

⌘ Unleash the (graphics) magic

Alyssa Rosenzweig



⌘ Status

⌘ XDC2022

- ~90% on GLES3

⌘ XDC2023

- Conformant GLES3.1
- OpenGL ~~3.1~~ 3.2!



⌘ Vulkan?

- ✓ Reverse-engineering
- ✓ Compiler
- ✓ Reference driver
- 👉 Vulkan: Your turn!



⌘ Geometry shaders

⌘ Why?

- OpenGL 3.2, Direct3D
- Real apps use geometry shaders (Blender!)
- Slow on most GPUs... just has to work

⌘ Mesh shaders?

- Doesn't help with transform feedback
- No hardware support until... Apple M3?
- Double emulation?



⌘ Idea



💡 Idea

- Run vertex + geometry shaders as ~compute
- Write outputs, index buffer, indirect draw
- Draw with a passthrough vertex shader



⌘ Where to write?

- Pipeline specified serially
- Need to write after all previous primitives
- Invocation n needs counts from $1..n - 1$

⌘ Idea 2

- Run vertex + geometry “count” shader
- Sum counts
- Run geometry shader
- Transform feedback works

⌘ Keeping it clean

⌘ Better `nir_builder`

- Complex `nir_builder` hard to maintain
- Don't write assembly – write C!
- Generated, seamless bindings



⌘ geometry.cl

```
void libagx_end_primitive(global int *indices, int total_verts,
                          int verts_in_prim, int total_prims,
                          int base_vert, int base_prim)
{
    int first_vert = base_vert + (total_verts - verts_in_prim);
    int first_prim = base_prim + (total_prims - 1);
    global int *out = &indices[first_vert + first_prim];

    for (uint i = 0; i < verts_in_prim; ++i) {
        out[i] = first_vert + i;
    }

    out[verts_in_prim] = -1;
}
```


⌘ Thank you

Alyssa Rosenzweig
Asahi Lina