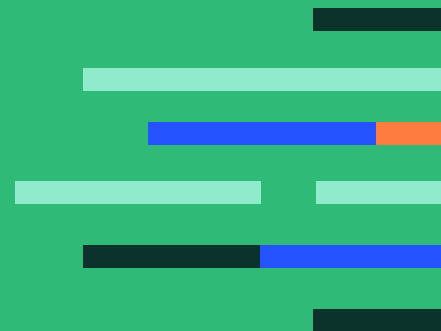




A Retrospection on 5 Years of `simplifiedrm`



“Being started in Hack Week 19, the `simplifiedrm` graphics driver reached the upstream kernel in 2021. Since then it has been adopted by all major Linux distributions. `Simplifiedrm` is nowadays the umbrella term for the Linux kernel's boot-up graphics. It's a success story out of SUSE Labs that has quietly impacted the Linux graphics stack.”

About this presentation

- About me: *“I’m Thomas. I do graphics drivers at SUSE.”*
- I presented about boot-up graphics an SUSE Labs Conf 2022 with a then-new driver called `simplifiedrm`.
- “Modern Graphics from Boot to Shutdown and Retiring fbdev”
 - <https://www.youtube.com/watch?v=WbAVXl2pUdc>
- Which is now (mostly) complete.
- This is a wrap-up talk to summarize and close the topic.

We need some sort of output during boot

(serial doesn't count)

- Unlocking disk encryption
- Errors and debugging
- Graphics fluff



How it works

- Firmware sets up a system framebuffer during boot
 - Memory aperture with pixel data
 - Resolution
 - Color format
- Kernel picks up the configuration and makes it available as graphics device
- Sometime later during boot, the boot-up graphics is being replaced by the hardware's native DRM driver
- Works with any common firmware

Until 5 years ago...

... the kernel would bind an fbdev driver to the system framebuffer,

- which is unmaintained,
- which does not always support 32-bit pixel formats,
- which does not support buffer sharing,
- which is only supported by fbcon and the aging Xorg.

Motivation

- Can we have a graphics stack without fbdev drivers?
- Hackweek 19 (Feb 2020)
 - <https://hackweek.opensuse.org/25/projects/support-for-drm-platform-drivers>

“This [hackweek] project could get us rid of the last fbdev drivers we're still shipping: efifb and vesafb.”

“It would be nice to support VESA and UEFI with modern DRM drivers. Doing so would give us basic support for very old, very modern or very obscure graphics hardware.”

Adding simpledrm and ofdrm

- **simpledrm:**
 - Commit 11e8f5fd223b ("drm: Add simpledrm driver")
 - Apr 2021
 - Supports DeviceTree, UEFI, VESA
 - Replaces simplefb, efifb, vesafb
- **ofdrm:**
 - Commit c8a17756c425 ("drm/ofdrm: Add ofdrm for Open Firmware framebuffers")
 - Oct 2022
 - Supports OpenFirmware
 - Replaces offb

Some history

Early adoption

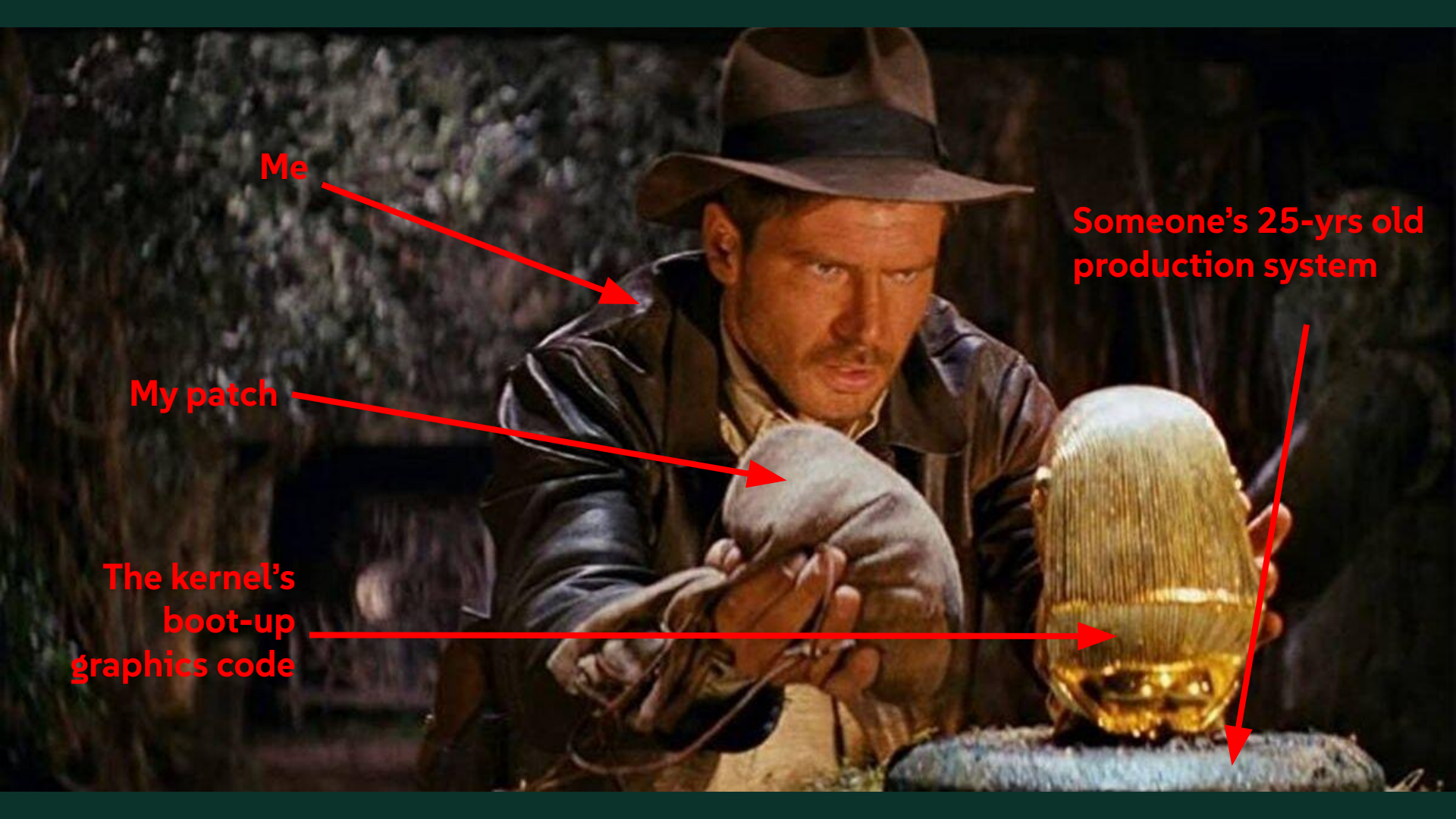
- Distributions started switching to `simplifiedrm` already in 2021/2022
 - Tumbleweed
 - Fedora
 - Alpine



More history

2022 to 2025

- Continued adoption by other distributions
 - Slow movers
 - Enterprise distros
- Constant fixing and improving ever since
 - More drivers
 - More features
 - Tuning
- As a side effect, we improved Linux' overall support for boot-up graphics.



Me

Someone's 25-yrs old
production system

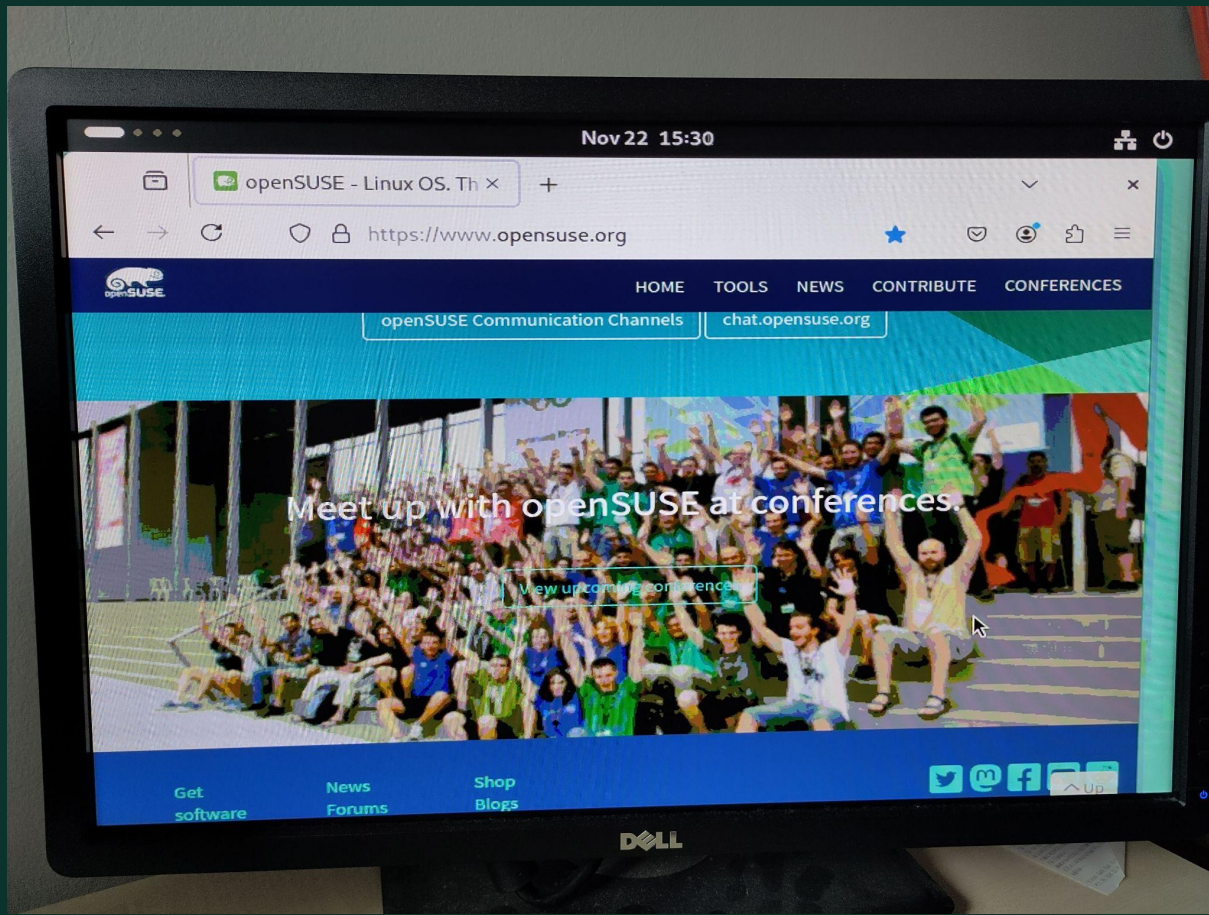
My patch

The kernel's
boot-up
graphics code

Even more history

Adding efidrm, vesadrm and corebootdrm

- **efidrm**
 - Commit 32ae90c66fb6 ("drm/sysfb: Add efidrm for EFI displays")
 - Apr 2025
- **vesadrm**
 - Commit a84eb6abe2b6 ("drm/sysfb: Add vesadrm for VESA displays")
 - Apr 2025
- **corebootdrm**
 - Commit a29a1f0ec8d6 ("drm/sysfb: corebootdrm: Add DRM driver for coreboot framebuffers")
 - Feb 2026
- These drivers provide additional features that can't be added to `simplifiedrm` easily.



Meanwhile

Hardware support

- Linus uses an Apple M2 (as early as Jul 2022)
 - https://lore.kernel.org/lkml/CAHk-=wgrz5BBk=rCz7W28Fj_o02s0Xi0OEQ3H1uQgOdFvHgX0w@mail.gmail.com/
- There was no (serious) native graphics driver at that point AFAIK
- But there was `simplifiedrm`
 - We applied patches for the display's XRGB2101010 pixel format in Dec 2021
- Indicates that boot-up graphics can run productive environments in the real world

Meanwhile

"X' dead, Jim."

- libweston: remove fbdev backend (Mar 22)
 - <https://gitlab.freedesktop.org/wayland/weston/-/commit/b3ba1becba806a3557b8acbf8c42f1782cffdea2>
- [...] GNOME 49 Preps for X11 Disabled-by-Default (Jun 25)
 - <https://www.phoronix.com/news/GNOME-49-Alpha-0-Packages>
- Ubuntu drops support for GNOME on Xorg (Jun 25)
 - <https://discourse.ubuntu.com/t/ubuntu-25-10-drops-support-for-gnome-on-xorg/62538>
- KDE Plasma 6.8 Set To Drop X11 Support Completely (Nov 25)
 - <https://itsfoss.com/news/kde-plasma-to-drop-x11-support/>
- After 15+ yrs, everyone's finally killing off legacy graphics stacks!

History and development

At long last, a sane graphics stack.

- Until 2 yrs ago, fbdev provided the boot-up graphics and by extension, the fallback in case the hardware's native driver is not available
- What is the only user space with any meaningful fbdev support? Xorg!
- So desktops had to keep Xorg support as a fallback
- With a DRM-only graphics stack, clear out all the legacy graphics support and run the Wayland compositor directly

“Since SimpleDRM got merged, I no longer am going to complain about the fbdev backend going away.” --

https://gitlab.freedesktop.org/wayland/weston/-/work_items/581#note_1225248

Conclusions

What didn't work.

- We should have started with more individual drivers
 - simpledrm reached its limits early; hence efidrm, etc
- DRM is still a bit on the high-end side
 - Helpers for low-end devices get written as they are needed
 - Hard to change later on
 - DRM default feature set is for higher end; emulation on lower end needed
 - Hard to find reviewer this these changes

Conclusions

What did work.

- Overall a huge success
 - Corollary: **Hackweeks make a difference!**
- A modern Linux graphics stack on almost any graphics hardware
 - Old and broken
 - Brand new
- Adding this new use case expanded DRM's capabilities and design
 - How do we emulate missing features?
- Fixed other components of the graphics stack
- As a side effect, there's less maintenance burden and technical debt in other components.
 - Modernizing GNOME (Oct 2025): https://www.youtube.com/watch?v=uCAIzx_x6rY
- Go the extra mile and remove legacy interfaces

Next steps

- Xwayland will provide X11 support in the future
- Build a console without fbdev
- Add more drivers/modes, if we ever have to



SUSE

Earlier attempts

- Some work done 10+ years ago by David Herrmann
 - XDC 2014: <https://www.youtube.com/watch?v=GTviGLOdd-E>
- Prototypes of a driver, DRM logging, user-space console
- Never got finished or merged

Fixing Xorg, Part 1

Running on sysfb devices

- To use the driver, open `/dev/dri/card0` and run `ioctl` on it
- But Xorg is from a past era when all this was in user space
 - Xorg parses the PCI bus and tries to match PCI devices to DRM device files
 - Ignores device files for non-PCI hardware
 - Because it expects them to be served by `fbdev`
 - Happens with any non-PCI hardware
- Patch Xorg to accept devices with specific DRM drivers
 - `"if (strcmp(driver, "simplifiedrm")) { accept_device_file(); }"`

Correctly Modelling the Device Hierarchy

Devices and Drivers

- Fbdev modules set up their own Linux device to which they bound
- Before we merged and DRM driver, we moved this into a few places in the architecture and firmware code
- During boot, Linux now creates a device for each detected system framebuffer
- Drivers bind to these sysfb devices as configured
 - Either new DRM drivers
 - Or fbdev drivers
 - Or a combination of both

Correctly Modelling the Device Hierarchy

Dependencies and unplugging

- The system framebuffer depends on the PCI hardware
- But the old fbdev drivers do not track device dependency correctly
- Unplugging the PCI device left the system framebuffer dangling
 - SIGBUS
- We try to detect the PCI device and make the sysfb device a child
- No unplugging while the child device still exists

Fixing Xorg, Part 2

Running on sysfb devices

- Re-parenting affects sysfs paths
- Xorg looked at the wrong sysfb path and failed to load a driver
 - Again tests PCI vs. something else
- Fix was easy; use the correct path
 - https://gitlab.freedesktop.org/xorg/xserver/-/merge_requests/1612
- Also fixed a much older bug with USB devices by accident
 - https://gitlab.freedesktop.org/xorg/driver/xf86-video-fbdev/-/work_items/9

Handing over to the native driver

The old way

- System framebuffer will be gone after loading the native driver
- Better remove it in a controlled way
- Matching between system framebuffer and native driver happens with framebuffer apertures
- IOW the system framebuffer must be somewhere in the video memory seen by native driver
- The original removal code walked over the registered fbdev framebuffers and removed the conflicting framebuffer
 - But forgot to remove the underlying Linux device
- But we just removed the fbdev framebuffer from the setup... oops

Handing over to the native driver

The new way

- We added dedicated aperture helpers to manage ownership
- Simpledrm et al register their Linux device as owner of the framebuffer aperture
- Native driver does a common hot unplug of any such devices in its video memory
 - Gets rid of the sysfb device
 - Makes unplugging in user space work correctly (udev events, etc)

Implementing nomodeset

An easy workaround for driver problems

- Display output is fundamental to using a computer, we cannot afford to not display something.
- No need to make hand over mandatory
- Pass **nomodeset** on the kernel command line to avoid loading native drivers
- Users can avoid broken hardware or drivers until a fix is available
- Think of Safe Mode under Windows
- Turned out to be fairly popular in Linux help forums : *“Try booting with nomodeset.”*

Fixing fbdev

- We then went through all old fbdev drivers and made them support aperture helpers and nomodeset
 - Handle aperture ownership
 - Do not load if nomodeset has been given
- All of the kernel's graphics drivers now share the same logic when interacting with each other during boot

Supporting and emulating color formats

- What happens if the system framebuffer has an odd color format?
 - Palettized colors
 - 15/24-bit colors
- Solution: Export the native format and emulate XRGB8888 on top
 - Native format is fast
 - XRGB8888 is compatible with common user space
 - Emulation converts XRGB8888 to the native format internally
 - Copies the result into the system framebuffer
- Result of a lengthy discussion, but now generally adopted throughout DRM drivers.
 - Panels
 - Odd hardware

Reading Monitor Characteristics

Aka EDID support

- Most system interfaces support EDID
- We make it available with the system framebuffer
- Fbdev never did that, but DRM has good support
- We finally use struct boot_params.edid_info after ~25 yrs
- Grub patch
 - <https://gitlab.freedesktop.org/gnu-grub/grub/-/commit/e1b9d92a8be7eb99e9f963a4b5280fea85b49877>
- Useful for size calculations, gamma correction and palette modes

Fixes to fbcon

- Fbcon support got more flexible
- [Nvidia-drm.ko](https://www.nvidia.com/en-us/drivers/linux/) finally got a real DRM-based console



SUSE