Tyr: a new Rust GPU driver

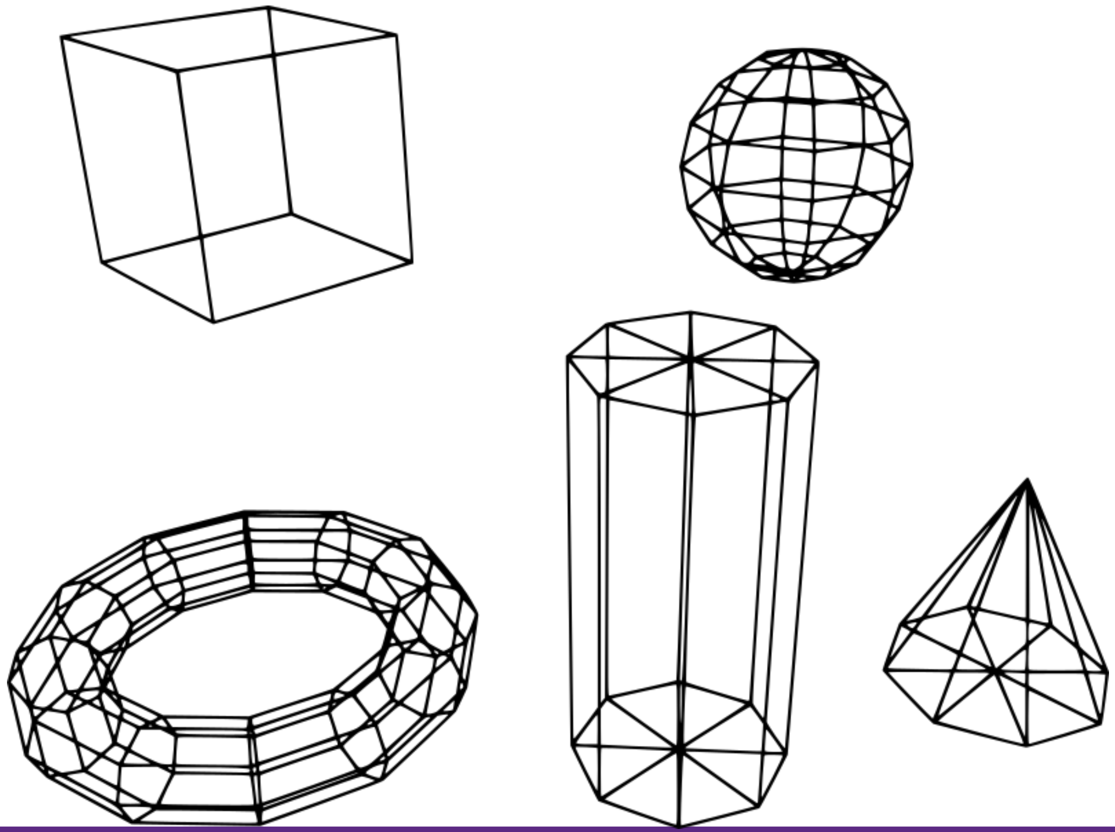
Alice Ryhl (Google)

Daniel Almeida (Collabora)

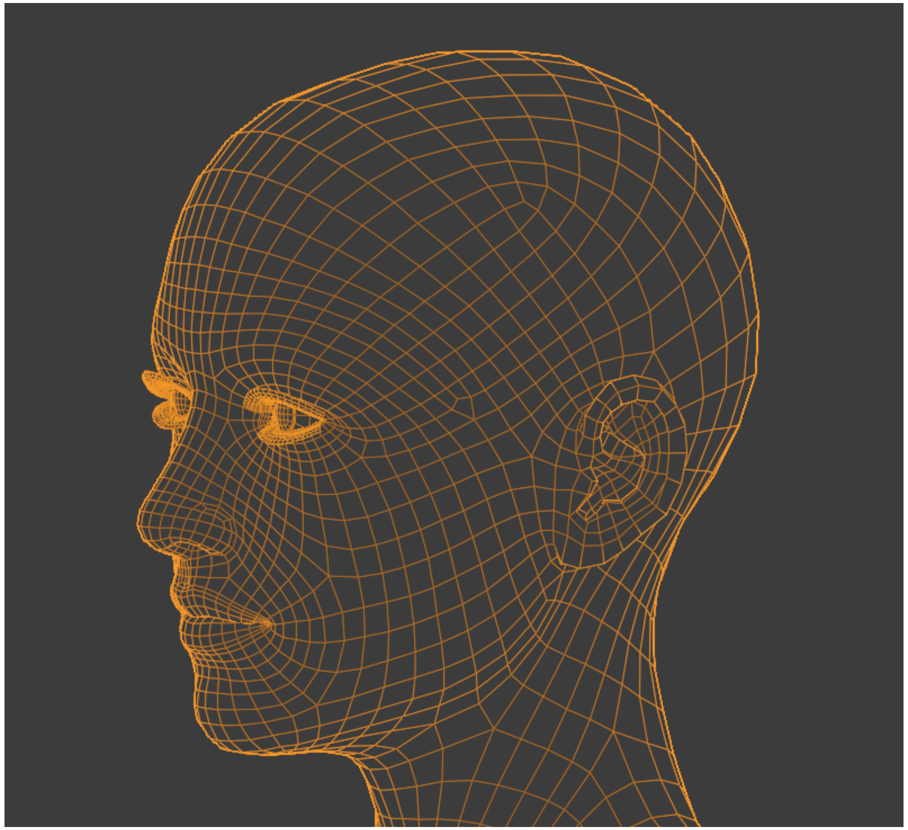
In this talk..

- Briefly discuss how GPU drivers work
  - Assumes VkCube and Vulkan
  - Discuss UMD vs KMD
- Discuss the KMD uAPI
- Discuss Arm’s CSF architecture and how Tyr works with it
- First triangle?
- Future plans

Simple meshes

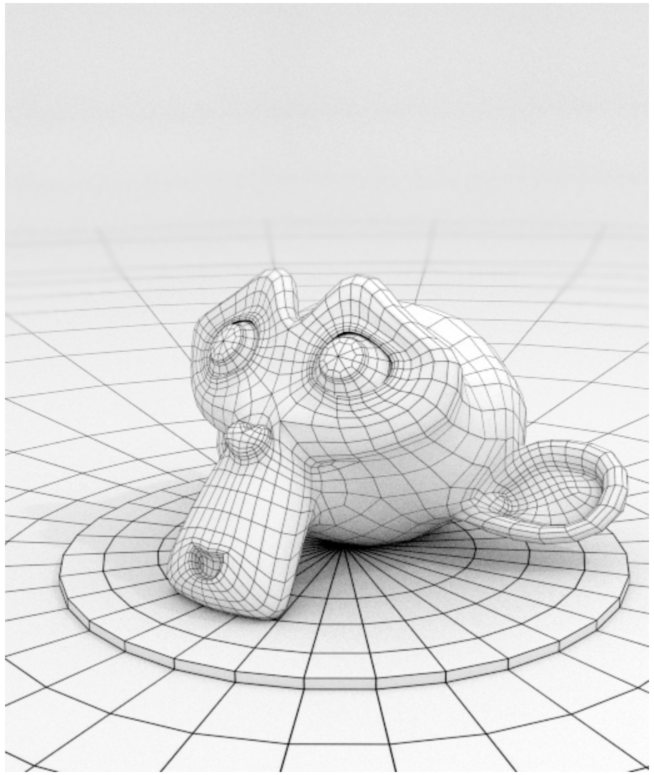


More elaborate meshes



https://
blender.stackexchange.com/
questions/213611/mesh-looks-
ok-in-wireframe-but-shading-is-
distorted

More elaborate meshes



https://blenderartists.org/t/suzanne-on-wire/678653

Textures

https://www.poliigon.com/texture/reclaimed-dutch-bond-brick-wall-texture/8320

Shaders

- Full blown programs
- At the very minimum, places the object on scene
- May apply any number of effects, like e.g.: rotation
- Executed by the shader cores in the GPU

Shaders

```
/* vkcube.vert */
#version 420 core

layout(std140, set = 0, binding = 0) uniform block {
    uniform mat4 modelviewMatrix;
    uniform mat4 modelviewprojectionMatrix;
    uniform mat3 normalMatrix;
};

layout(location = 0) in vec4 in_position;
layout(location = 1) in vec4 in_color;
layout(location = 2) in vec3 in_normal;

vec4 lightSource = vec4(2.0, 2.0, 20.0, 0.0);

layout(location = 0) out vec4 vVaryingColor;

void main()
{
    gl_Position = modelviewprojectionMatrix * in_position;
    vec3 vEyeNormal = normalMatrix * in_normal;
    vec4 vPosition4 = modelviewMatrix * in_position;
    vec3 vPosition3 = vPosition4.xyz / vPosition4.w;
    vec3 vLightDir = normalize(lightSource.xyz - vPosition3);
    float diff = max(0.0, dot(vEyeNormal, vLightDir));
    vVaryingColor = vec4(diff * in_color.rgb, 1.0);
}
```

COLLABORA

Open First

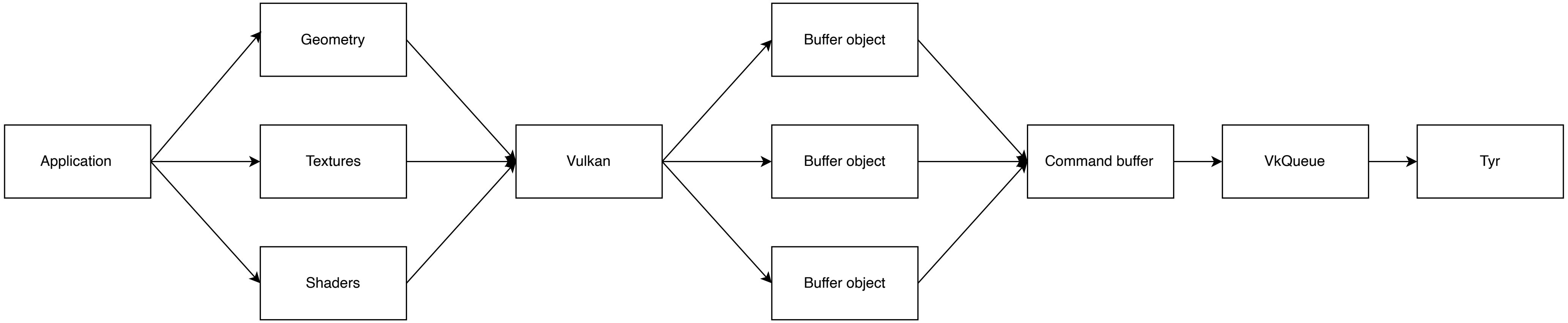
Shaders

```
/* vkcube.frag */
#version 420 core

layout(location = 0) in vec4 vVaryingColor;
layout(location = 0) out vec4 f_color;

void main()
{
    f_color = vVaryingColor;
}
```

COLLABORA

Open First

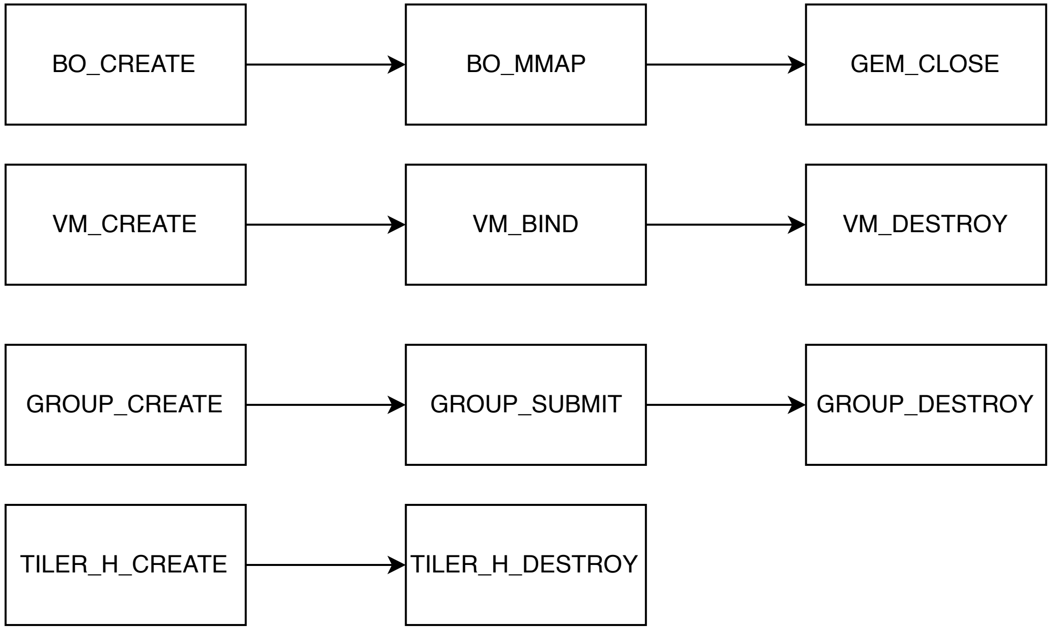
Overview


Application
Geometry
Textures
Shaders
Vulkan
Buffer object
Buffer object
Buffer object
Command buffer
VkQueue
Tyr

COLLABORA

The KMD offers a much simpler API

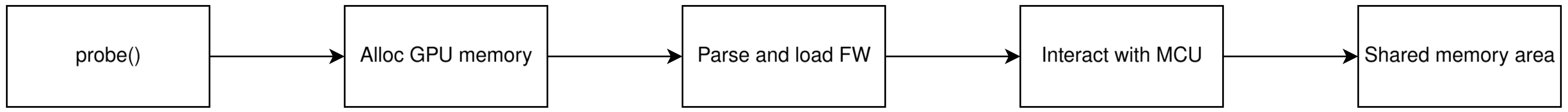
Kernel driver userspace API



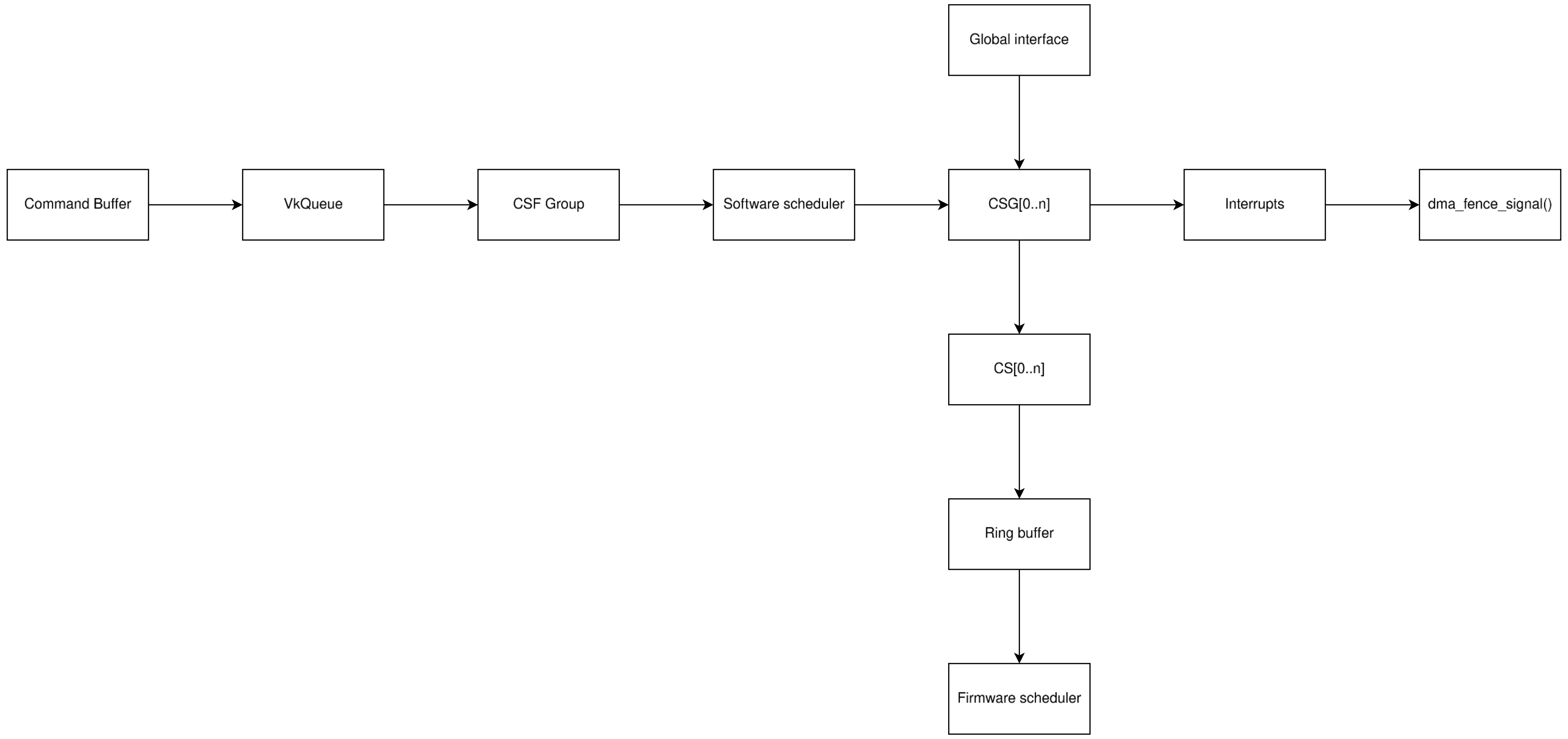
COLLABORA

Newer Mali GPUs use firmware-assisted scheduling

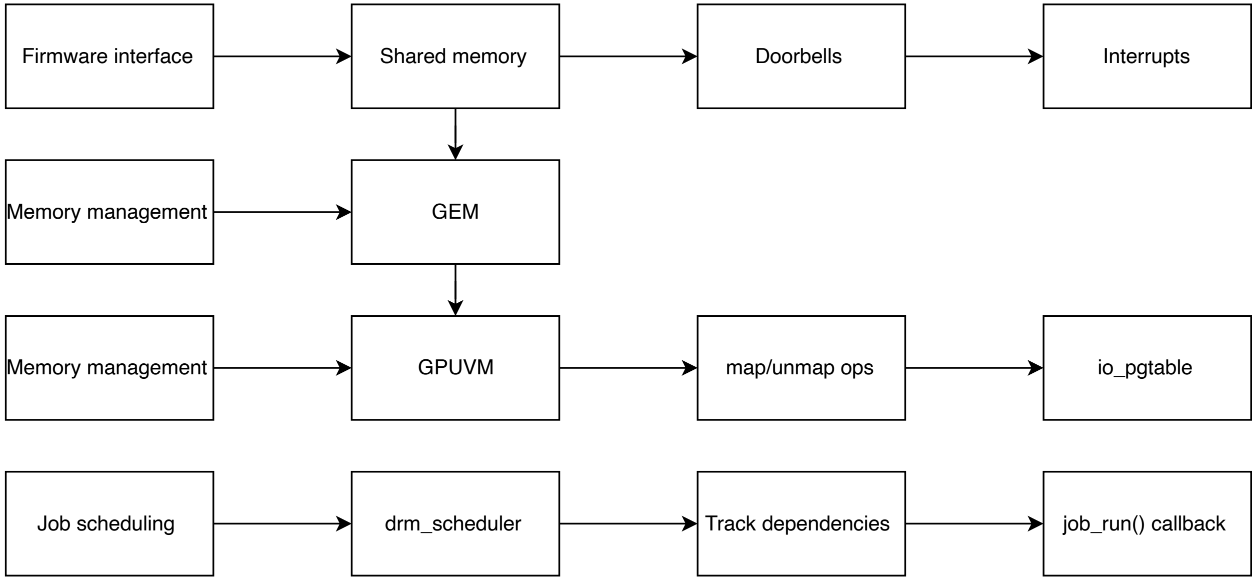
Booting the CSF MCU



Job submission and execution



Overview


Firmware interface
Shared memory
Doorbells
Interrupts
Memory management
GEM
Memory management
GPUVM
map/unmap ops
io_pgtable
Job scheduling
drm_scheduler
Track dependencies
job_run() callback

Merged Rust abstractions

- Clocks and regulators: get the HW ready
- IO memory: access the register interface
- Interrupts: receive notifications from the MCU
- Delayed works: implement the software scheduler tick

Work-in-progress abstractions

- GPUVM: VM management, map/unmap() ops
- gem_shmem: Shmem-backed GEM Objects
- drm_scheduler/job_manager: Track dependencies
- dma_fence/drm_syncobjs: Job synchronization

Downstream branch

- Boot the MCU
- VM management (Sync VM_BINDS only!)
- Group creation and job submission
- Bare-bones job synchronization (DRM_IOCTL_SYNCOBJ_WAIT works)

Upstream branch

- Probe() works
- No gem_shmem: can’t boot MCU
- No GPUVM: no VM management, also can’t boot MCU
- No drm_scheduler/job_manager: can’t submit work

COLLABORA

This only works on the rk3588 (i.e.: Mali G610, Valhall)

IGT

- GEM tests (BO_CREATE, BO_MMAP, etc)
- VM tests (VM_CREATE, VM_BIND, VM_DESTROY)
- Group creation and submission
- Initial patches submitted upstream

COLLABORA

Drawing our first triangle downstream

Drawing our first triangle

- Missing the tiler code
- Missing intra-job synchronization
- …that’s it?

COLLABORA

That’s it! We’re close :)

COLLABORA

Next steps

Downstream: next steps

- Draw a triangle (hopefully by LPC2025?)
- Get our first compute CTS test to pass
- Implement a software scheduler
- Deploy on actual devices for testing

Upstream: next steps

- Merge GPUVM
- Boot the MCU
- Implement BO ioctls
- Implement Sync VM_BINDS

Downstream: more long-term plans

- Implement power management (mobile devices need this)
- Implement a software scheduler
- Implement the reset logic (people tend to like this)
- Async VM_BINDS
- Augment IGT test suite

Downstream: more long-term plans

- Improve CTS results (should be similar to Panthor)
- Support for multiple Mali GPUs

Upstream: more long-term plans

- Discuss the drm_scheduler vs job_manager issue
- Submit jobs
- Pass the first CTS test in upstream code
- Power management
- Reset logic

Thank you!