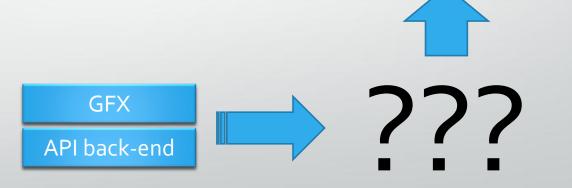
High-level Rendering

for gfx-rs

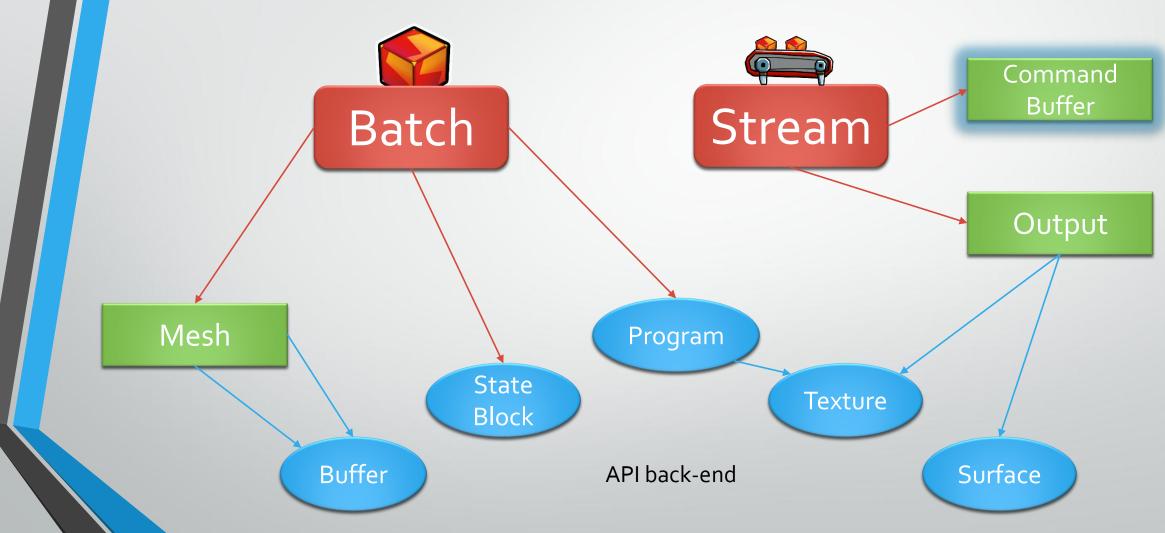
The Plan

- 1. Take Rust
- 2.Build GFX
- 3.???4.Profit!





GFX level



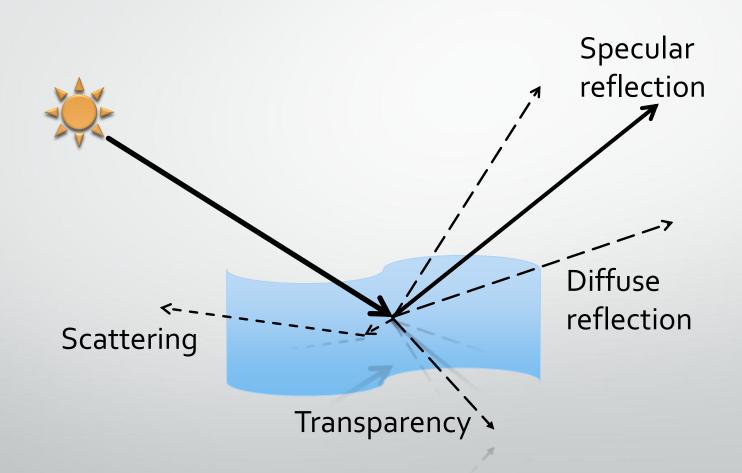
Abstracting the Unknowns

- Light information?
 - Different kinds -
 - Count
- Mesh data?
 - Skinning •
 - Attribute names
- Shader code... </>

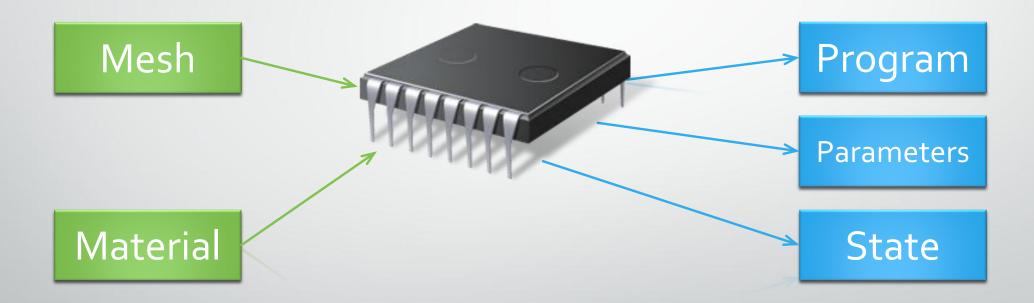
Technique



What is a Material?



Rendering Technique



Phase level







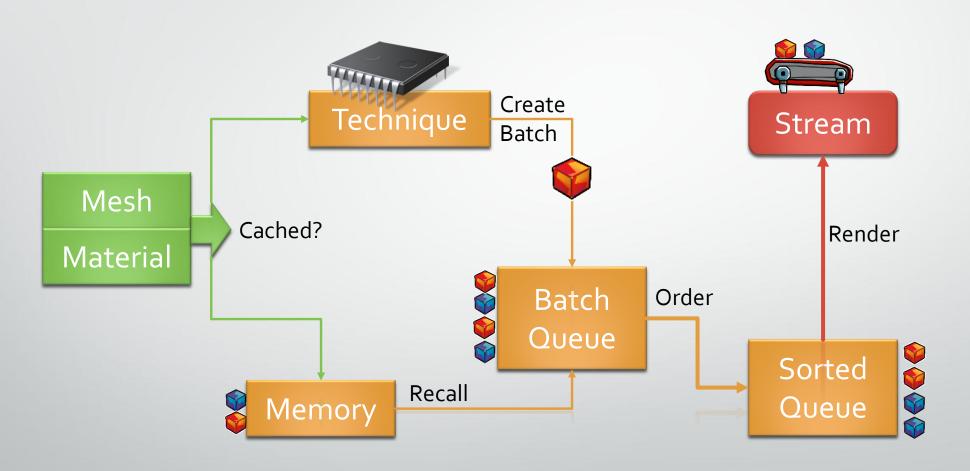






- Everything ends up being a *Batch*, which can be:
 - <u>Cached</u>: avoid batch validation cost
 - Sorted: avoid state switch cost
 - Sent for <u>rendering</u> into a Stream

Render Phase

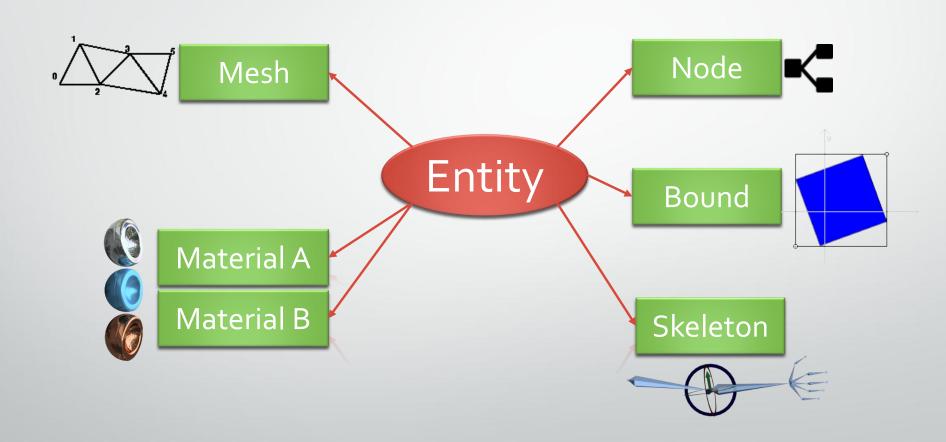


Scene level

- Space
 - Abstract transformations
 - Parent-child relationships
- Scenes, Cameras, Entities
 - Abstract bounds
 - Frustum culling
- Compound objects



Entity



Too much Abstraction?

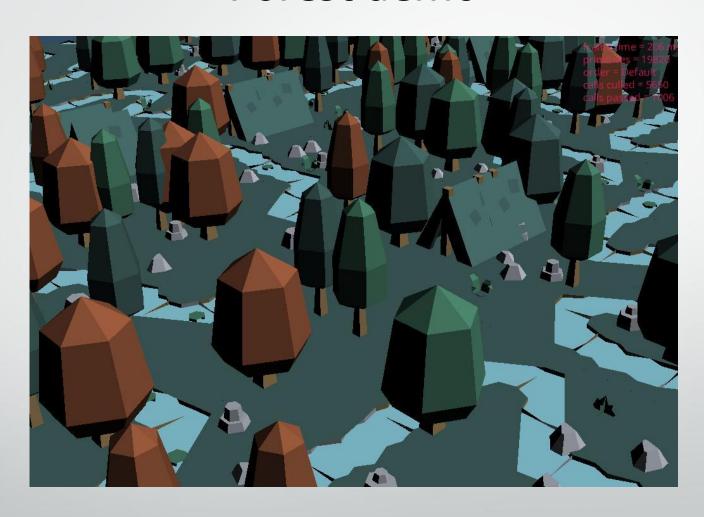
```
pub fn draw<'b, R, M, V, I, H, S>(&mut self, entities: I, phase: &mut H, stream: &mut S)
            -> Result<::Report, ::Error> where
W: 'b,
W::Transform: 'b,
W::NodePtr: 'b,
W::SkeletonPtr: 'b,
B: 'b,
R: gfx::Resources + 'b,
M: 'b,
V:::ViewInfo<W::Scalar, W::Transform>,
I: Iterator<Item = &'b ::Entity<R, M, W, B>>,
 H: gfx_phase::AbstractPhase<R, M, V>,
S: gfx::Stream<R>,
```



GFX Pipeline

- Standard material
 - Physically Based Rendering (*)
- Complete rendering solutions
 - Forward renderer (*)
 - Forward+/Clustered renderer (**)
 - Deferred renderer (**)
- Lots of TODO!

Forest demo



Links

- GFX Scene: https://github.com/kvark/gfx_scene
 - Technique/Phase
 - Entity + Scene
 - persistent draw queue
- GFX Pipeline: https://github.com/kvark/gfx_pipeline
 - standard material
 - basic rendering
- Claymore: https://github.com/kvark/claymore
 - standard scene/world
 - Blender export + import
 - applications