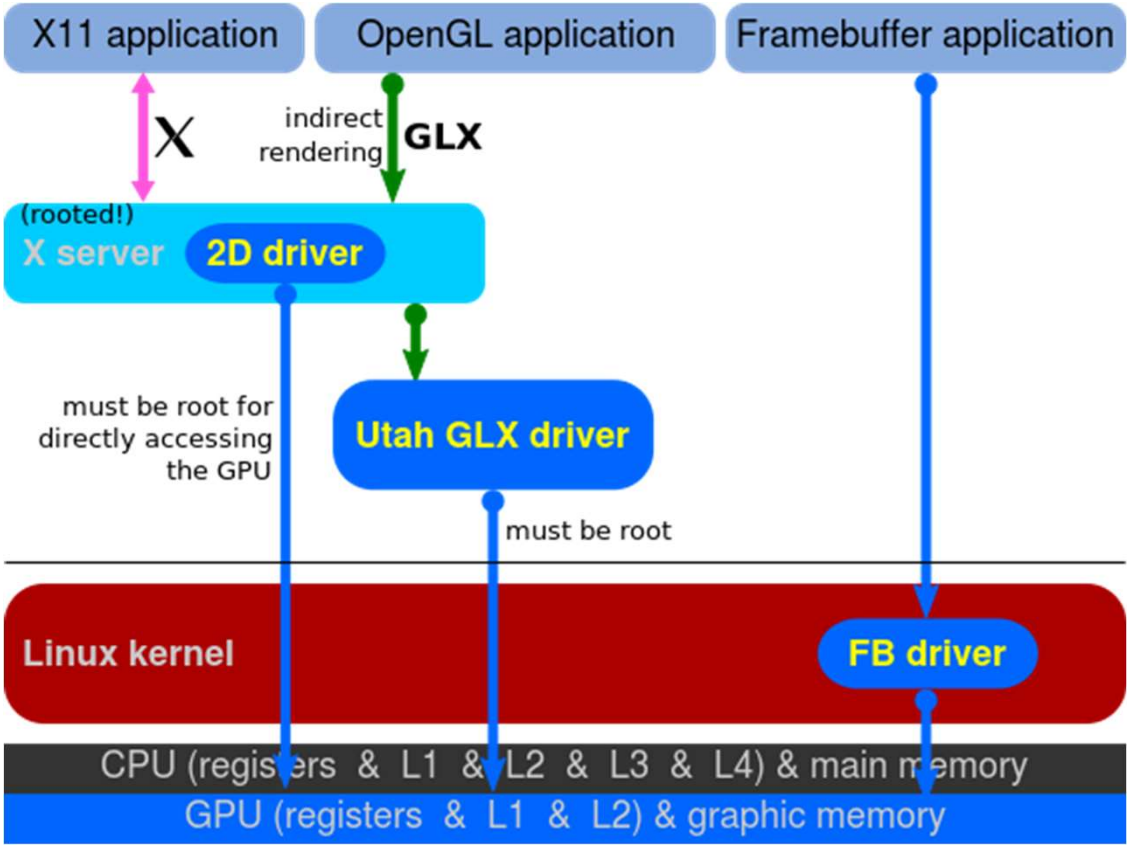
History of X Server Acceleration



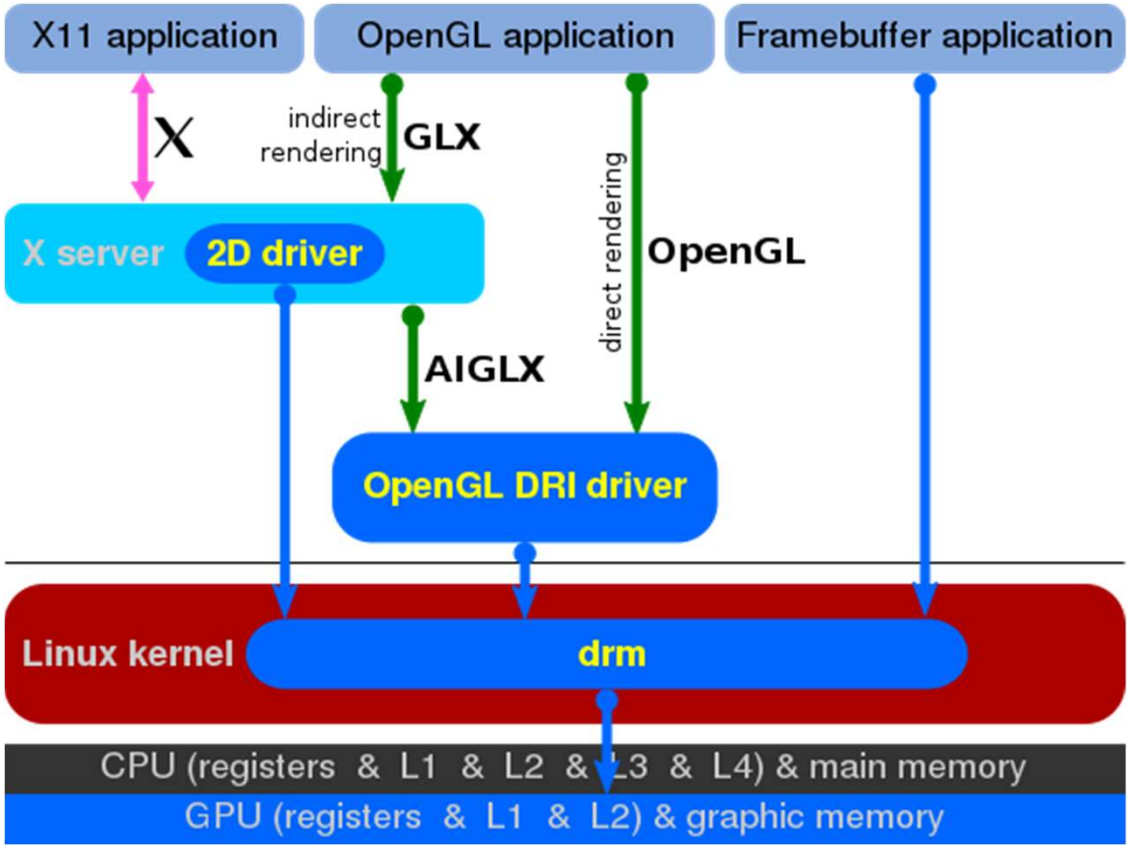
No Accel


X11 application
X
(rooted!)
X server
DIX
XAA
DDX (2D driver)
DIX (Device-Independent X)
XFree86 Acceleration Architecture
DDX (Device-Dependent X)
must be root for directly accessing the hardware (i.e. the (2D) GPU)
Linux kernel
CPU (registers & L1 & L2 & L3 & L4) & main memory
GPU (registers & L1 & L2) & graphic memory

In Direct Rendering



In Direct Rendering + Texture_From_Pixmap



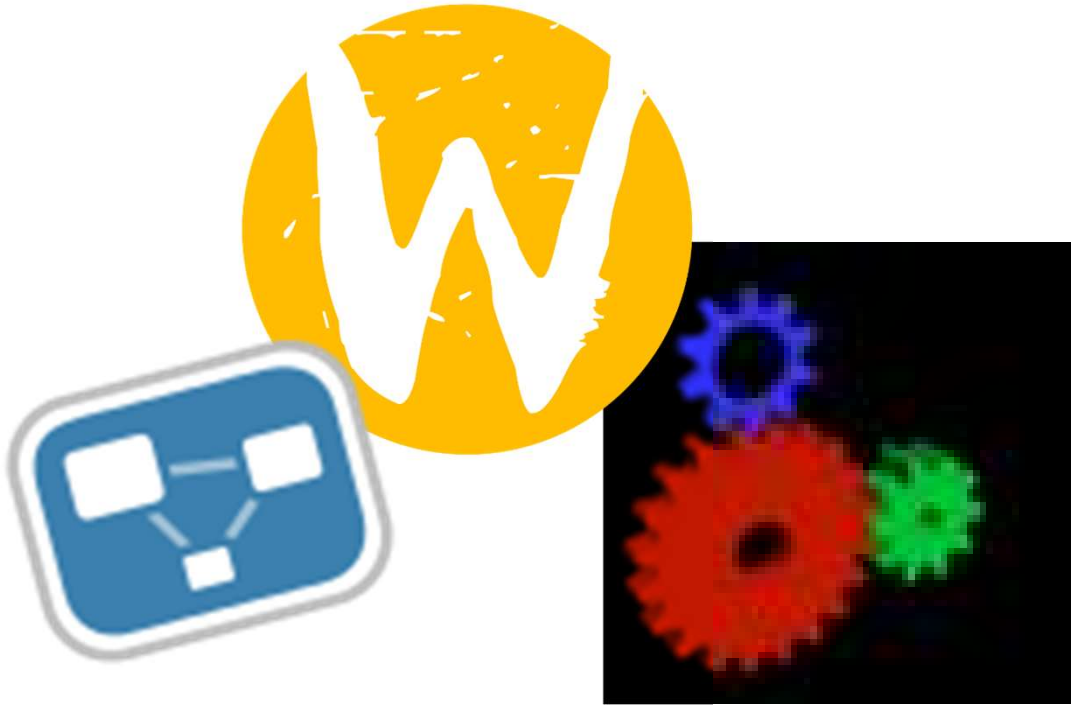
AIGLX(Accelerated Indirect GLX)

X Video Driver (Glamor)

- What is Glamor?
  - The glamor module is an open-source 2D graphics common driver for the X Window System as implemented by X.org. It supports a variety of graphics chipsets which have OpenGL/EGL/GBM supports.

- It's a GL-based rendering acceleration library for X server
  - It uses GL functions and shader to complete the 2D graphics operations.
  - It uses normal texture to represent a drawable pixmap if possible.
  - It calls GL functions to render to the texture directly. It's somehow hardware independent. And could be a building block of any X server's DDX driver:
  - Xorg's DDX driver could leverage glamor-egl package to create an egl context without any native X system. The xf86-video-modesetting driver uses glamor by default but other drivers have support for it (xf86-video-ati and xf86-video-intel). This package can support every platform which has OpenGL and gbm and drm libraries.

Reasons For Moving To Wayland



freedesktop.org

Reasons For Moving To Wayland (1/4)

		X	Wayland
Rendering	Support for H/W Overlay by WM	No	Yes
	Support for Drawing API	Yes	No
Input	Resource Limitation	255 Keycodes	**in theory, 2^{32}-1**
	Support for Server-side Input Transform	No	Yes
Security	Protocol Code	Hand written binary protocol parsing code	Generated by wayland-scanner
	Input Event	Different clients can receive events on the same window and add other events.	Server sends corresponding events to a client.
Extensibility	Ease of Porting	Hard to adapt	Relatively easy
	Extension of protocol	Hard to add while minimizing impact to existing system functions (DIX, DDX)	Relatively easy
Event System	Race condition	Client – Server – WM	Client – Server (WM)

Reasons For Moving To Wayland (2/4)

- The Fixings Of Wayland
  - The entire protocol is versioned.
  - No drawing API
  - Composition is mandatory under Wayland
  - As a security precaution: Screensaver and locker are apart of the compositor

- A Few Generic Advantages Of Wayland
  - Every frame is perfect.
  - Minimal
  - H/W specific backends.

Reasons For Moving To Wayland (3/4)

- Protocol Code
  - X Server Security Disaster: “It’s Worse Than It Looks” http://www.phoronix.com/scan.php?page=news_item&px=MTU1NzA
  - X11 Protocol Code: 30 year old hand written binary protocol parsing code
    - GLX protocol: 80,000 lines of insecure codes
    - XCB to generate server-side protocol handling & byte swapping
      - X can’t make it possible for all the clients to move to XCB, since missing pieces.
  - Wayland Protocol Code: XML parser generated code

- Screen Locking
  - In X: It requires lots of tricks, and even then, you are not absolutely guaranteed.
  - In Wayland: It can be implemented in the display server which also manages all windows and input.
  - Desktop-shell interface "privileged", which means that only a known client can bind to it. The easiest way to have a known or authenticated client is to have the compositor execute the client directly, and set up a secure communication channel to it, allowing only this client to bind to the interface.
  - http://ppaalanen.blogspot.kr/2011/11/wayland-desktop-shell.html

Reasons For Moving To Wayland (4/4)

- Why X Is Not Our Ideal Window System
  - http://www.std.org/~msm/common/protocol.pdf
  - Race Conditions

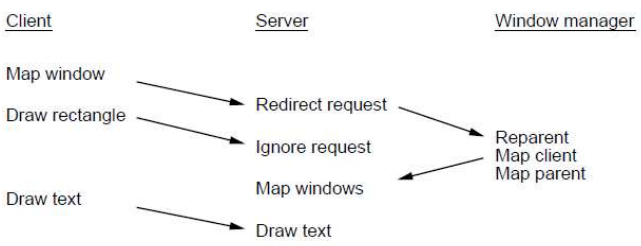


Figure 3-7: Map window race

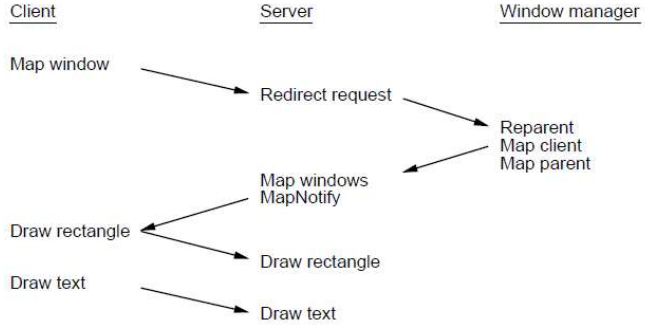


Figure 3-8: MapNotify solves the map window race

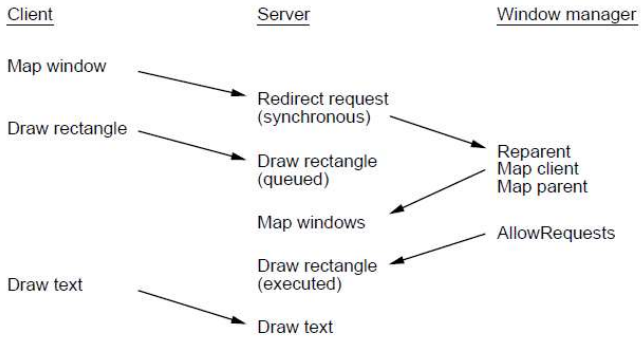
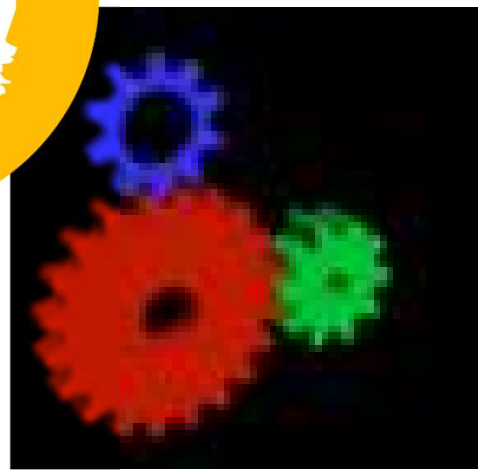


Figure 3-9: Synchronous redirection solves the map race

Thank you

freedesktop.org



Mesa 3D