linaro™

Arm Solutions at Lightspeed

AOSP Bring-up Using Software Rendering

Amit Pundir

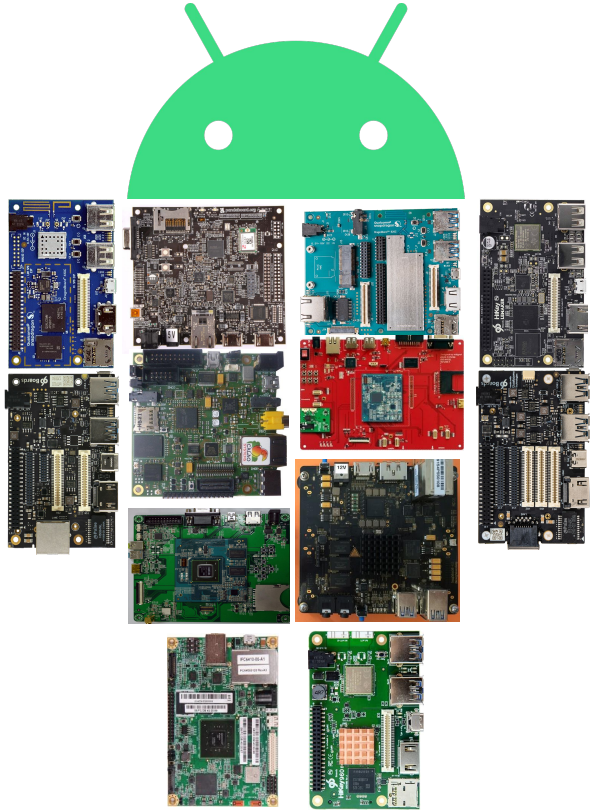
aosp-devroom, FOSDEM ’25

About Me!

- Senior Engineer at Linaro
- 10+ years of **AOSP** (**A**ndroid **O**pen **S**ource **P**roject) bringup and maintenance
- pundir on #aosp-developers, #linaro-android IRC channels at OFTC.net

Agenda!

- Software Rendering in AOSP
  - Why?
  - How?



Why Software Rendering?

Software rendering support in AOSP is useful for several reasons:

- Missing hardware acceleration (GPU) support
  - Running AOSP on virtual platforms or on the devices with no GPU support or on low-end devices with limited GPU capabilities
- Missing or limited software support
  - Running AOSP on an early device prototype with limited s/w support
  - Broken software support due to incompatible (legacy) vendor blobs
- Headless devices: Embedded / IoT / low-power devices
  - Devices that do not require or include a display but may still need some form of graphical rendering to interact with Android's internal UI components.
  - Software rendering maybe be more power efficient for devices which avoid heavy reliance on graphics hardware for complex computations
- Testing and Development
  - To debug or isolate GPU related bugs

Software rendering with SwANGLE!

AOSP recommends using SwiftShader and ANGLE libraries for software rendering

- SwiftShader's GL libraries got deprecated, and replaced with **ANGLE** (**A**lmost **N**ative **G**raphics **L**ayer **E**ngine)
- **SwANGLE**: **ANGLE** (GLES implementation) on top of **Sw**iftShader's Vulkan implementation (Pastel)

```
# TODO(b/65201432): Swiftshader needs to create executable memory.
PRODUCT_REQUIRES_INSECURE_EXECMEM_FOR_SWIFTSHADER := true

# ANGLE provides an OpenGL implementation built on top of Vulkan.
PRODUCT_PACKAGES := \
    libEGL_angle \
    libGLESv1_CM_angle \
    libGLESv2_angle \
    vulkan.pastel

PRODUCT_VENDOR_PROPERTIES := \
    ro.hardware.egl=angle \
    ro.hardware.vulkan=pastel
```

linaro_swr-trunk_staging-userdebug

- Reference: linaro_swr-trunk_staging-userdebug target in AOSP
  - A generic AOSP build target using software rendering
    - Developed and tested on Qcom target devices, but generic enough to boot on any arm64 platform
  - Use SKIA GL instead of the default Vulkan backend otherwise we see a lot of display glitches due to missing sync support

```
PRODUCT_VENDOR_PROPERTIES := \
    debug.hwui.renderer=skiagl
```

  - For rendering simplicity, we disable the compressed image format

```
PRODUCT_VENDOR_PROPERTIES := \
    vendor.minigbm.debug=nocompression
```

SwANGLE with virtual drm/kms driver!

- AOSP bring-up using Virtual KMS (vkms) driver and SwANGLE
  - Enable VKMS display driver module
  - Set hwcomposer to use the correct display

```
PRODUCT_VENDOR_PROPERTIES := \
    vendor.hwc.drm.device=/dev/dri/card0
```

  - Label /dev/dri/cardX a gpu_device to keep selinux Gods happy
  - Disable prime shader cache
    - Speeds up the boot time significantly by disabling the shader pre-runs before booting up to UI

```
PRODUCT_PROPERTY_OVERRIDES := \
    service.sf.prime_shader_cache=0
```

Amit Pundir <amit.pundir@linaro.org>
IRC: pundir on #linaro-android, #aosp-developers @OFTC
aosp-devroom, FOSDEM ‘25