Vulkanised 2025

The 7th Vulkan Developer Conference
Cambridge, UK | February 11-13, 2025

Shipping a game with Vulkan and Rust in 100 days

Kane Rogers-Wong, Studio Head - Adequate Games

tl;dr

tl;dr

Shipped **Windows** and **macOS**

tl;dr

Shipped **Windows** and **macOS**

Most things worked **great**

tl;dr

Shipped **Windows** and **macOS**

Most things worked **great**

Some things **didn’t**

tl;dr

Shipped **Windows** and **macOS**

Most things worked **great**

Some things **didn’t**

You should try Rust **if you want to**

Why we used Rust and Vulkan

Why we used Rust and Vulkan

How we used Rust and Vulkan

Why we used Rust and Vulkan

How we used Rust and Vulkan

What went **well**

Why we used Rust and Vulkan

How we used Rust and Vulkan

What went **well**

What **didn’t**

Context

Hi, I’m Kane!

Indie studio in Melbourne, Australia

Indie studio in Melbourne, Australia

Started in **mobile VR**, perf focussed

Indie studio in Melbourne, Australia

Started in **mobile VR**, perf focussed

We use **Rust** for all the things

Indie studio in Melbourne, Australia

Started in **mobile VR**, perf focussed

We use **Rust** for all the things

Team of 5, **one graphics engineer**

Co-development **pays the bills**

Co-development **pays the bills**

Approached for **graphics expertise**

Co-development **pays the bills**

Approached for **graphics expertise**

Project was in a **death spiral**

Co-development **pays the bills**

Approached for **graphics expertise**

Project was in a **death spiral**

Took over the project

Co-development **pays the bills**

Approached for **graphics expertise**

Project was in a **death spiral**

Took over the project

Rewrote **>99%**

“**Minecraft 1.8 compatible**” client

“Minecraft 1.8 compatible” client

Had some **protocol**, **chunking** code

“Minecraft 1.8 compatible” client

Had some **protocol**, **chunking** code

Everything else from scratch

“Minecraft 1.8 compatible” client

Had some **protocol**, **chunking** code

Everything else from scratch

Needed to ship to beta in **3 months**

“Minecraft 1.8 compatible” client

Had some **protocol**, **chunking** code

Everything else from scratch

Needed to ship to beta in **3 months**

How hard could it be?

Voxel (1x1x1m cube) “terrain”

Voxel (1x1x1m cube) “terrain”

Extremely large (~120,000km) world

Voxel (1x1x1m cube) “terrain”

Extremely large (~120,000km) world

Shared, dynamic multiplayer world

Voxel (1x1x1m cube) “terrain”

Extremely large (~120,000km) world

Shared, dynamic multiplayer world

Custom textures, resources

Voxel (1x1x1m cube) “terrain”

Extremely large (~120,000km) world

Shared, dynamic multiplayer world

Custom textures, resources

Dynamic **voxel lighting**

Voxel (1x1x1m cube) “terrain”

Extremely large (~120,000km) world

Shared, dynamic multiplayer world

Custom textures, resources

Dynamic **voxel lighting**

Plus **entities**, **items**, etc.

Voxel (1x1x1m cube) “terrain”

Extremely la 0km) world

Shared, dyn yer world

Custom text es

Dynamic **vox**

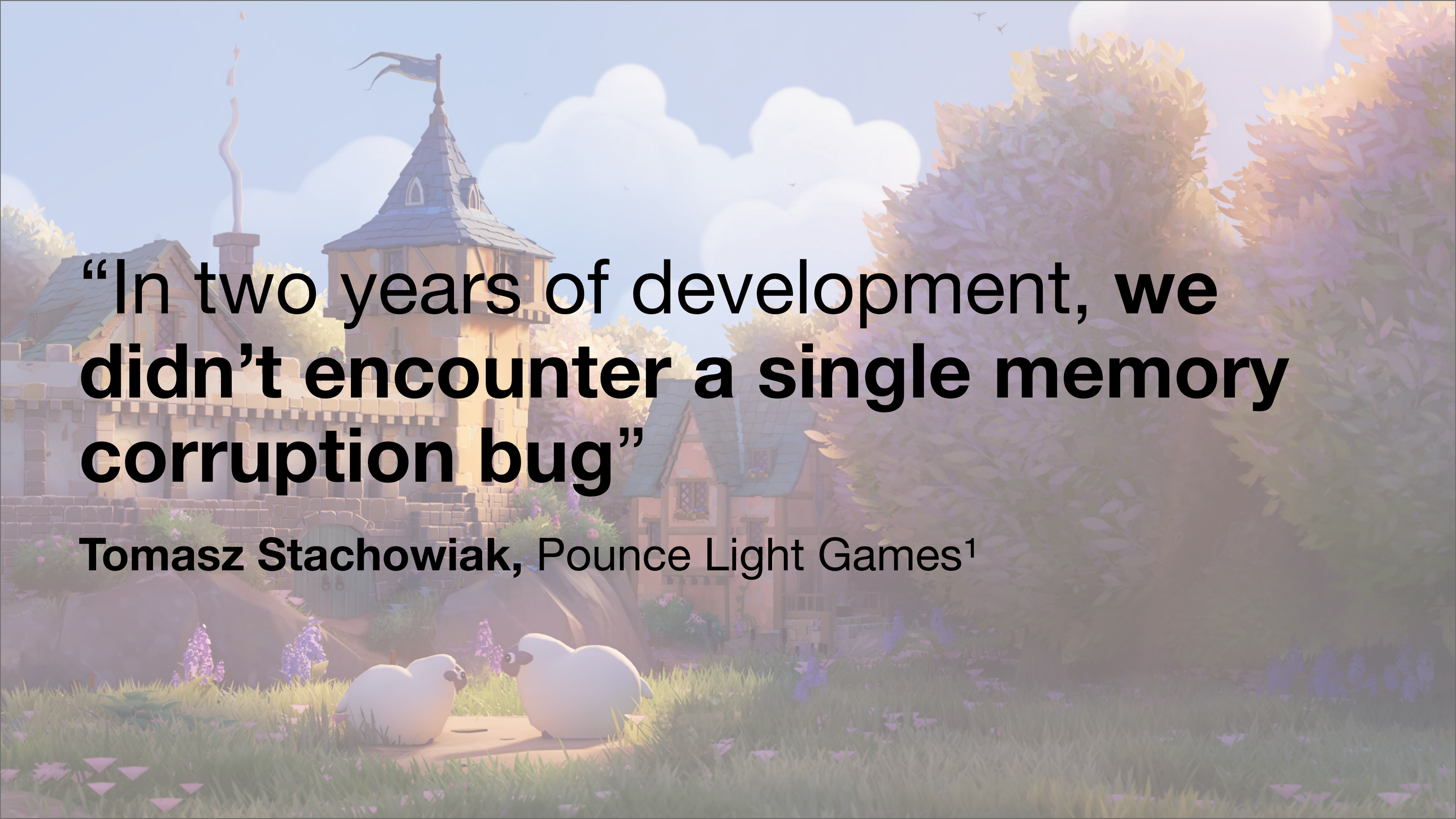
Plus **entities**, **items**, etc.

Why Rust and Vulkan?

Rust


I'll be designated driver tonight!
we know
Don't talk while eating you could choke and die haha


Tiny
Glade



“In two years of development, **we didn’t encounter a single memory corruption bug**”

Tomasz Stachowiak, Pounce Light Games[1]

“If all you do is write Safe Rust, you will **never** have to worry about type-safety or memory-safety.

You will **never** endure a dangling pointer, a use-after-free, or any other kind of Undefined Behaviour.”

The Rust Nomicon

How Rust and Vulkan?

Strong opinion alert

Trying to extend Rust's safety model to the GPU is a *profoundly* terrible idea*

Trying to extend Rust's safety model to the GPU is a *profoundly* terrible idea*

**unless you're WebGPU, in which case, fine*

Rust lets us opt out of its safety guarantees.


unsafe {
}

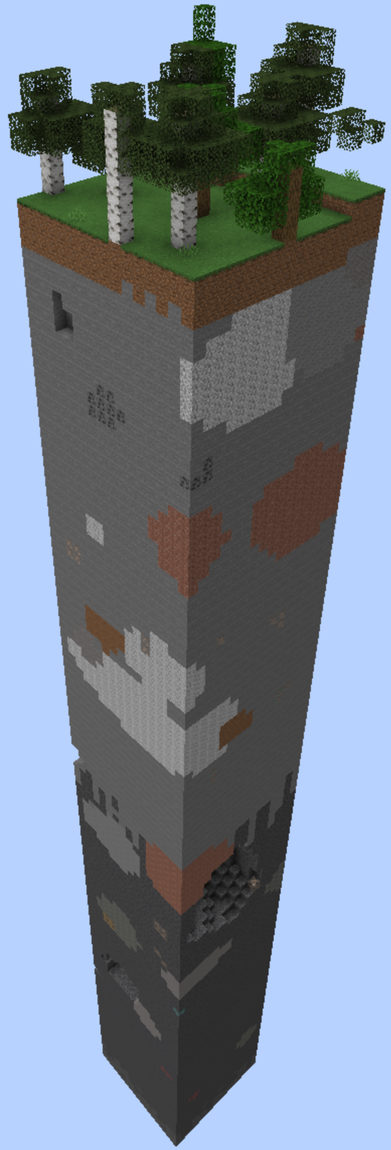

hollup…Let him cook

```rust
unsafe {
    let entry = ash::Entry::load()?;
    let instance = entry.create_instance(
        &vk::InstanceCreateInfo::default().application_info(
            &vk::ApplicationInfo::default()
                .api_version(vk::API_VERSION_1_3)
                .application_name(c"Hello Vulkanised"),
        ),
        None,
    )?;
}
```


ME
LUNARG
VALIDATION LAYERS
imgflip.com

Challenge #1

Render some cubes



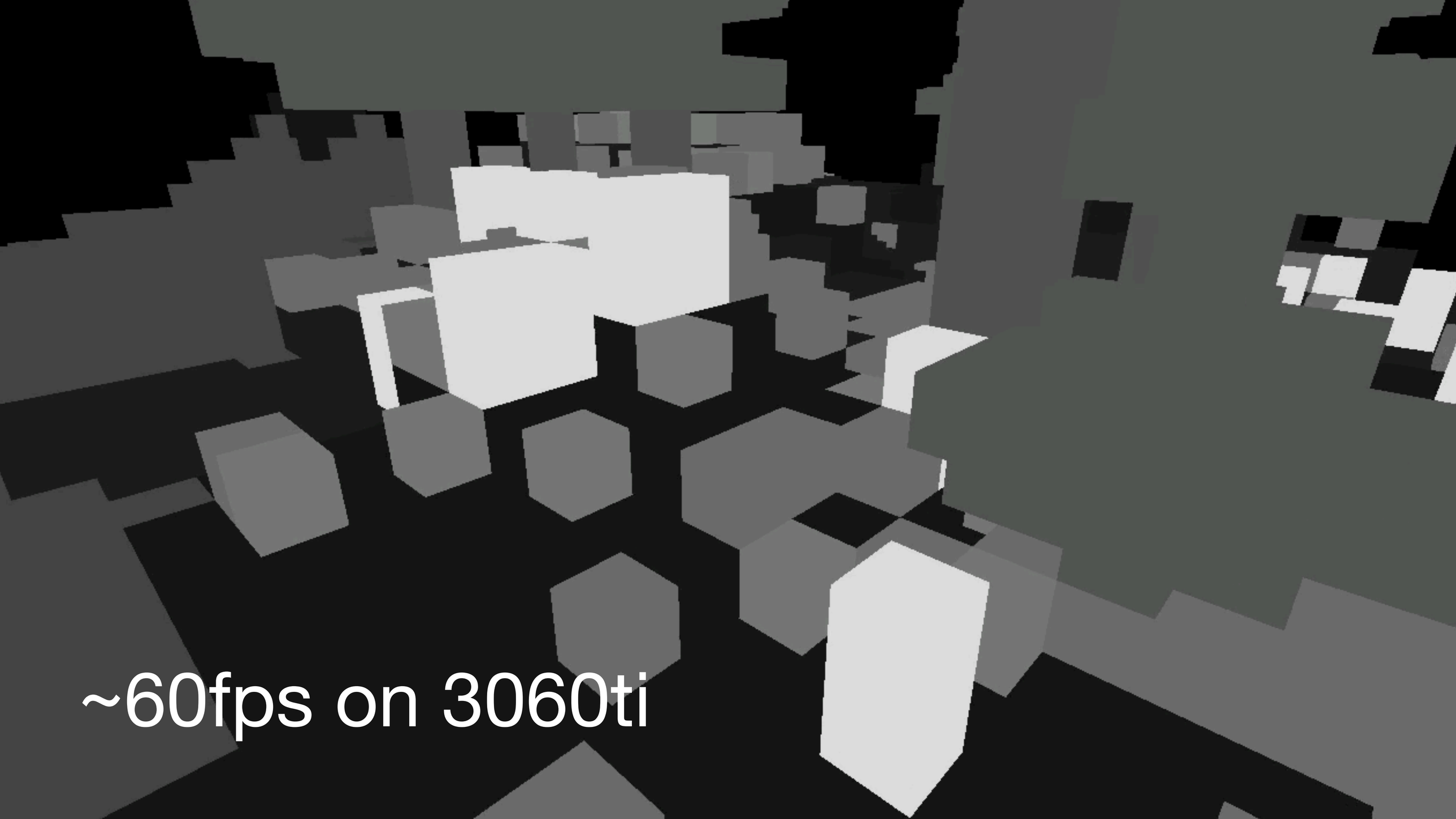
Chunk

16x256x16

```
for chunk in chunks {
    for block in chunk.blocks {
        draw_cube(block);
    }
}
```

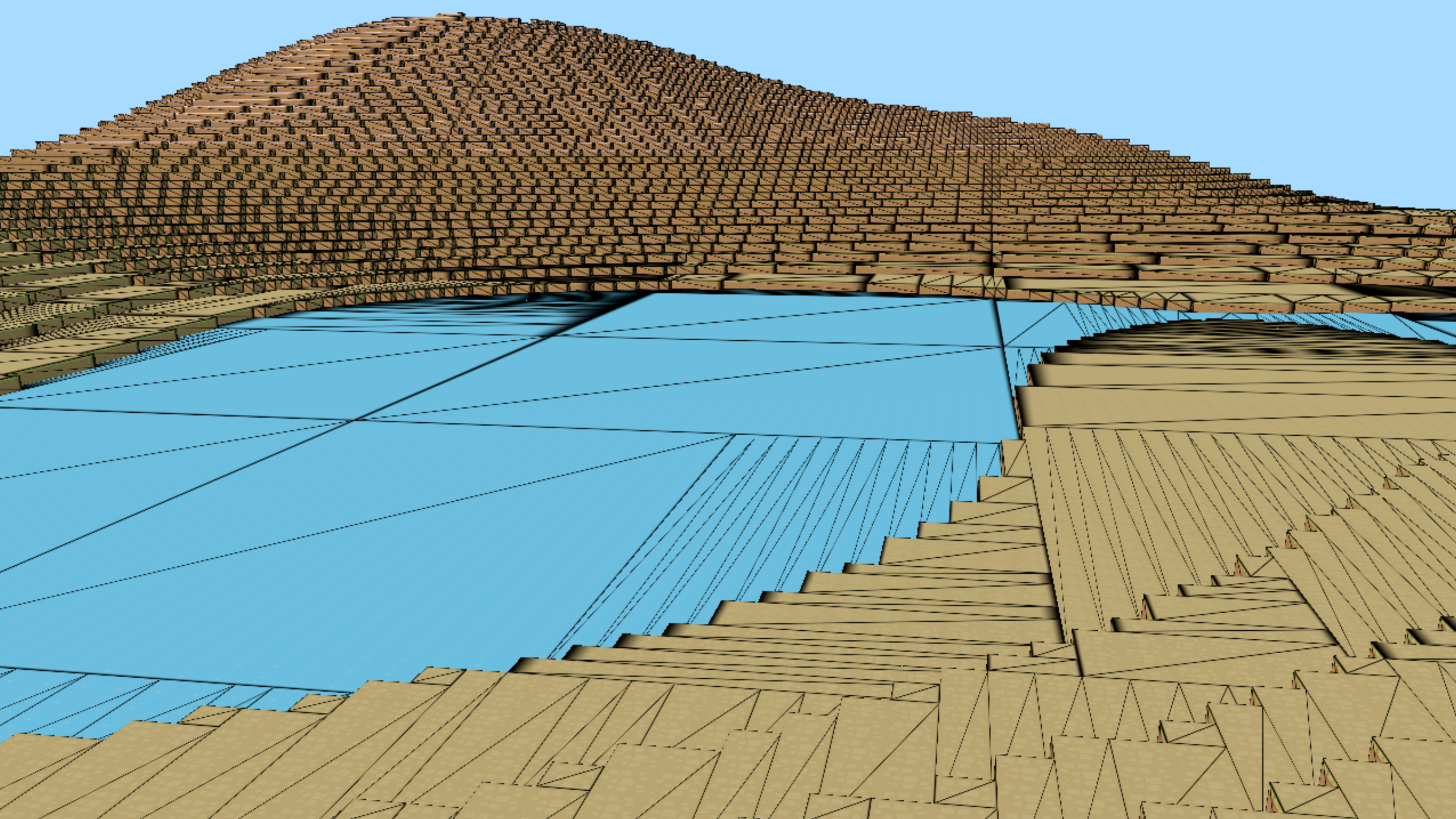
```
for chunk in chunks {
    for block in chunk.blocks {
        draw_cube(block);
    }
}
```

~1,840,800 draw calls *per frame*


~60fps on 3060ti

Challenge #2

Render some cubes in a *non-insane* way




AIN'T NOBODY GOT
TIME FOR THAT
memecenter.com
MemeCenter

Build a buffer of “Faces” **per chunk**

Build a buffer of “Faces” **per chunk**

vkCmdDraw(…, **face_count * 6**, ..)

Build a buffer of “Faces” **per chunk**

vkCmdDraw(…, **face_count * 6**, ..)

face_id = **gl_VertexIndex / 6**

Build a buffer of “Faces” **per chunk**

vkCmdDraw(…, **face_count * 6**, ..)

face_id = **gl_VertexIndex / 6**

Face face = faces_buffer[**face_id**]

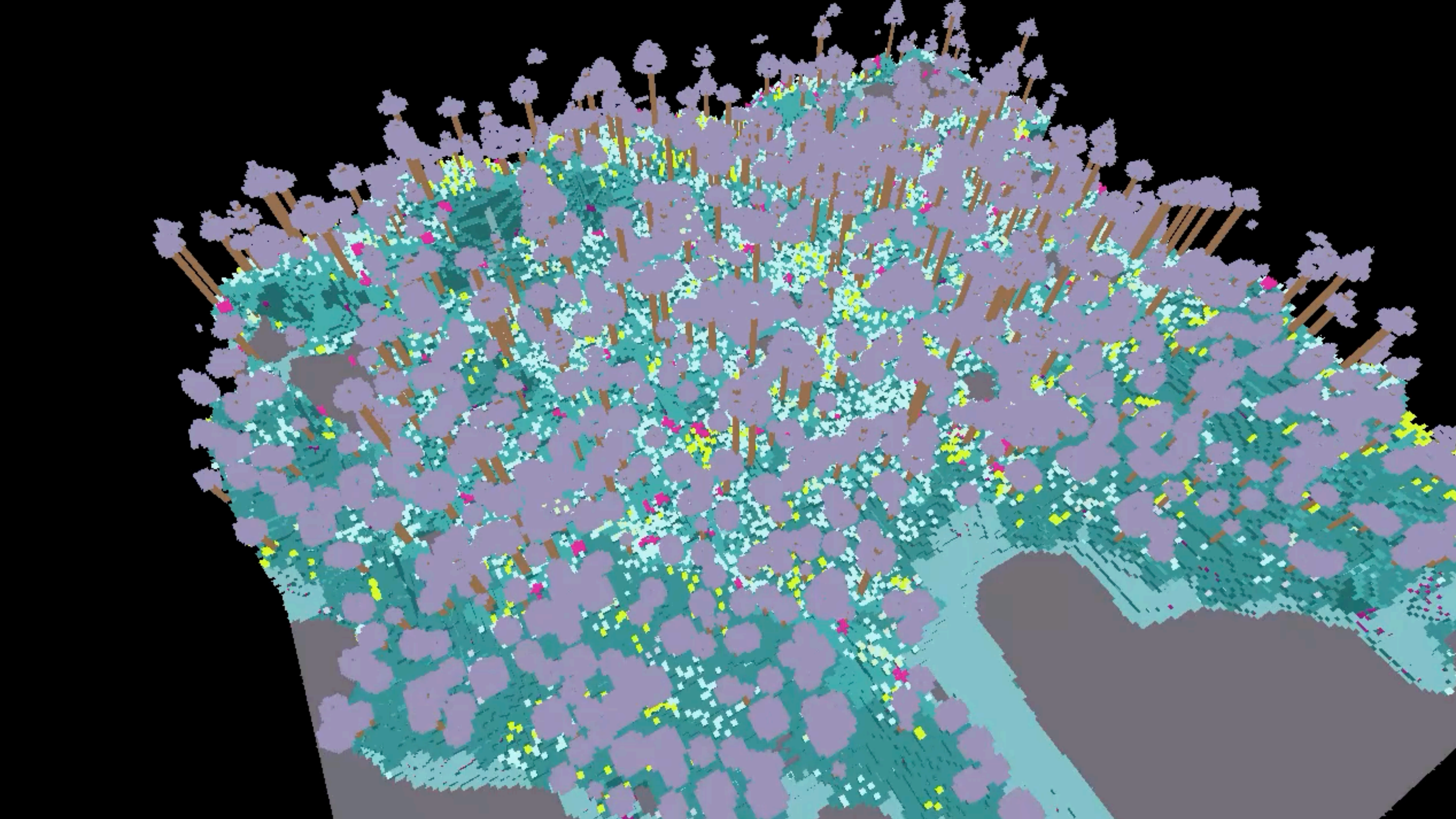
Build a buffer of “Faces” **per chunk**

vkCmdDraw(…, **face_count * 6**, ..)

face_id = **gl_VertexIndex / 6**

Face face = faces_buffer[**face_id**]

vec3 pos = quad_vertices[**gl_VertexIndex % 6**]



Aside

Minecraft “model” format

```
{
    "elements": [
        {   "from": [ 0, 0, 0 ],
            "to": [ 16, 16, 16 ],
            "faces": {
                "down":  { "texture": "#down", "cullface": "down" },
                "up":    { "texture": "#up", "cullface": "up" },
                "north": { "texture": "#north", "cullface": "north" },
                "south": { "texture": "#south", "cullface": "south" },
                "west":  { "texture": "#west", "cullface": "west" },
                "east":  { "texture": "#east", "cullface": "east" }
            }
        }
    ]
}
```

```
        "wood": "blocks/planks_oak",
        "tripwire": "blocks/trip_wire"
    },
    "elements": [
        {   "from": [ 7.75, 2.5, 0 ],
            "to": [ 8.25, 2.5, 6.7 ],
            "rotation": { "origin": [ 8, 0, 0 ], "axis": "x", "angle": -22.5, "rescale": true },
            "faces": {
                "down":  { "uv": [ 0, 8, 16, 6 ], "texture": "#tripwire", "rotation": 90 },
                "up":    { "uv": [ 0, 6, 16, 8 ], "texture": "#tripwire", "rotation": 90 }
            }
        },
        {   "from": [ 6.2, 4.2, 6.7 ],
            "to": [ 9.8, 5, 10.3 ],
            "faces": {
                "down":  { "uv": [ 5, 3, 11, 9 ], "texture": "#hook" },
                "up":    { "uv": [ 5, 3, 11, 9 ], "texture": "#hook" },
                "north": { "uv": [ 5, 3, 11, 4 ], "texture": "#hook" },
                "south": { "uv": [ 5, 8, 11, 9 ], "texture": "#hook" },
                "west":  { "uv": [ 5, 8, 11, 9 ], "texture": "#hook" },
                "east":  { "uv": [ 5, 3, 11, 4 ], "texture": "#hook" }
            }
        },
        {   "from": [ 7.4, 4.2, 9.1 ],
            "to": [ 8.6, 5, 9.1 ],
            "faces": {
                "north": { "uv": [ 7, 8, 9, 9 ], "texture": "#hook" }
            }
        },
        {   "from": [ 7.4, 4.2, 7.9 ],
            "to": [ 8.6, 5, 7.9 ],
            "faces": {
                "south": { "uv": [ 7, 3, 9, 4 ], "texture": "#hook" }
            }
        },
        {   "from": [ 7.4, 4.2, 7.9 ],
            "to": [ 7.4, 5, 9.1 ],
            "faces": {
                "east": { "uv": [ 7, 8, 9, 9 ], "texture": "#hook" }
            }
        },
        {   "from": [ 8.6, 4.2, 7.9 ],
            "to": [ 8.6, 5, 9.1 ],
            "faces": {
                "west": { "uv": [ 7, 3, 9, 4 ], "texture": "#hook" }
            }
        },
        {   "from": [ 7.4, 5.2, 10 ],
            "to": [ 8.8, 6.8, 14 ],
            "rotation": { "origin": [ 8, 6, 14 ], "axis": "x", "angle": -22.5 },
            "faces": {
```

There are 441 chunks in the world

293 40 351 351 351 425

Challenge #3

Get it running elsewhere

Knew we had **very limited time**

Knew we had **very limited time**

Zero ability to deal with old/integrated cards

Knew we had **very limited time**

Zero ability to deal with old/integrated cards

Needed to use **modern features**

Knew we had **very limited time**

Zero ability to deal with old/integrated cards

Needed to use **modern features**

Windows minspec: **GTX 1060**

Knew we had **very limited time**

Zero ability to deal with old/integrated cards

Needed to use **modern features**

Windows minspec: **GTX 1060**

Mac minspec: **M1 Pro**

Bindless with **Buffer Device Address**

Bindless with **Buffer Device Address**

Single “**All the Textures**” descriptor set

Bindless with **Buffer Device Address**

Single “**All the Textures**” descriptor set

Worried about MoltenVK support, played it safe

Bindless with **Buffer Device Address**

Single “**All the Textures**” descriptor set

Worried about MoltenVK support, played it safe

Didn’t use **dynamic rendering**

Bindless with **Buffer Device Address**

Single “**All the Textures**” descriptor set

Worried about MoltenVK support, played it safe

Didn’t use **dynamic rendering**

Didn’t use **sync2**

Bindless with **Buffer Device Address**

Single “**All the Textures**” descriptor set

Worried about MoltenVK support, played it safe

Didn’t use **dynamic rendering**

Didn’t use **sync2**

Overly conservative, big mistake

Challenge #4

Render the other things

Entities are ~anything that’s **not a block**

Entities are ~anything that’s **not a block**

Can move, animate, hold other entities

Entities are ~anything that’s **not a block**

Can move, animate, hold other entities

Used **glTF** as our model format

Entities are ~anything that’s **not a block**

Can move, animate, hold other entities

Used **glTF** as our model format

Single **vertex/index buffer** + “All the Textures”

Entities are ~anything that’s **not a block**

Can move, animate, hold other entities

Used **glTF** as our model format

Single **vertex/index buffer** + “All the Textures”

Store **offsets per model** + **texture IDs**

Entities are ~anything that’s **not a block**

Can move, animate, hold other entities

Used **glTF** as our model format

Single **vertex/index buffer** + “All the Textures”

Store **offsets per model** + **texture IDs**

Worked **incredibly well**

Light is important for visuals + gameplay

Light is important for visuals + gameplay

Lights can be **placed/removed arbitrarily**

Light is important for visuals + gameplay

Lights can be **placed/removed arbitrarily**

Caves, tunnels, pits, etc affect sunlight

Light is important for visuals + gameplay

Lights can be **placed/removed arbitrarily**

Caves, tunnels, pits, etc affect sunlight

Flood fill algorithm

Light is important for visuals + gameplay

Lights can be **placed/removed arbitrarily**

Caves, tunnels, pits, etc affect sunlight

Flood fill algorithm

Store **light value per block** in a buffer

Light is important for visuals + gameplay

Lights can be **placed/removed arbitrarily**

Caves, tunnels, pits, etc affect sunlight

Flood fill algorithm

Store **light value per block** in a buffer

Update on chunk modification

Light is important for visuals + gameplay

Lights can be **placed/removed arbitrarily**

Caves, tunnels, pits, etc affect sunlight

Flood fill algorithm

Store **light value per block** in a buffer

Update on chunk modification

Fetch light value **in shader**


Hytopia Closed Beta v0.0.7
2
5
16
2
64
64

Challenge #5

Fix all the **crimes**

Sync quickly emerged as a major issue

Sync quickly emerged as a major issue

Any resource can be updated at **any time**

Sync quickly emerged as a major issue

Any resource can be updated at **any time**

Server can send new **textures**, **models**, **blocks**

Sync quickly emerged as a major issue

Any resource can be updated at **any time**

Server can send new **textures**, **models**, **blocks**

Focussed on **correctness** first


VKDEVICEWAITIDLE
LIGHTYEAR
VKDEVICEWAITIDLE EVERYWHERE
imgflip.com

Big, dumb sledgehammer

Big, dumb sledgehammer

Used in **all** buffer and image operations

Big, dumb sledgehammer

Used in **all** buffer and image operations

Render Passes were a **mess**

Big, dumb sledgehammer

Used in **all** buffer and image operations

Render Passes were a **mess**

Predictably **terrible performance**

Big, dumb sledgehammer

Used in **all** buffer and image operations

Render Passes were a **mess**

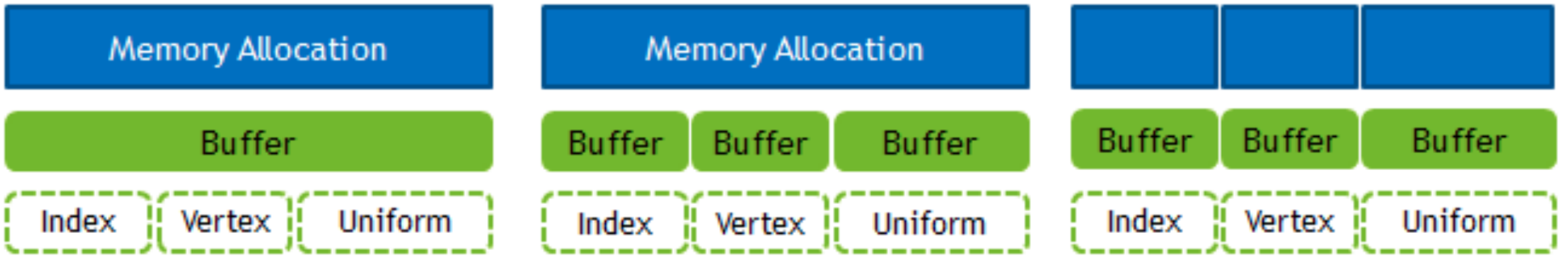
Predictably **terrible performance**

Don’t try this at home

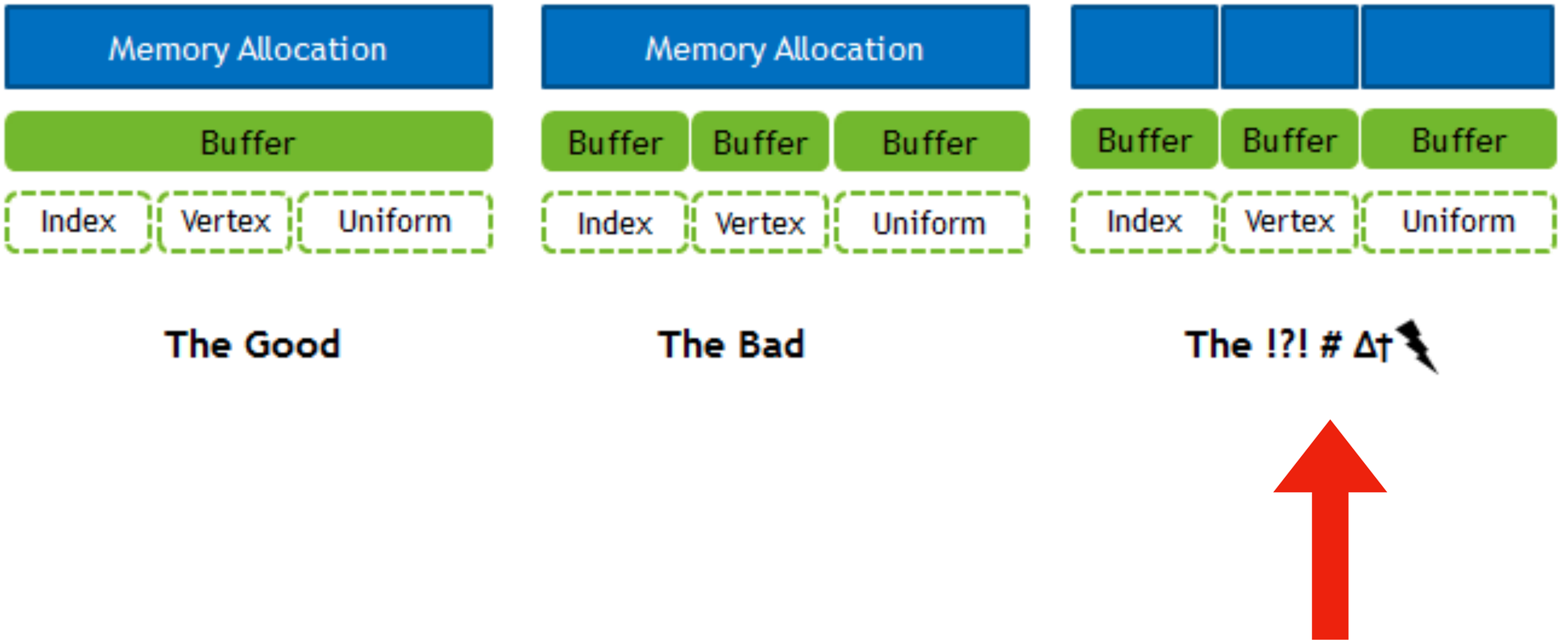

BUT WAIT
Baking Soda
THERE'S MORE
quickmeme.com

CRIMES

Vulkan. NVIDIA.



The Good **The Bad** **The !?! # Δ†**


Vulkan.
NVIDIA.
Memory Allocation
Buffer
Index
Vertex
Uniform
Memory Allocation
Buffer
Buffer
Buffer
Index
Vertex
Uniform
Buffer
Buffer
Buffer
Index
Vertex
Uniform
The Good
The Bad
The !?! # Δ†

The worst possible buffer management

The worst possible buffer management

Allocating/freeing ~each frame

The worst possible buffer management

Allocating/freeing ~each frame

Started as hitches, ended up as **hangs**

The worst possible buffer management

Allocating/freeing ~each frame

Started as hitches, ended up as **hangs**

Still working on features with **one week to ship**

The worst possible buffer management

Allocating/freeing ~each frame

Started as hitches, ended up as **hangs**

Still working on features with **one week to ship**

Needed to burn it all down and start again



Removed all usage of **vkDeviceWaitIdle**

Removed all usage of **vkDeviceWaitIdle**

Switch to **dynamic rendering**

Removed all usage of **vkDeviceWaitIdle**

Switch to **dynamic rendering**

Ran **syncval**

Removed all usage of **vkDeviceWaitIdle**

Switch to **dynamic rendering**

Ran **syncval**

Did the **right things**

Removed all usage of **vkDeviceWaitIdle**

Switch to **dynamic rendering**

Ran **syncval**

Did the **right things**

Everything worked out **great**!

Allocated **DEVICE_LOCAL** memory on init

Allocated **DEVICE_LOCAL** memory on init

Sub-allocated with offsets

Allocated **DEVICE_LOCAL** memory on init

Sub-allocated with offsets

https://github.com/sebbbi/OffsetAllocator

Allocated **DEVICE_LOCAL** memory on init

Sub-allocated with offsets

https://github.com/sebbbi/OffsetAllocator

Scratch buffer for transfers

Allocated **DEVICE_LOCAL** memory on init

Sub-allocated with offsets

https://github.com/sebbbi/OffsetAllocator

Scratch buffer for transfers

Get base address + offset

Allocated **DEVICE_LOCAL** memory on init

Sub-allocated with offsets

https://github.com/sebbbi/OffsetAllocator

Scratch buffer for transfers

Get base address + offset

Everything worked out **great**!

We shipped it!

The good:

Rust let us move **quickly**

Simple, bindless renderer **allowed high reuse**

Buffer Device Address is *the cat's pyjamas*

Vulkan gave us the confidence to come back and fix things when we needed to

MoltenVK is great! Try it!

Eternal gratitude to:

RenderDoc / Baldur

Validation layers, DEBUG_UTILS, vkConfig, Shader Assisted Debugging, Syncval

Allowed **non-graphics** programmers to submit useful, actionable graphics bugs!

Vulkan docs, **spec**, sync examples

TU Wien - **amazing** explanation of sync

The bad:

Should have used more modern features earlier

Overly conservative about MoltenVK

Should have thought more about data flow

Summary

Shipped **Windows** and **macOS**

Most things worked **great**

Some things **didn’t**

You should try Rust **if you want to**

Thanks!

References:

[1] Stachowiak, Tomasz. (2024). Rendering Tiny Glades With Entirely Too Much Ray Marching. Presented at the Graphics Programming Conference 2024.

https://www.youtube.com/watch?v=jusWW2pPnA0