



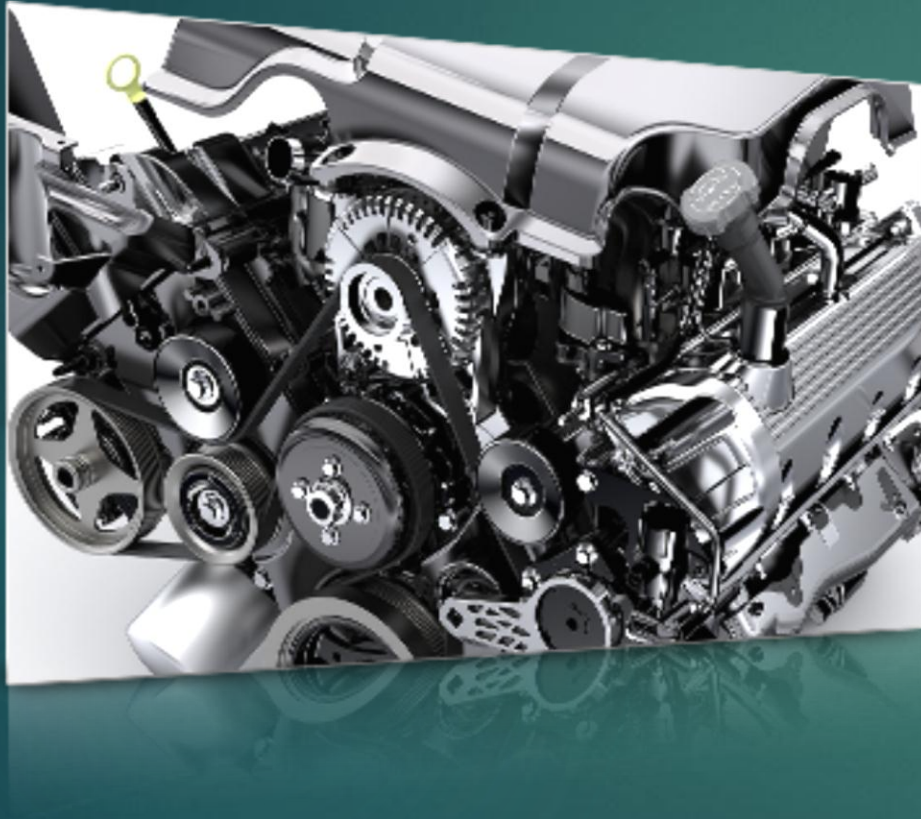
# GFX-rs

ARCHITECTURE

# Why?

- ▶ Safety in types and hardware states
- ▶ API abstraction
- ▶ Scalable performance
- ▶ Convenience

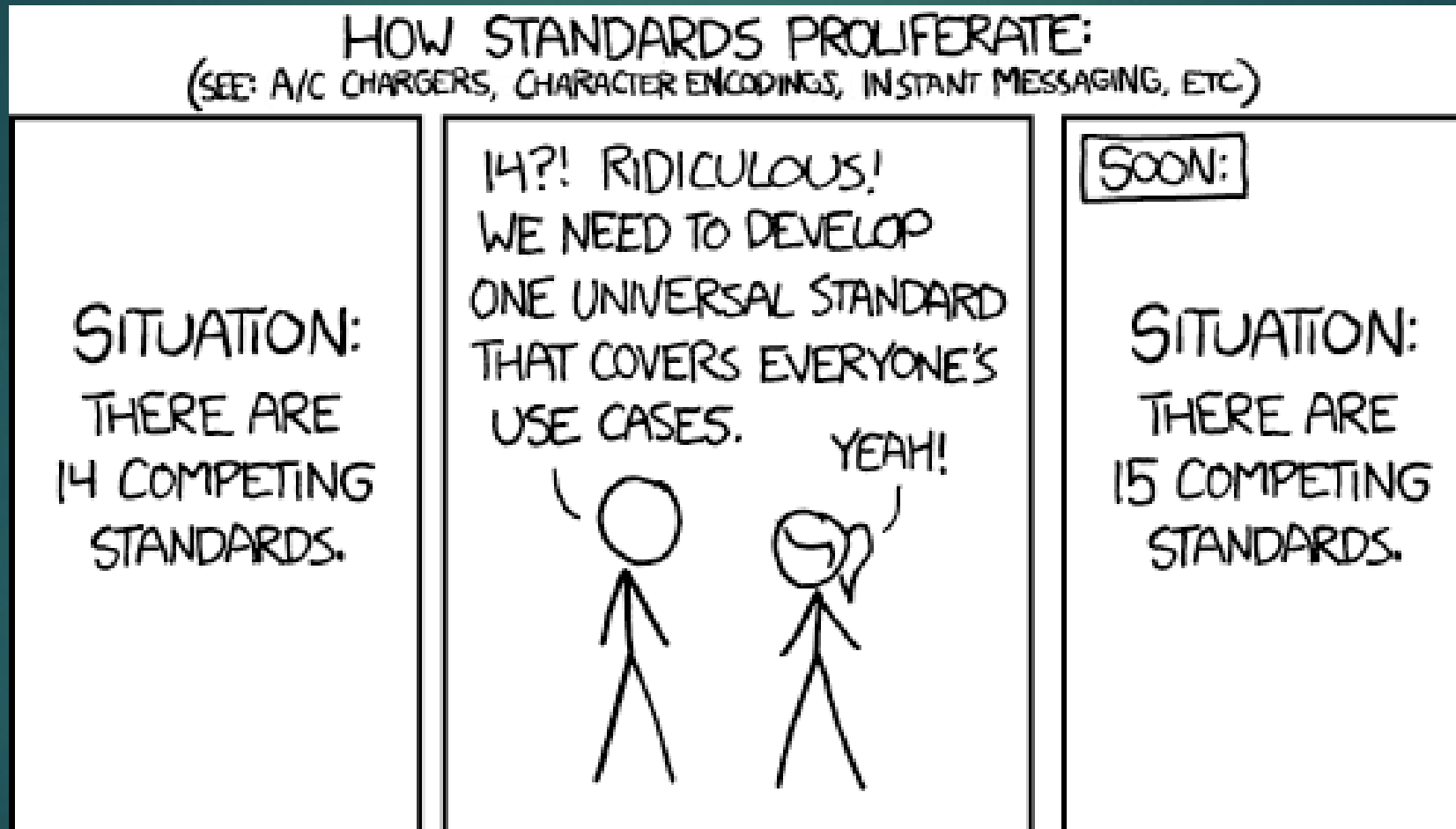
# 1. Safety



“Most bugs are a result of the execution state not being exactly what you think it is.”

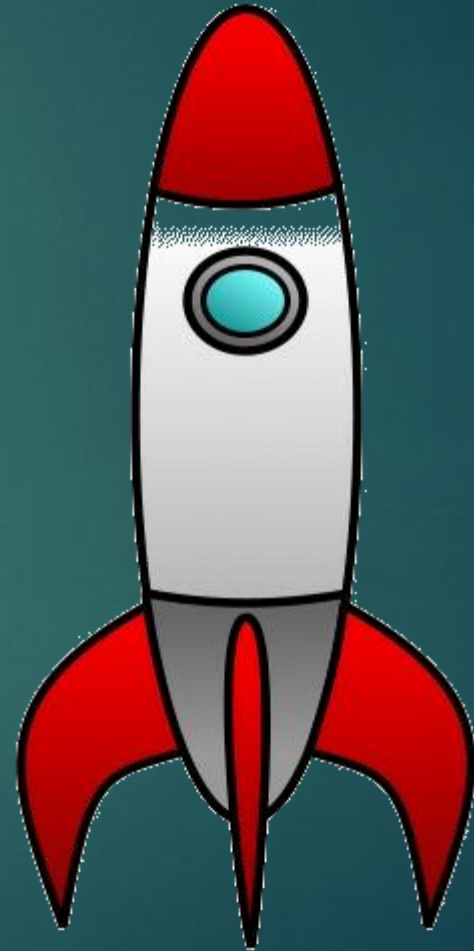
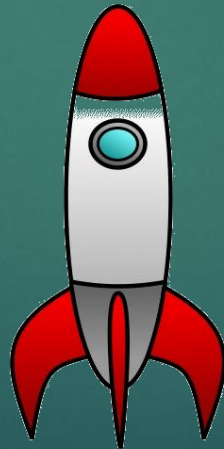
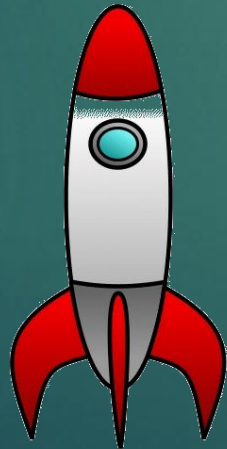
@John Carmack, 2007

## 2. Abstraction



### 3. Performance

- ▶ Automatic state caching
- ▶ Multi-threading



## 4. Convenience

```
#[vertex_format]
```

```
struct Vertex {
```

```
    #[as_float]
```

```
    #[name = "a_Pos"]
```

```
    pos: [i8, ..3],
```

```
    #[as_float]
```

```
    #[name = "a_TexCoord"]
```

```
    tex_coord: [u8, ..2],
```

```
}
```

```
#[shader_param(CubeBatch)]
```

```
struct Params {
```

```
    #[name = "u_Transform"]
```

```
    transform: [[f32, ..4], ..4],
```

```
    #[name = "t_Color"]
```

```
    color: gfx::shade::TextureParam,
```

```
}
```



```
// sleep(1) //
```

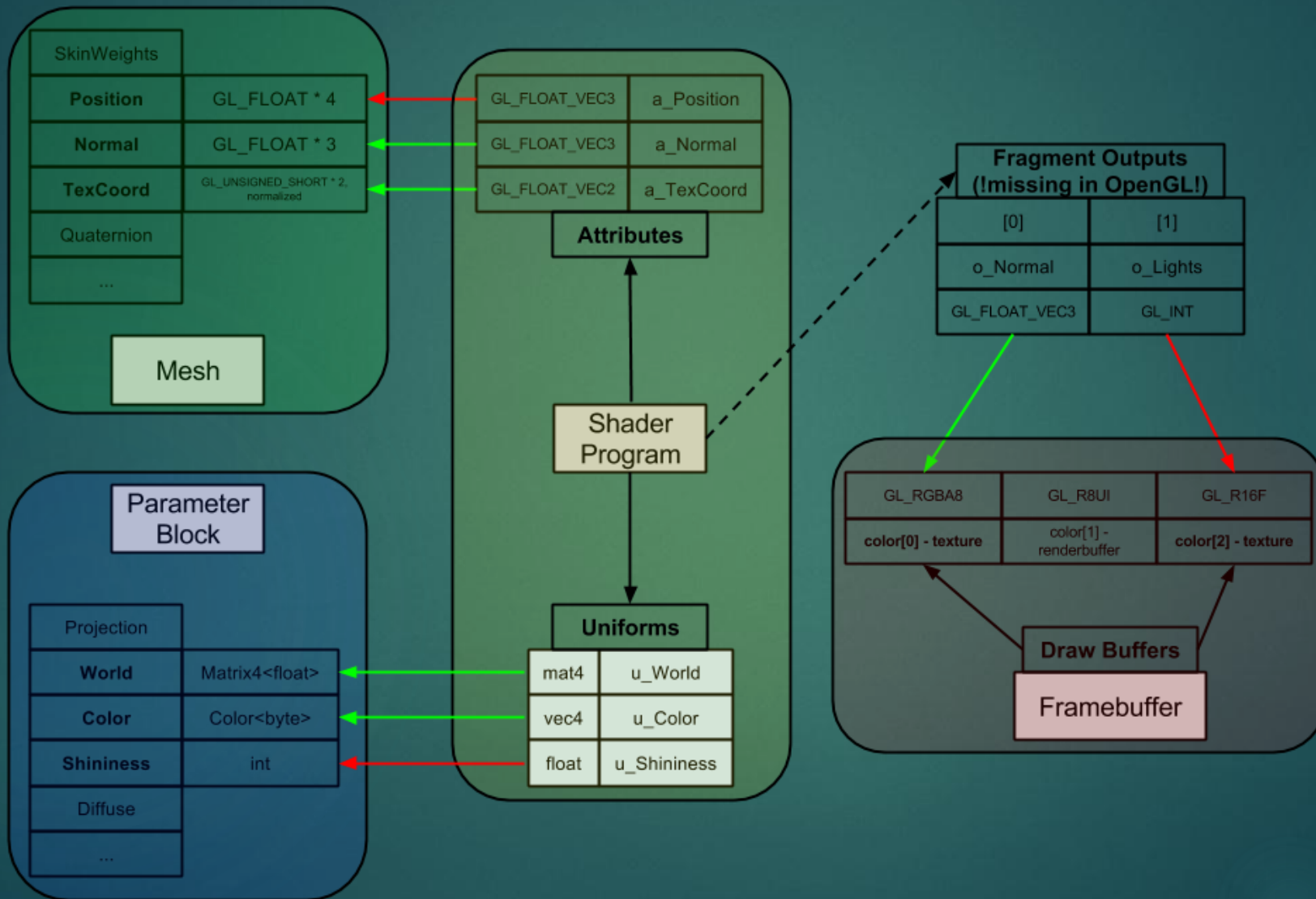


# How?

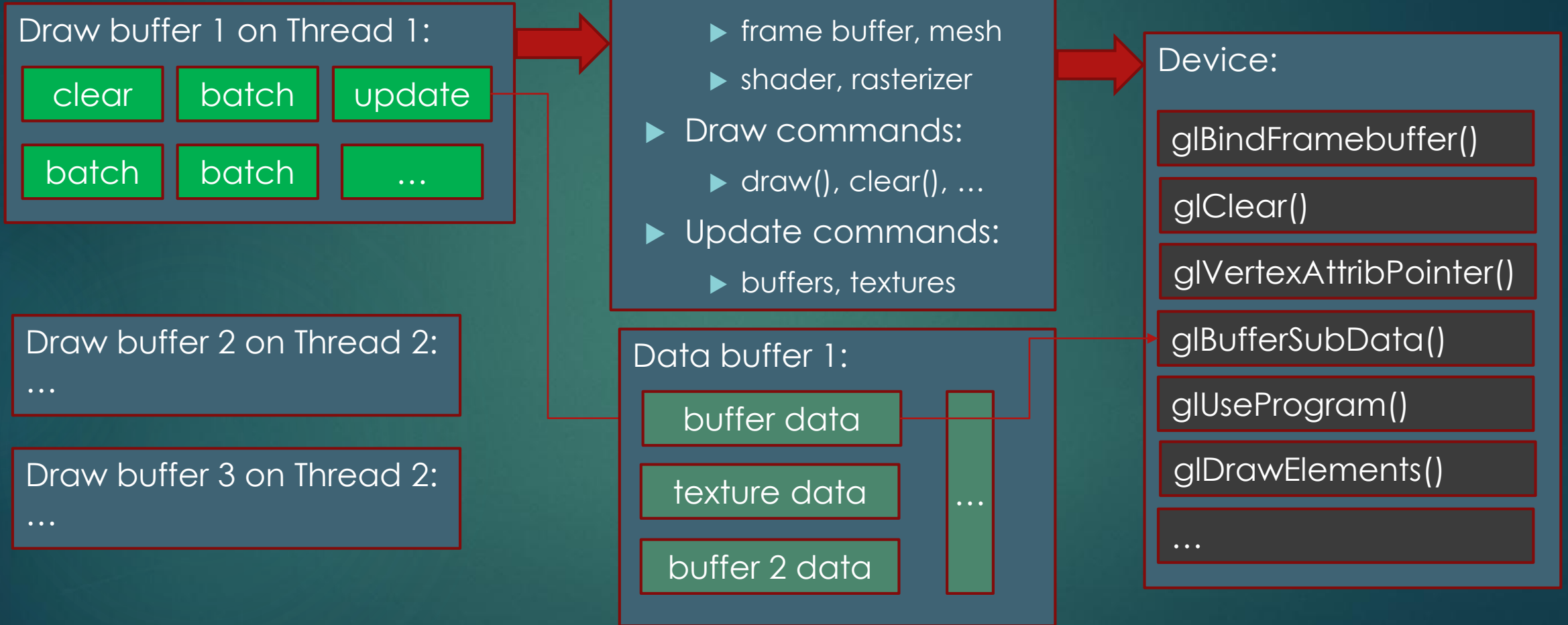
- ▶ Bind-less and state-less\* interface:
  - ▶ `“rederer.draw(&batch, &frame);”`
- ▶ Matching input requirements against the provided data:
  - ▶ Vertex Attributes
  - ▶ Shader Uniforms
  - ▶ Textures and Buffers
- ▶ Draw buffers construction != execution
  - ▶ Multi-threaded rendering
  - ▶ Buffer serialization and reuse
- ▶ State caching



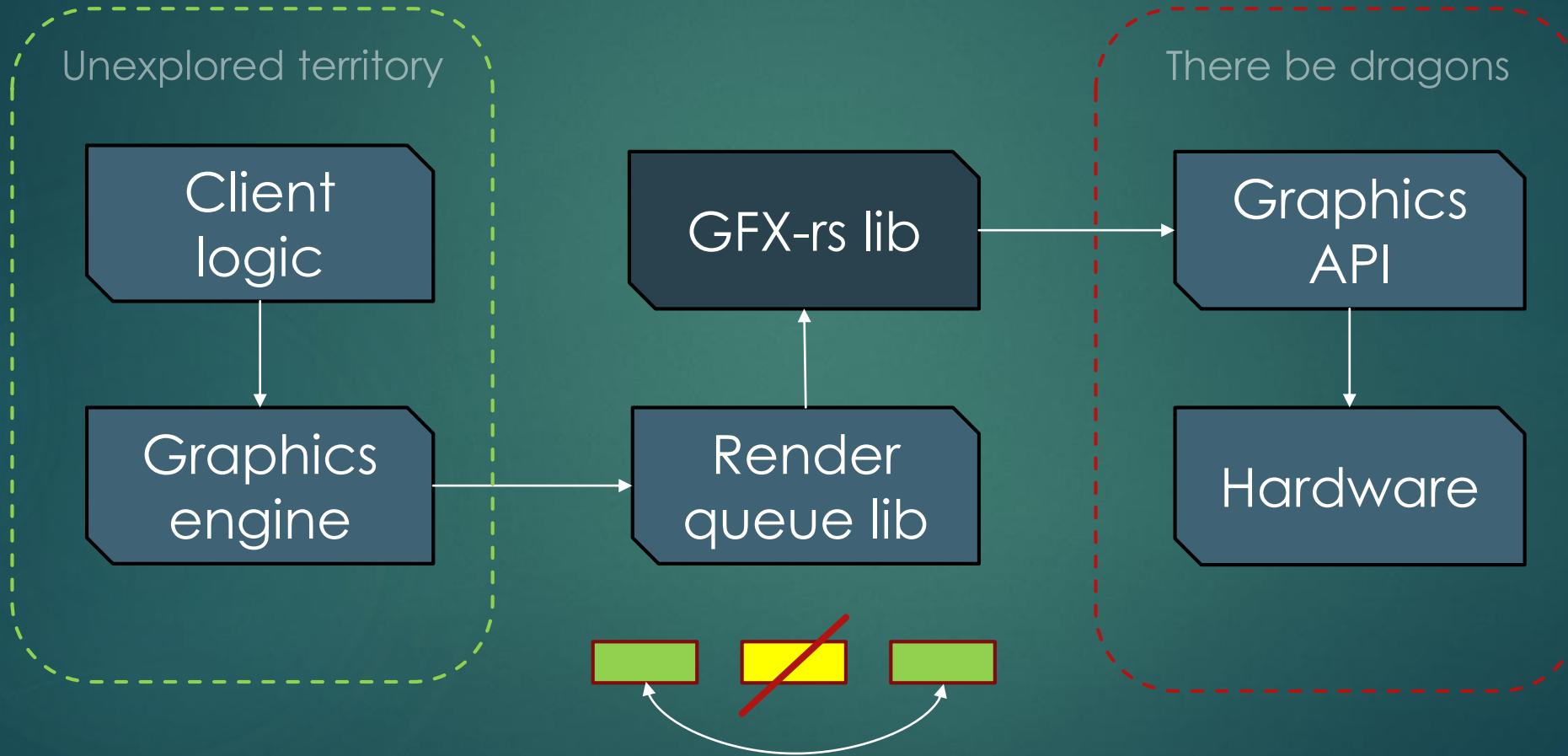
# Draw-call verification



# Draw buffers

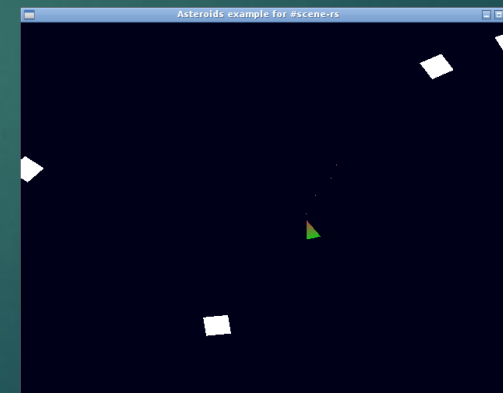
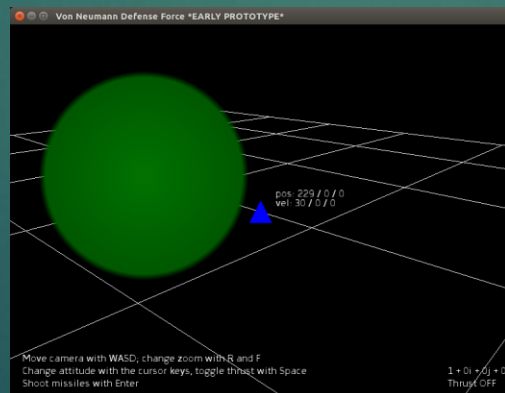
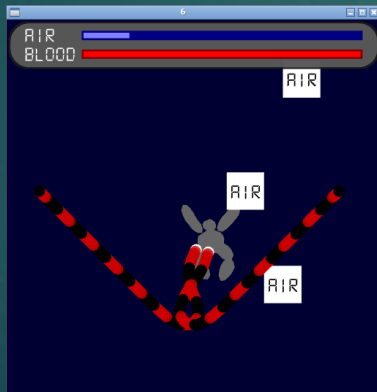


# Application structure in layers



# Existing projects

- ▶ [https://github.com/PistonDevelopers/gfx\\_graphics](https://github.com/PistonDevelopers/gfx_graphics)
  - ▶ <https://github.com/bvssvni/rust-snake>
- ▶ *Von Neumann Defense Force*
  - ▶ Sci-fi MMO by Hanno Braun, <http://www.vndf.de>
- ▶ My experiments in <https://github.com/kvark/scene-rs>
  - ▶ Yet another asteroids game!



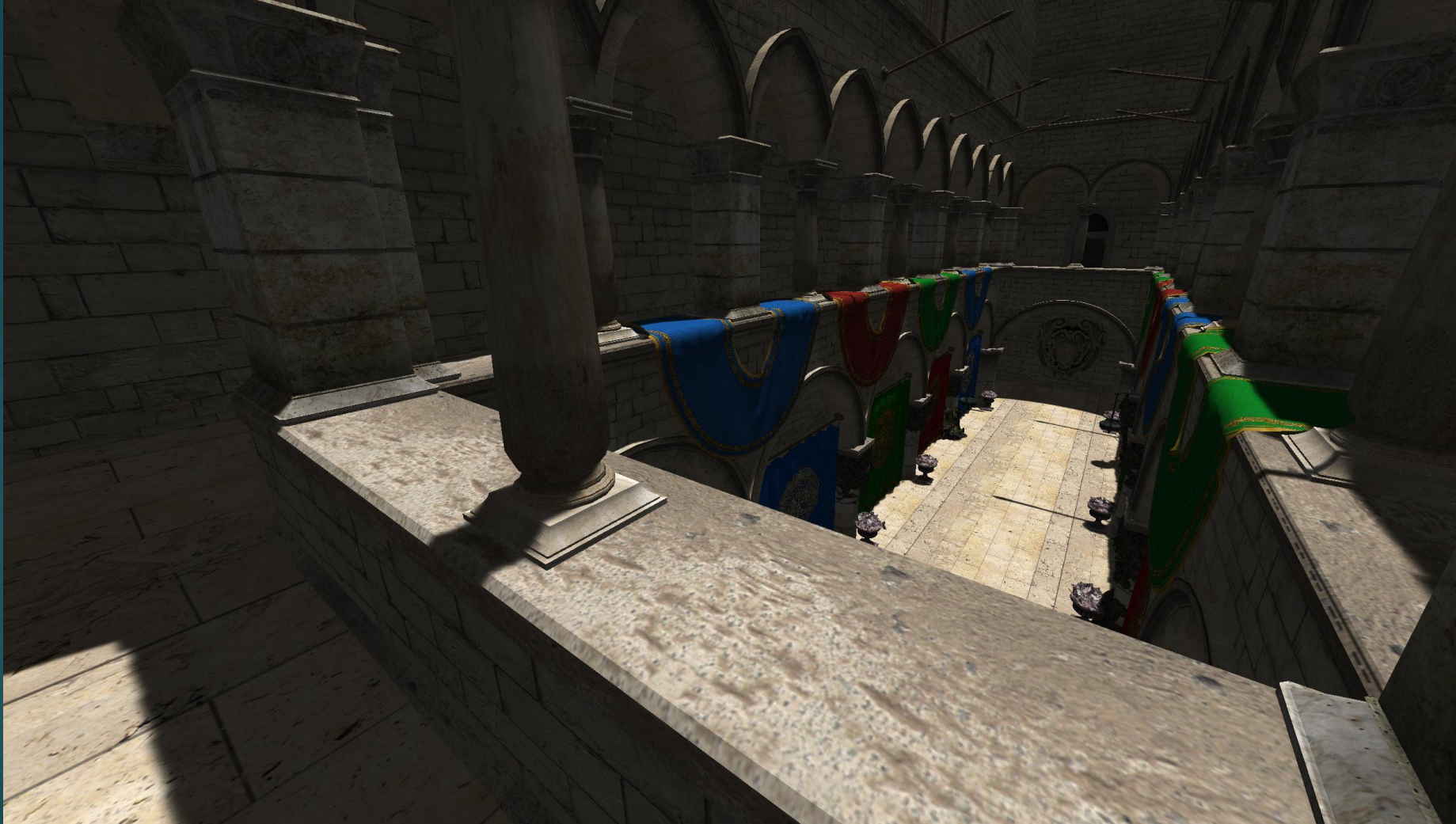


# Hematite





# Snowmew





# Join GFX community

- ▶ Use it, give feedback
- ▶ Say hello at <https://gitter.im/gfx-rs/gfx-rs>
- ▶ Lots of work to do:
  - ▶ Latest and greatest features
  - ▶ More API back-ends
  - ▶ Shader abstraction
  - ▶ Bug fixes and refactoring