



TRAVERSE RESEARCH

Volumetric rendering using rust

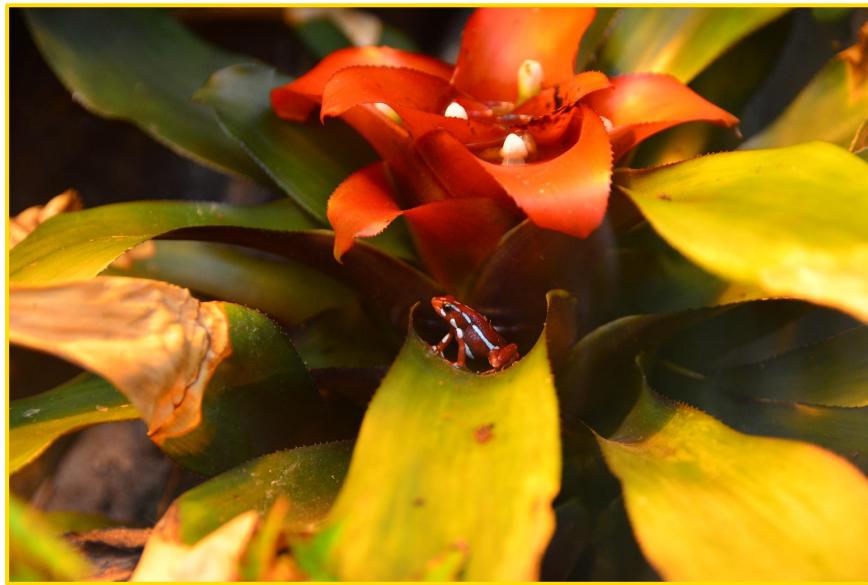
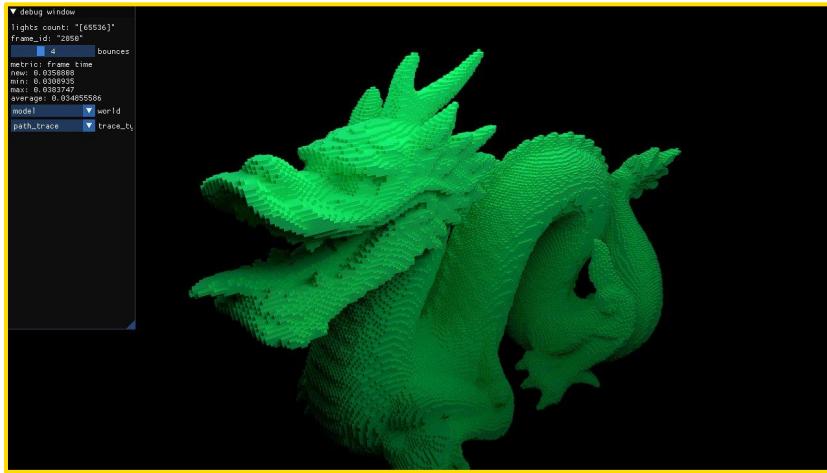
NEXT SLIDE



ABOUT ME:

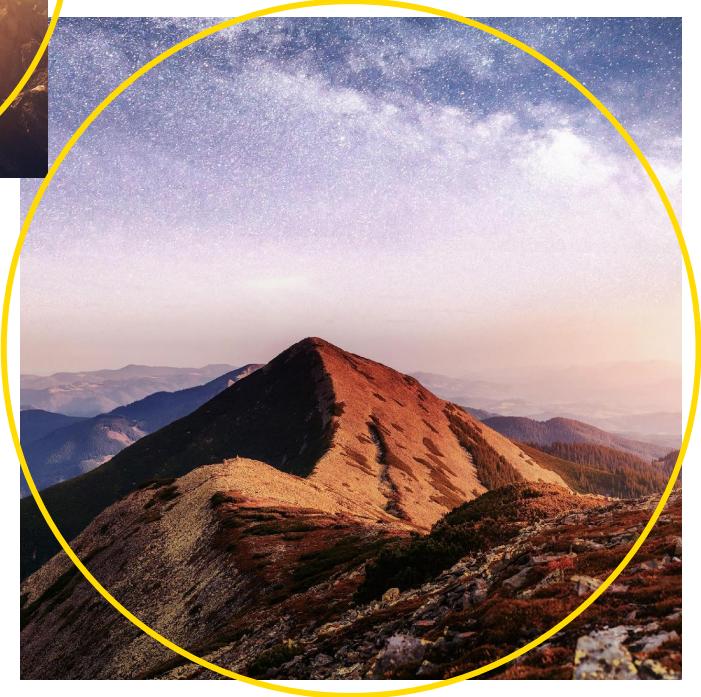
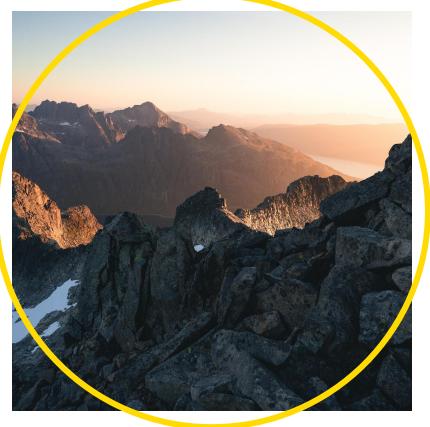
Rosalie de Winther

- Skating
- Volumetric rendering
- Programming languages
- Photography



TODAY

- 1 What do we want?
- 2 How is it done?
- 3 How to improve?
- 4 VDB
- 5 Breda
- 6 Active research



VOLUMETRIC?

- Things that have volume
- Triangle meshes used for opaque surfaces

VOLUMETRIC?

- Clouds



VOLUMETRIC?

- Clouds
- Dust/smoke



VOLUMETRIC?

- Clouds
- Dust/smoke
- Explosions





TRAVERSE RESEARCH

BUT HOW?

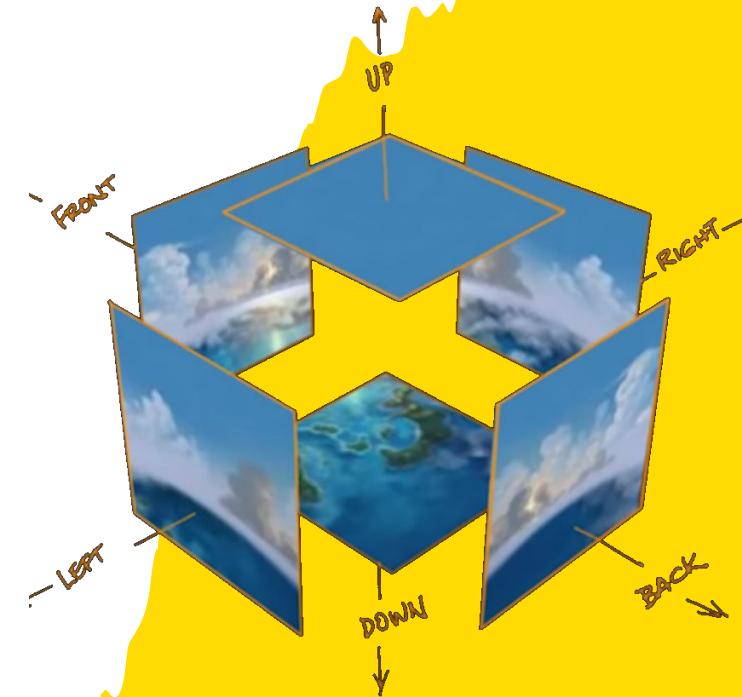


TRAVERSE RESEARCH

HOW

Cube map:

- Static
- Only for sky



HOW

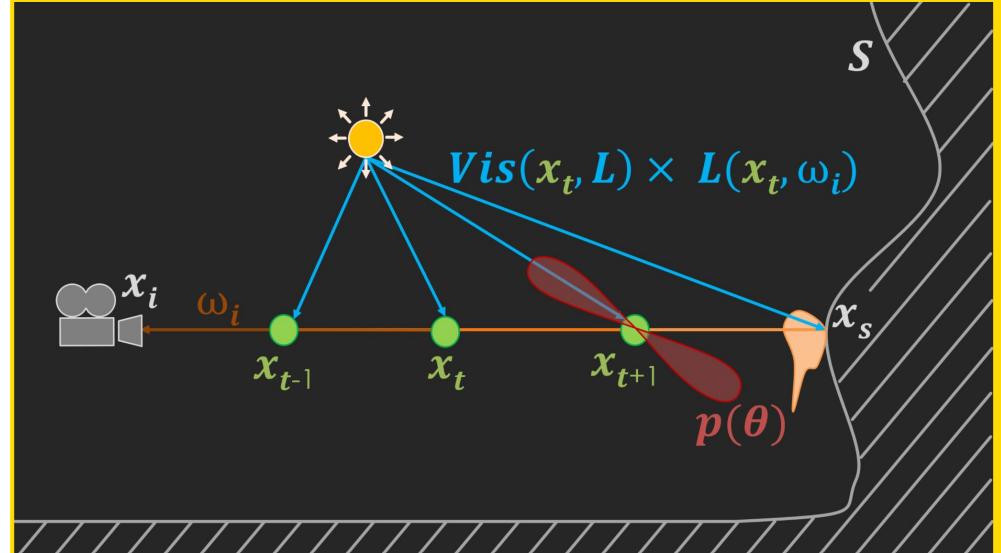
Texture + deformation



HOW

Analytical ray marching:

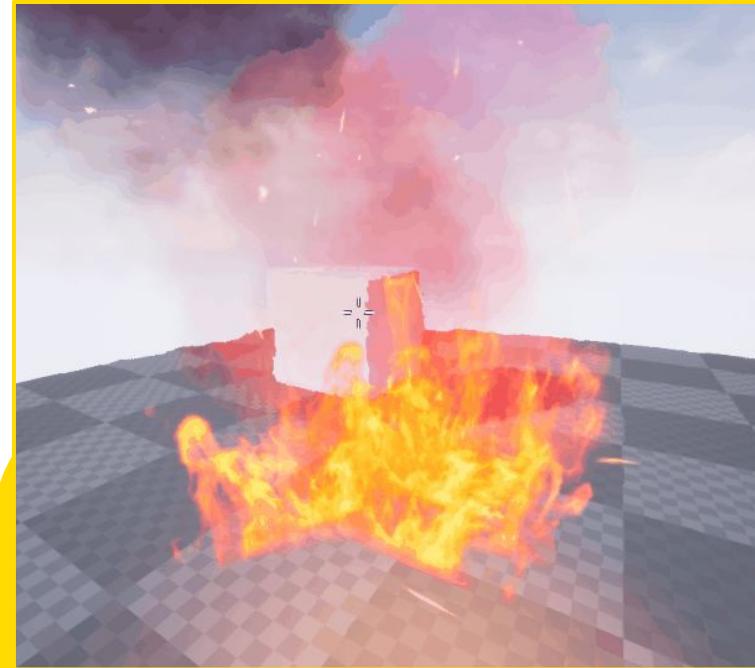
- 2.5D
- Use noise textures
- Used by modern engines currently



HOW

Smoke effect textures:

- Particle based
- Pre-rendered
- 2d textures

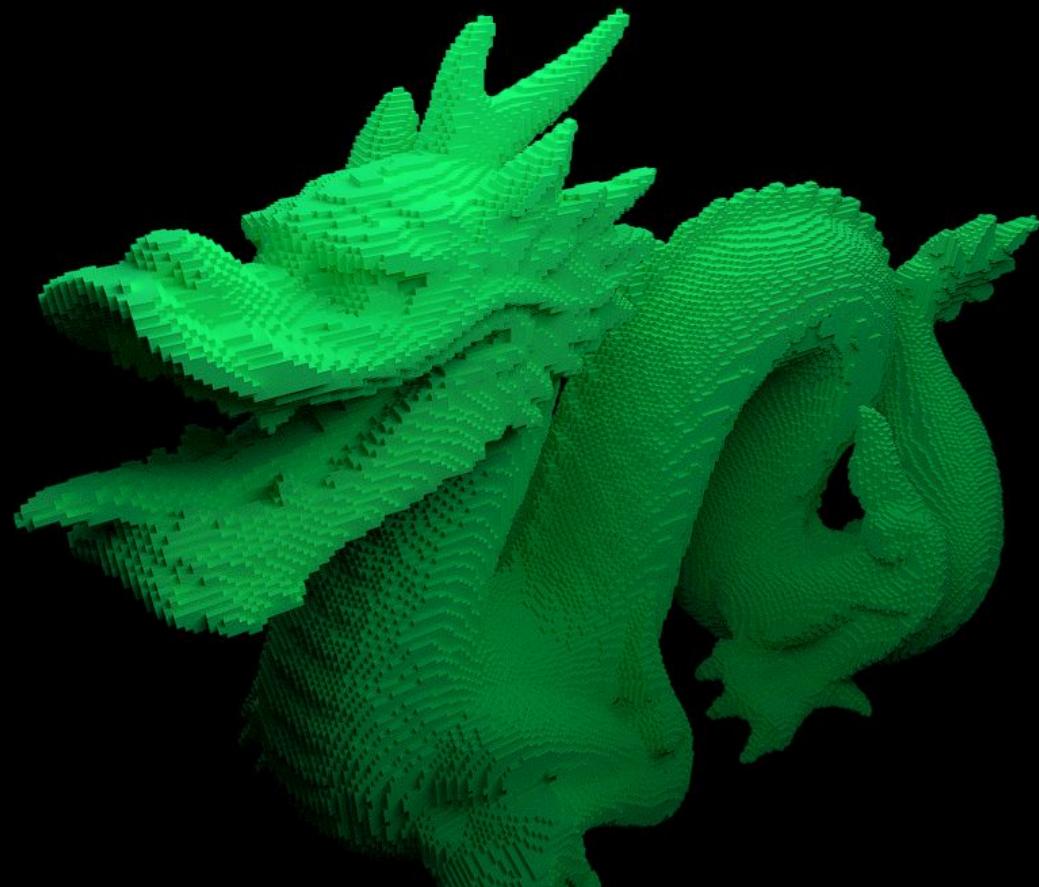


What now...



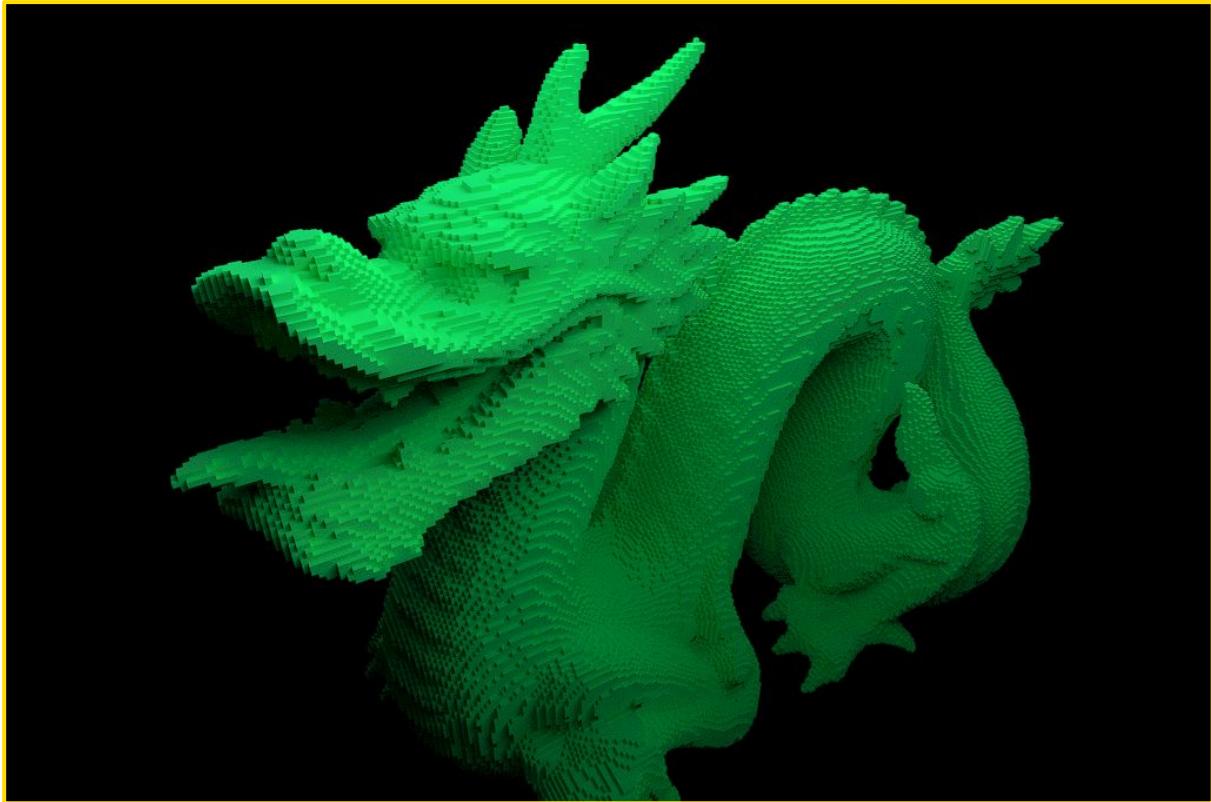
TRAVERSE RESEARCH

What now...



VOXELS

- 3D data representation
- Perfect for volume data
- But...



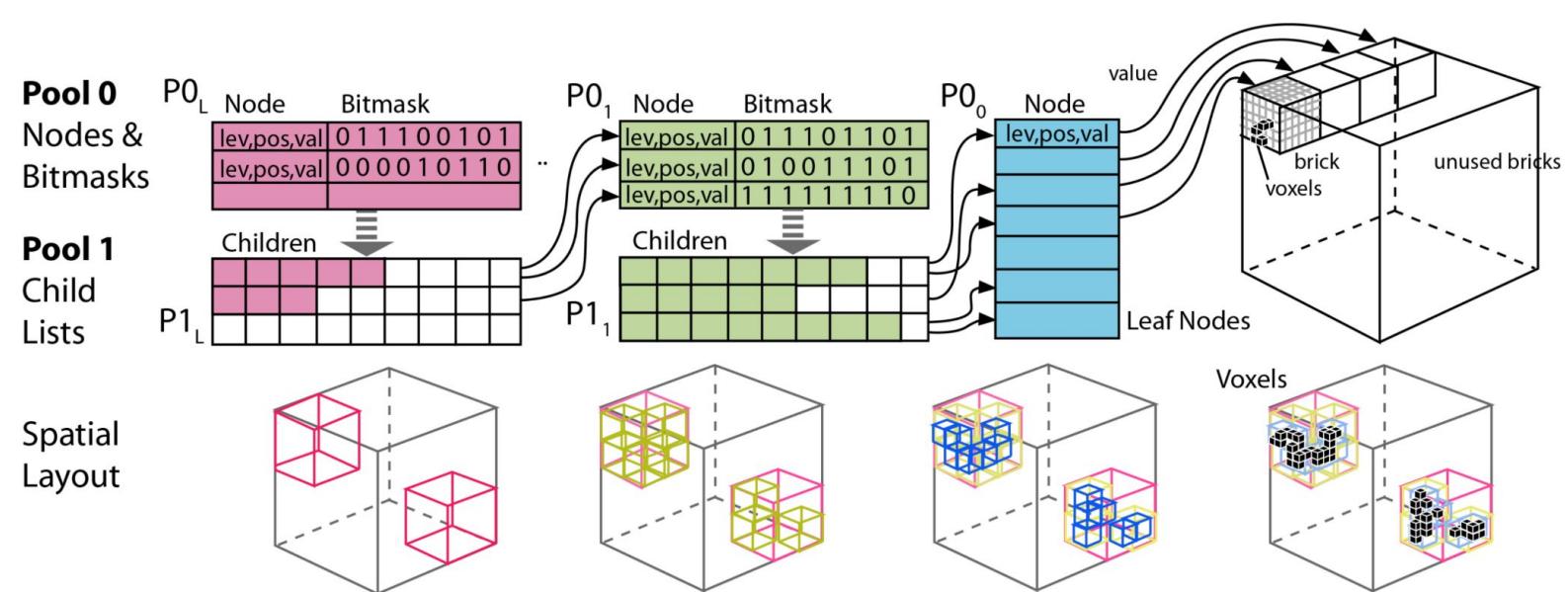
VOXELS

- Memory footprint
- Half res Disney cloud = 45GB
- Need a sparse data structure



VDB

- Default for movie rendering
- Sparse 3 layer B+tree
- Faster sampling
- Massively smaller memory footprint



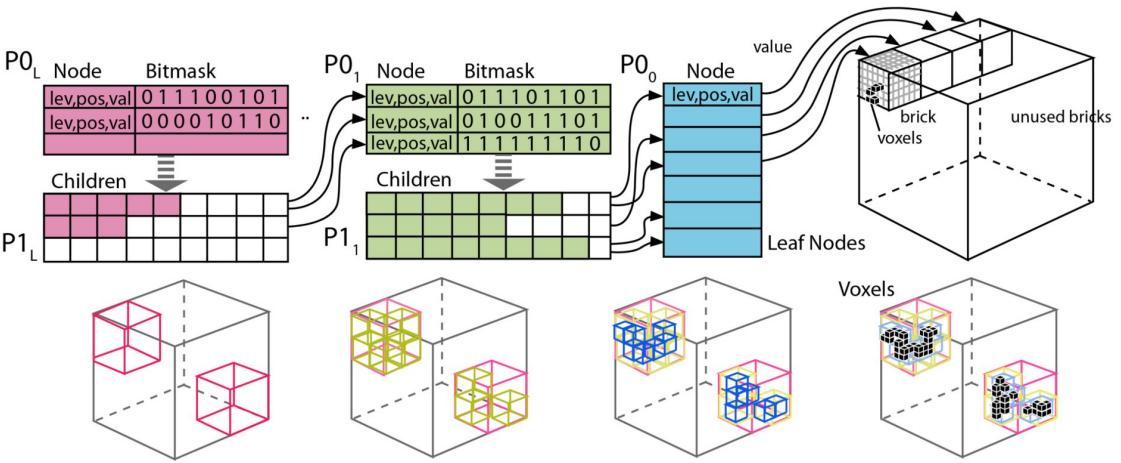
VDB

```
#[repr(C)]
#[derive(Zeroable, Clone, Copy, NoUninit)]
pub struct NodeL1 {
    pub child_mask: [u32; L1_CHILDREN / 32],
    pub child_pointer: u32,
}

#[repr(C)]
#[derive(Zeroable, Clone, Copy, NoUninit)]
pub struct NodeL2 {
    pub child_mask: [u32; L2_CHILDREN / 32],
    pub child_pointer: u32,
}

#[repr(C)]
#[derive(Clone, Copy, Zeroable, NoUninit)]
pub struct Leaf {
    pub value_mask: [u32; L3_CHILDREN / 32],
    pub values_pointer: u32,
}
```

Pool 0
Nodes &
Bitmasks
Pool 1
Child
Lists



VDB

```
pub struct VdbVolumeAsset<T: Pod> {
    pub name: String,
    pub l1: Vec<NodeL1>,
    pub l2: Vec<NodeL2>,
    pub l3: Vec<Leaf>,
    pub voxel_data: Vec<T>,

    pub aabb_min: Vec3,
    pub aabb_max: Vec3,

    pub translation: Vec3,
    pub scale: Vec3,

    pub global_density_majorant: f32,
}
```

Pool 0
Nodes &
Bitmasks

Node	Bitmask
lev, pos, val	011100101
lev, pos, val	000010110

Pool 1
Child
Lists

P1 _L

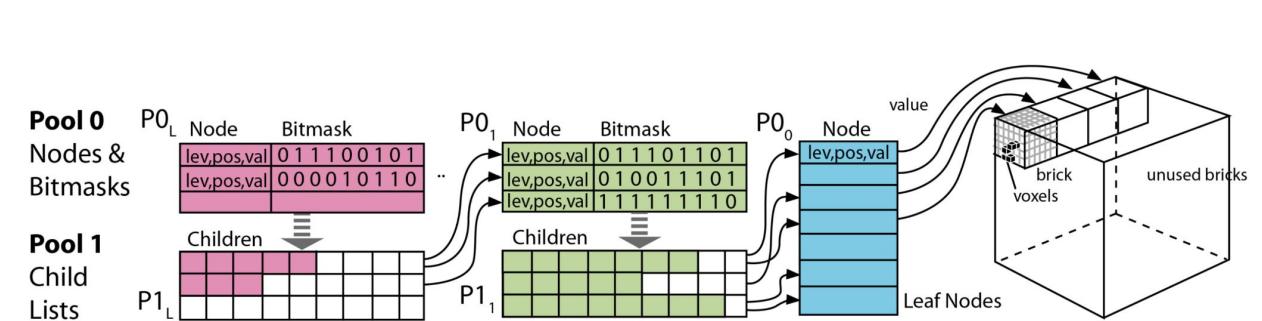
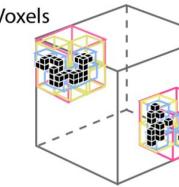
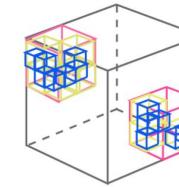
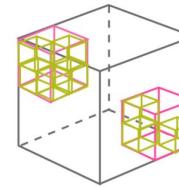
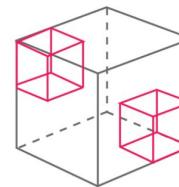
P0 ₁	Node	Bitmask
	lev, pos, val	011101101
..	lev, pos, val	010011101
	lev, pos, val	111111110

Children

..

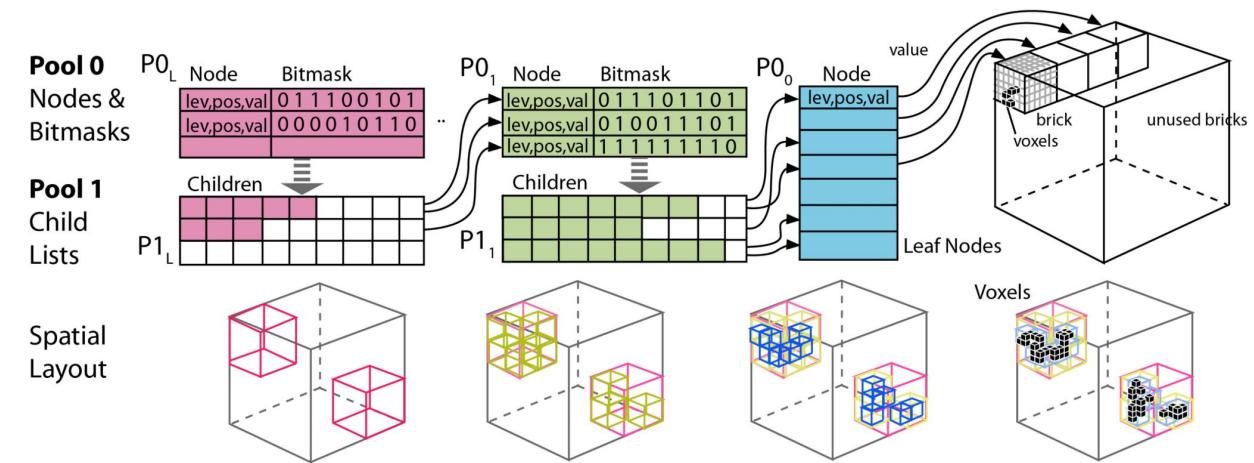
Children

Spatial
Layout



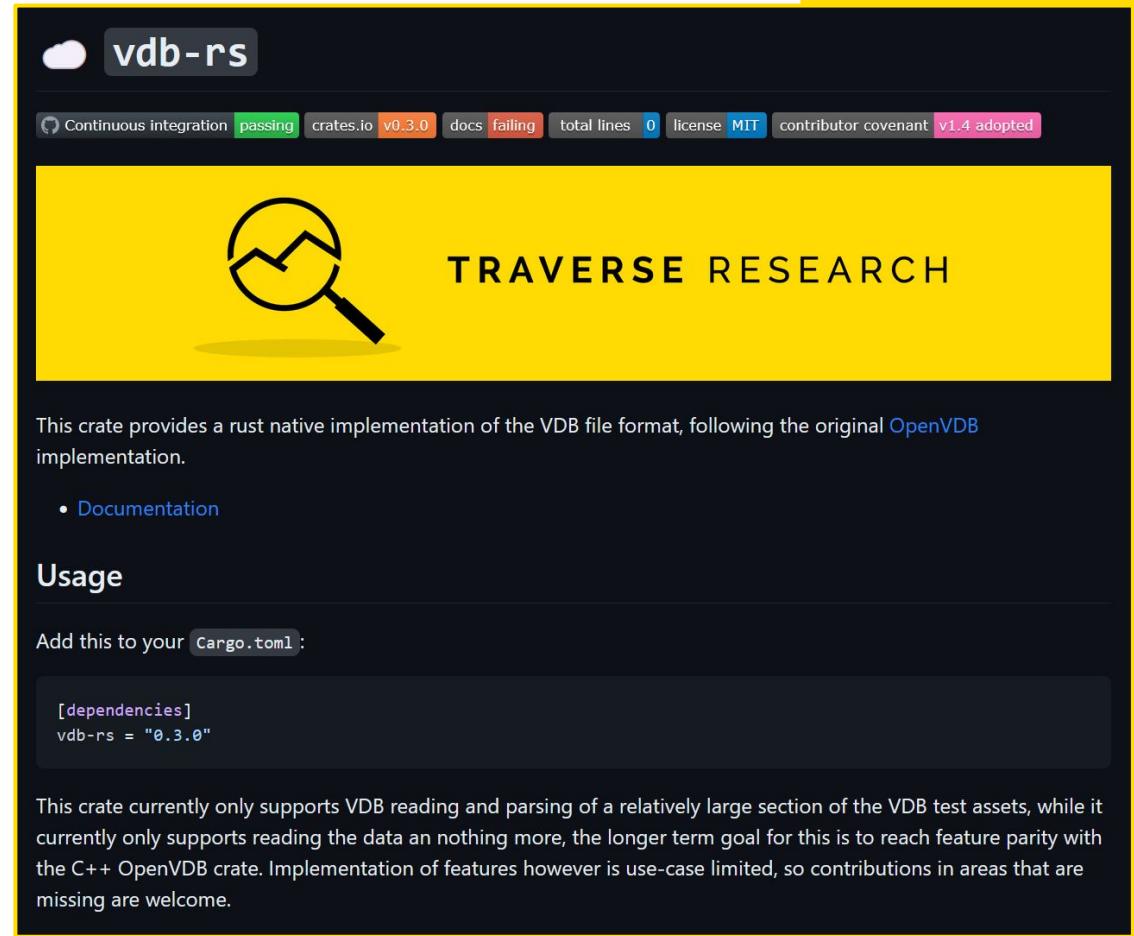
VDB

```
pub struct GpuVdbVolume {  
    pub constants: Arc<dyn Buffer>,  
    pub l1_buffer: Arc<dyn Buffer>,  
    pub l2_buffer: Arc<dyn Buffer>,  
    pub l3_buffer: Arc<dyn Buffer>,  
    pub voxel_data_buffer: Arc<dyn Buffer>,  
}
```



VDB-RS

- <https://github.com/Traverse-Research/vdb-rs>
- Supports Houdini, blender and more



The screenshot shows the crate page for `vdb-rs` on crates.io. The header features the `vdb-rs` logo and navigation links for continuous integration (passing), crates.io (v0.3.0), docs (failing), total lines (0), license (MIT), and contributor covenant (v1.4 adopted). The main content area has a yellow background with a magnifying glass icon and the text "TRAVERSE RESEARCH". It describes the crate as providing a rust native implementation of the VDB file format, following the original OpenVDB implementation. It includes a link to documentation and a section for usage with a code snippet for Cargo.toml. A note at the bottom states that the crate currently only supports VDB reading and parsing.

This crate provides a rust native implementation of the VDB file format, following the original [OpenVDB](#) implementation.

- [Documentation](#)

Usage

Add this to your `Cargo.toml`:

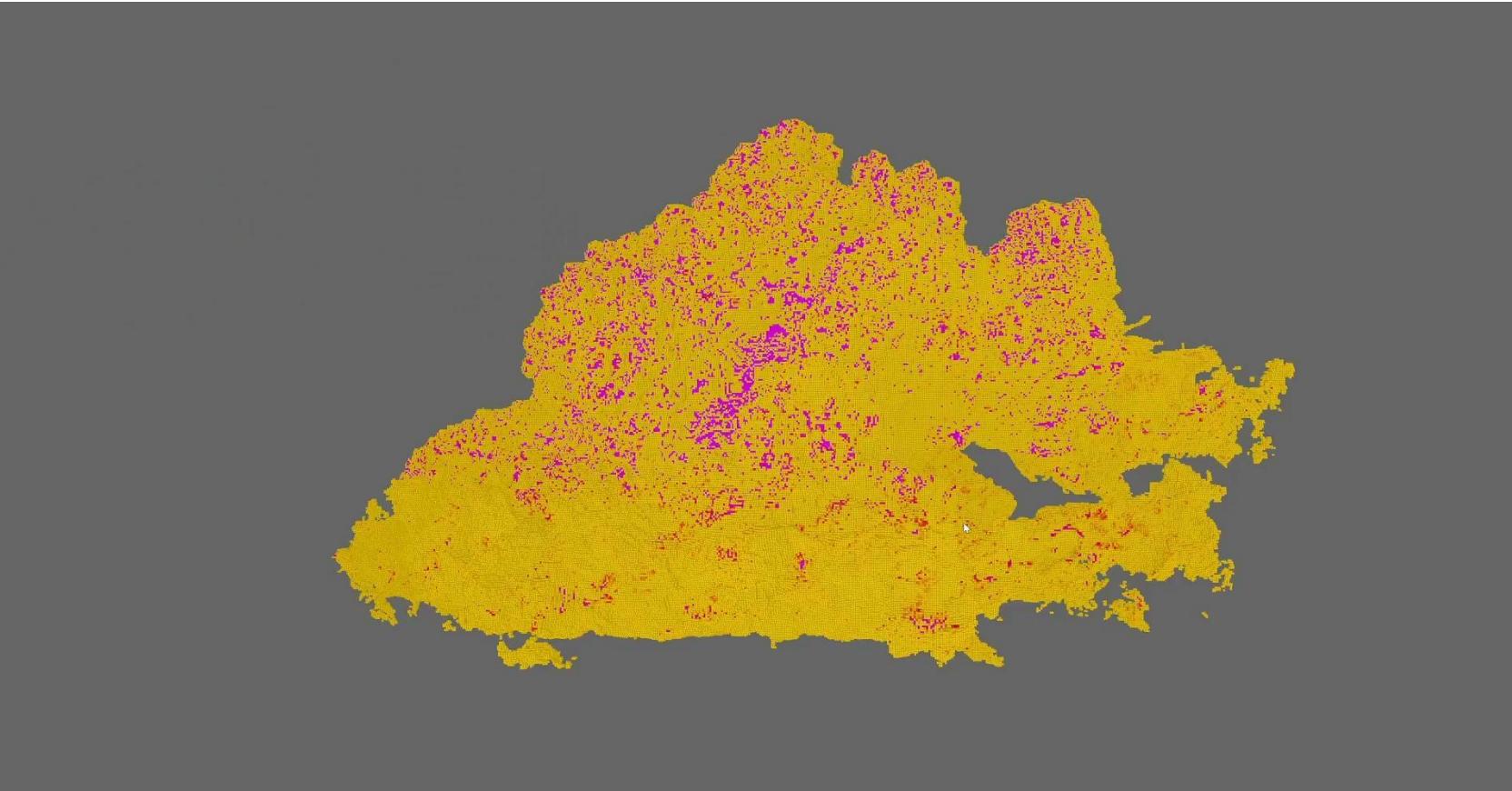
```
[dependencies]
vdb-rs = "0.3.0"
```

This crate currently only supports VDB reading and parsing of a relatively large section of the VDB test assets, while it currently only supports reading the data and nothing more, the longer term goal for this is to reach feature parity with the C++ OpenVDB crate. Implementation of features however is use-case limited, so contributions in areas that are missing are welcome.



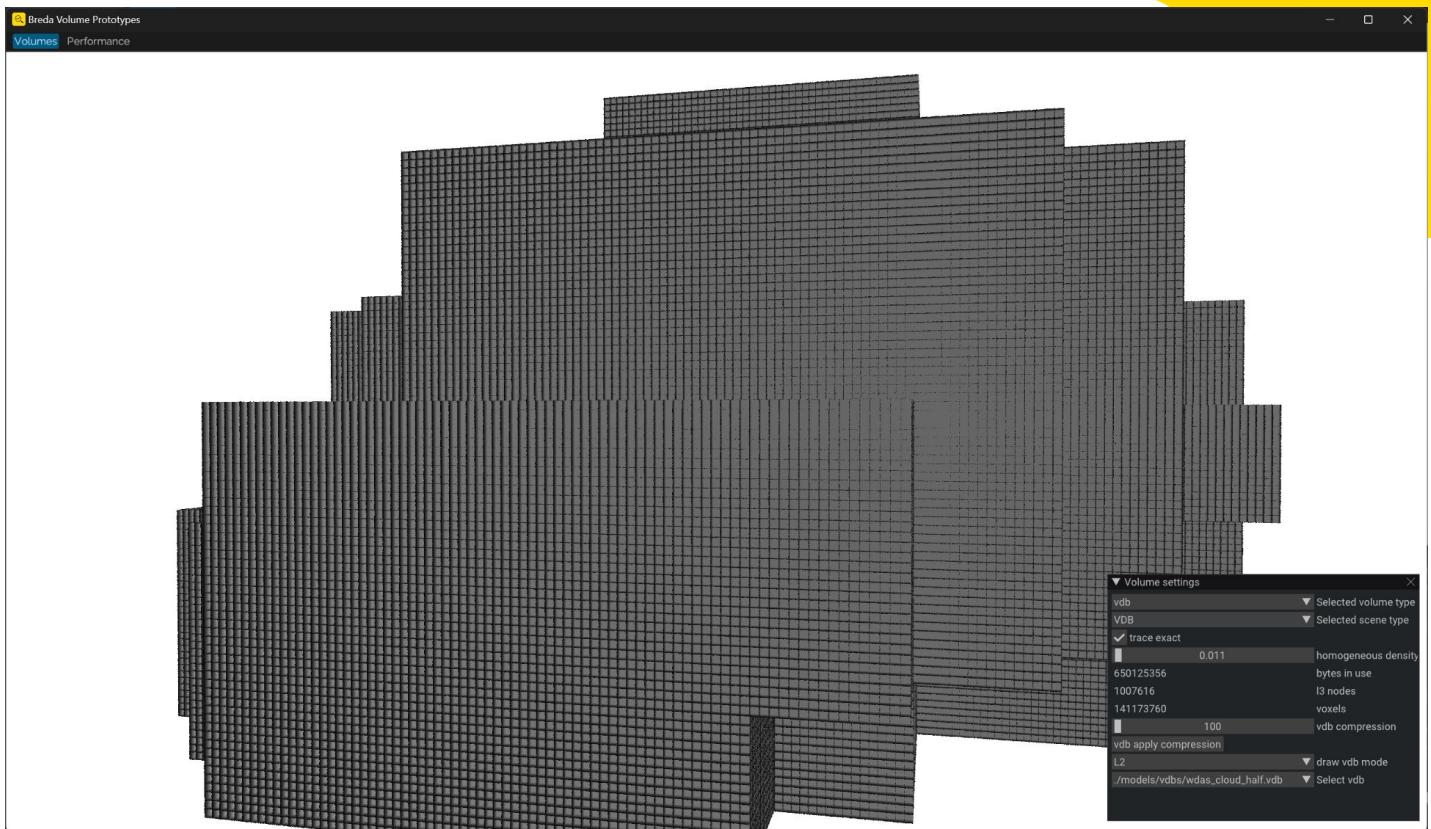
VDB-RS

- Quarter res Disney cloud due to rendering limitations
- Using bevy



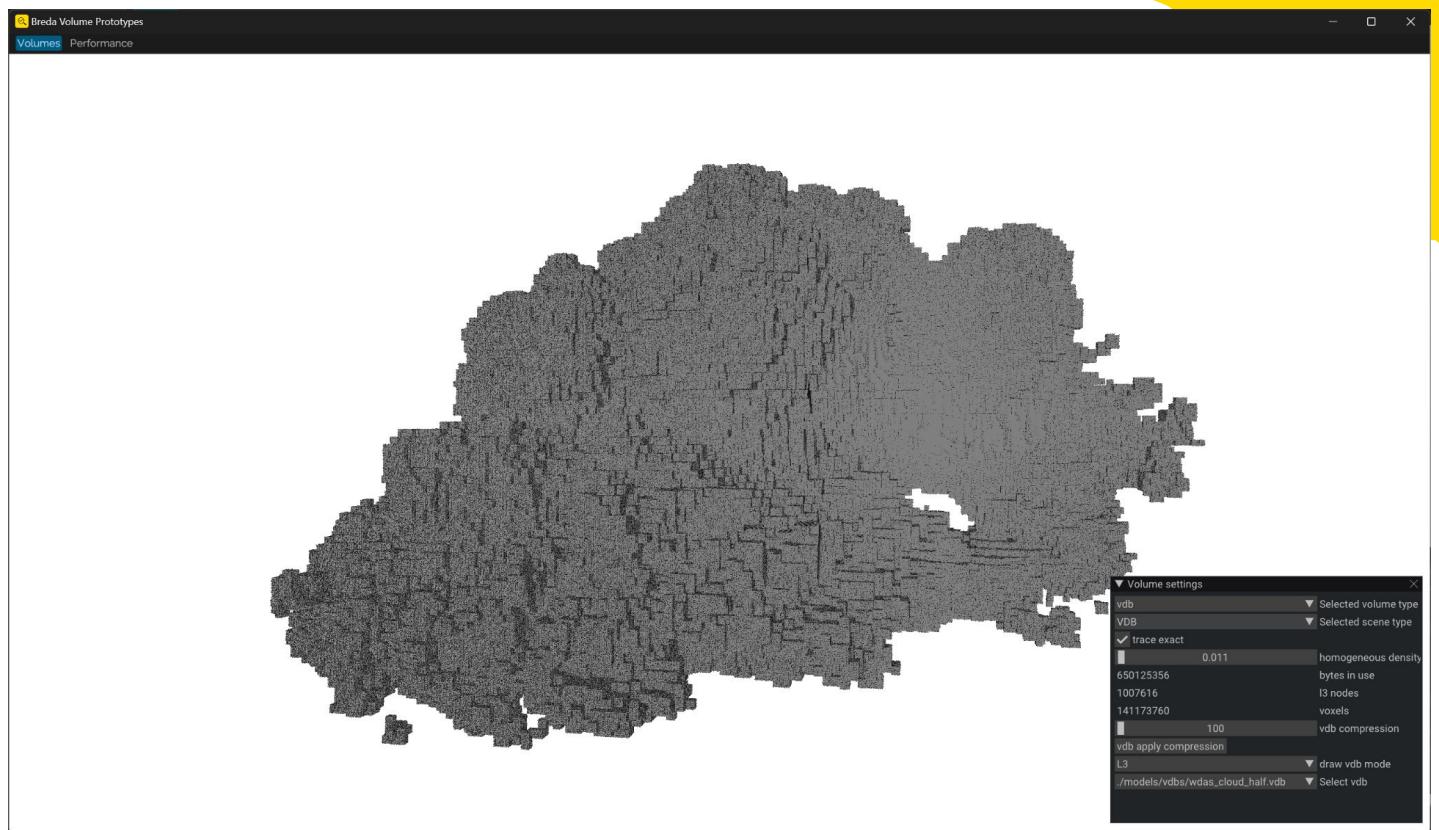
VDB-RS

- L2 nodes



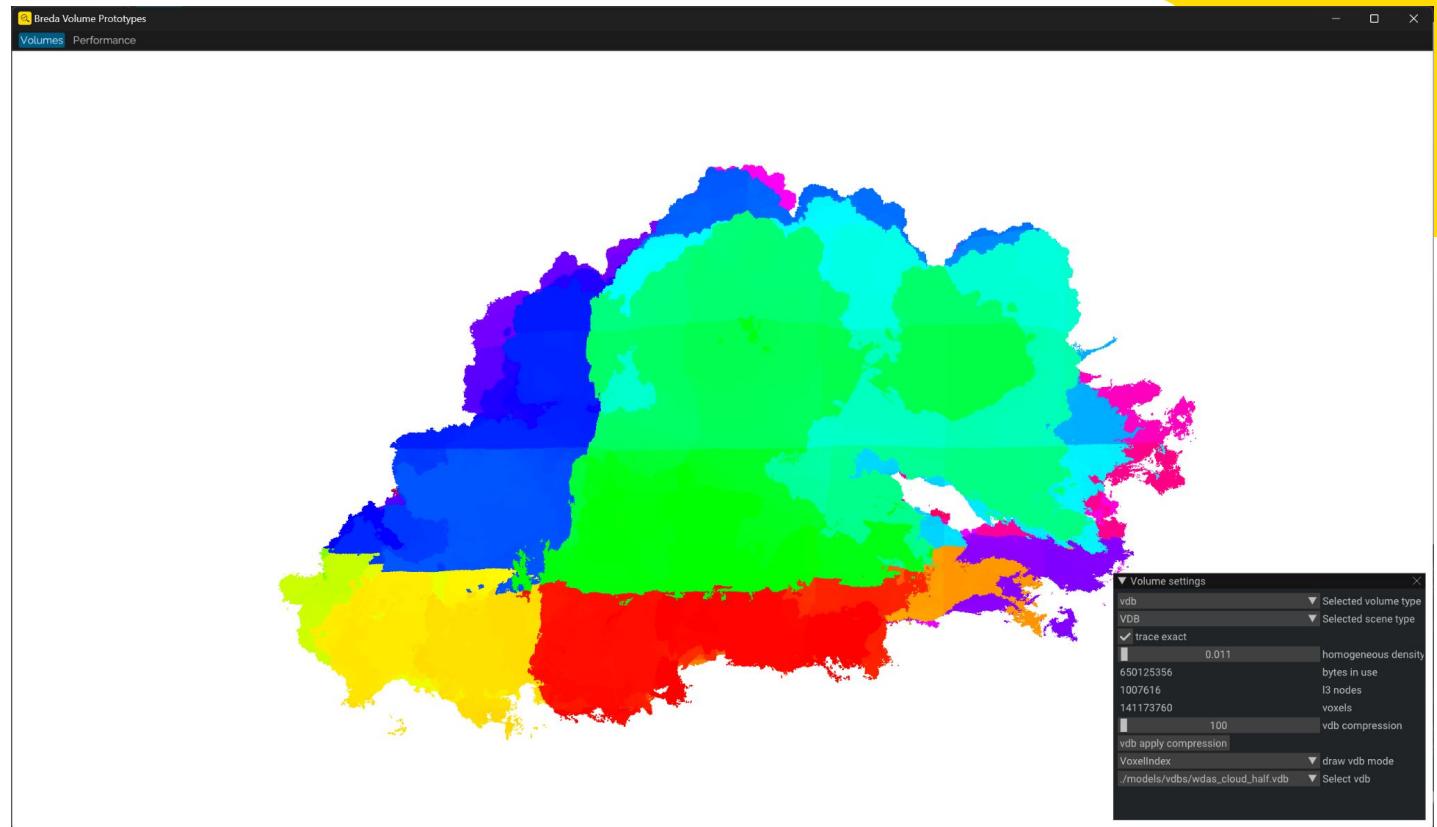
VDB-RS

- L3 nodes



VDB-RS

- Voxel data indices indicated by hue



BREDA OUR BELOVED

BREDA OUR BELOVED

- With many many thanks to Emilio



62



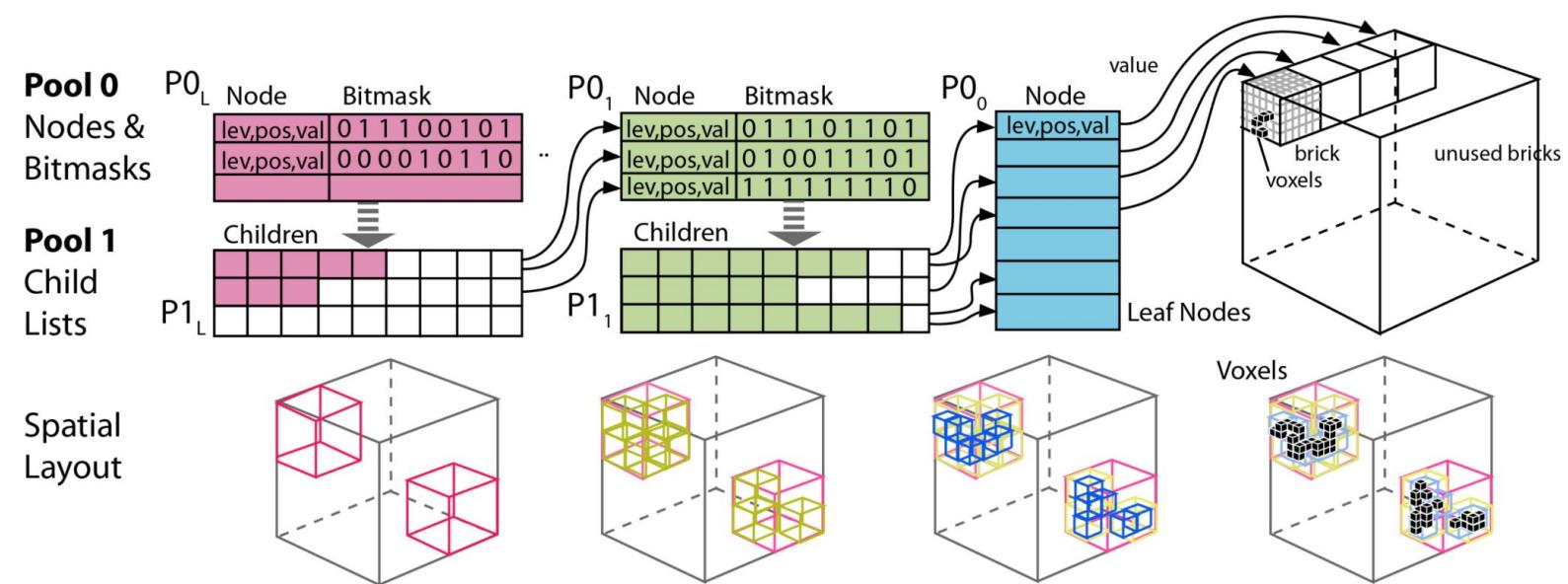
6



ACTIVE RESEARCH

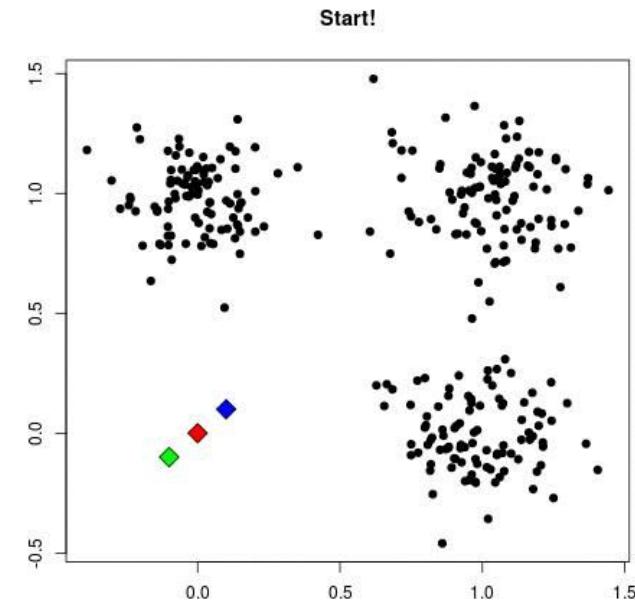
ACTIVE RESEARCH: COMPRESSION

- Remember the VDB structure
- 8^3 brick size = 512 voxels
- Find similar bricks



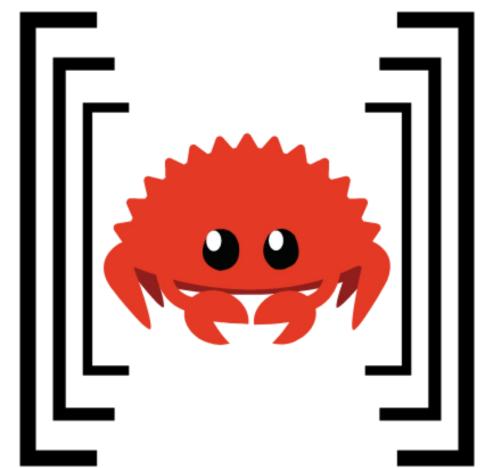
ACTIVE RESEARCH: COMPRESSION

- Clustering
- Complexity of $O(512nk) = O(nk)$
- Distance metrics?



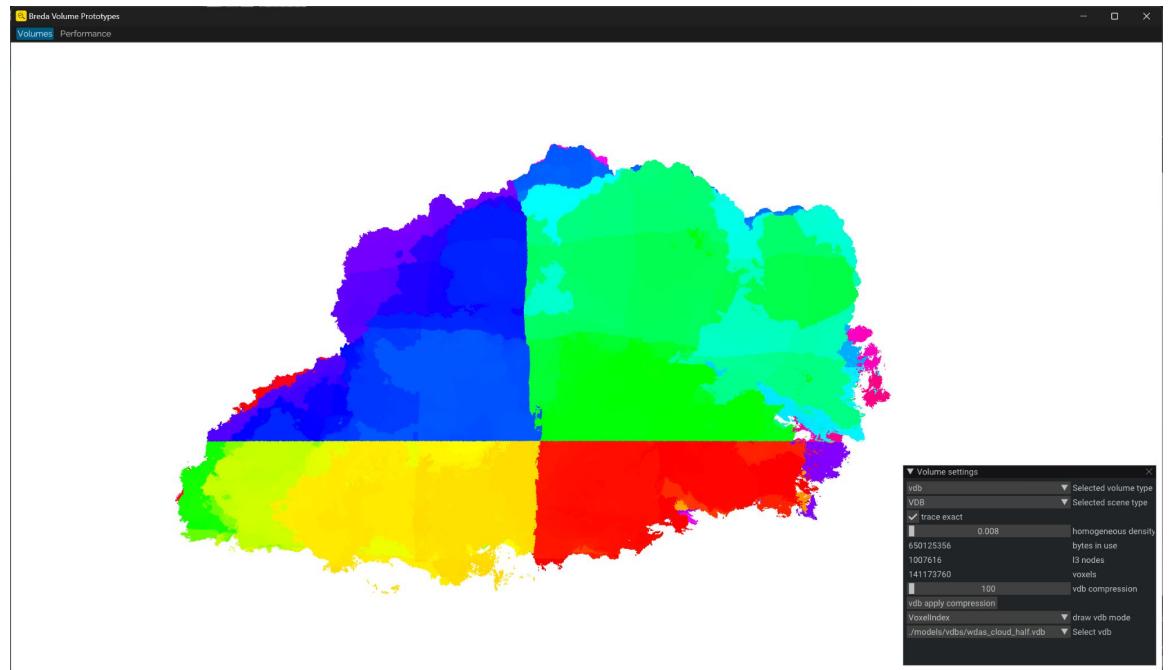
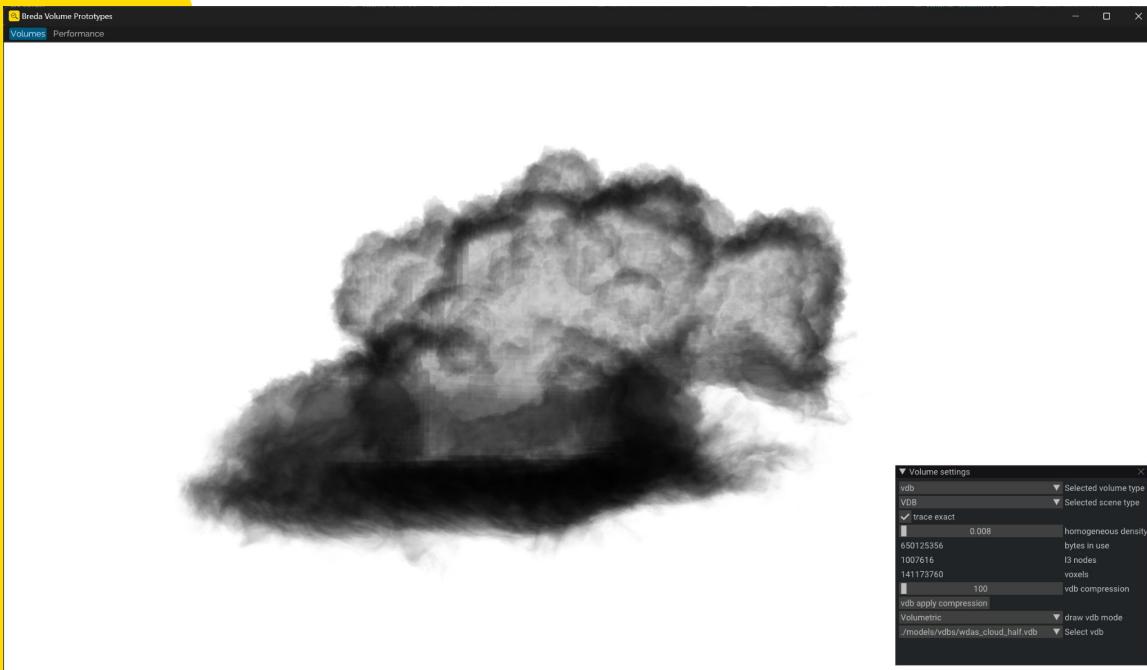
ACTIVE RESEARCH: COMPRESSION

- Nalgebra: complex
- Ndarray: numpy-like



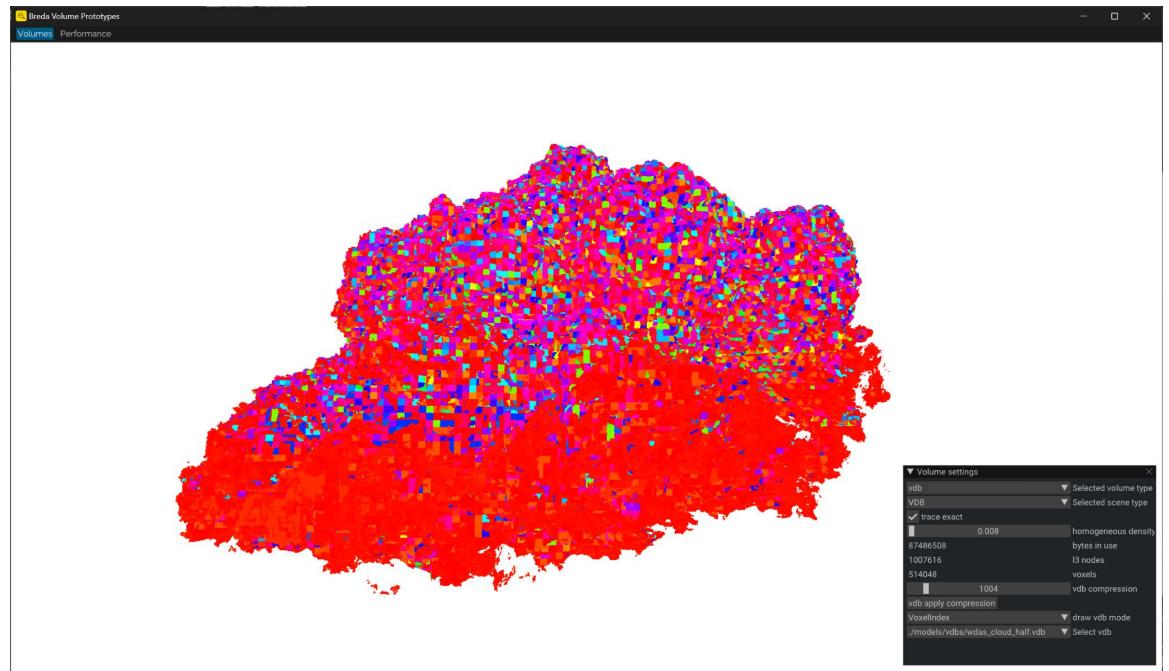
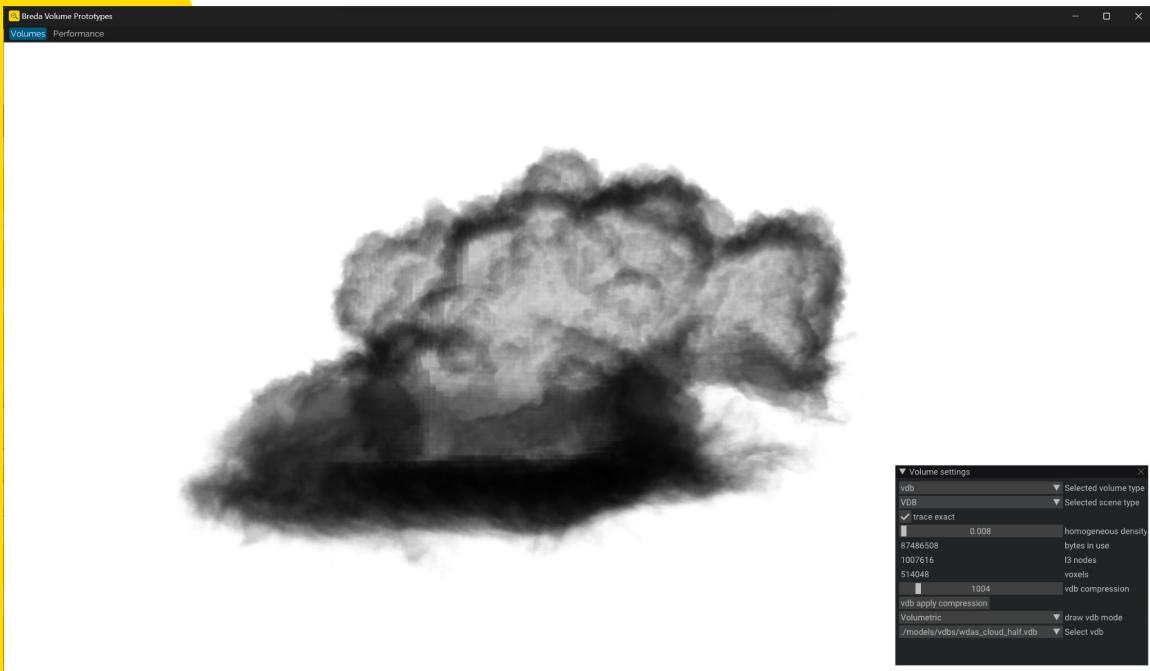
ACTIVE RESEARCH: COMPRESSION

- Ground truth
- 650mb



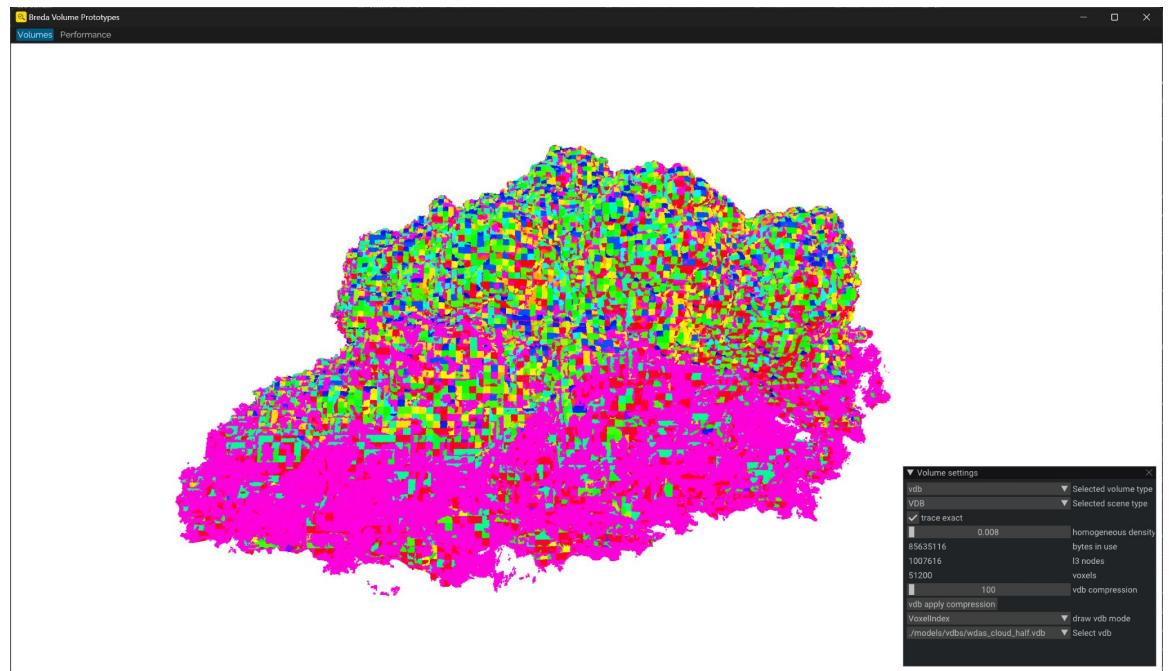
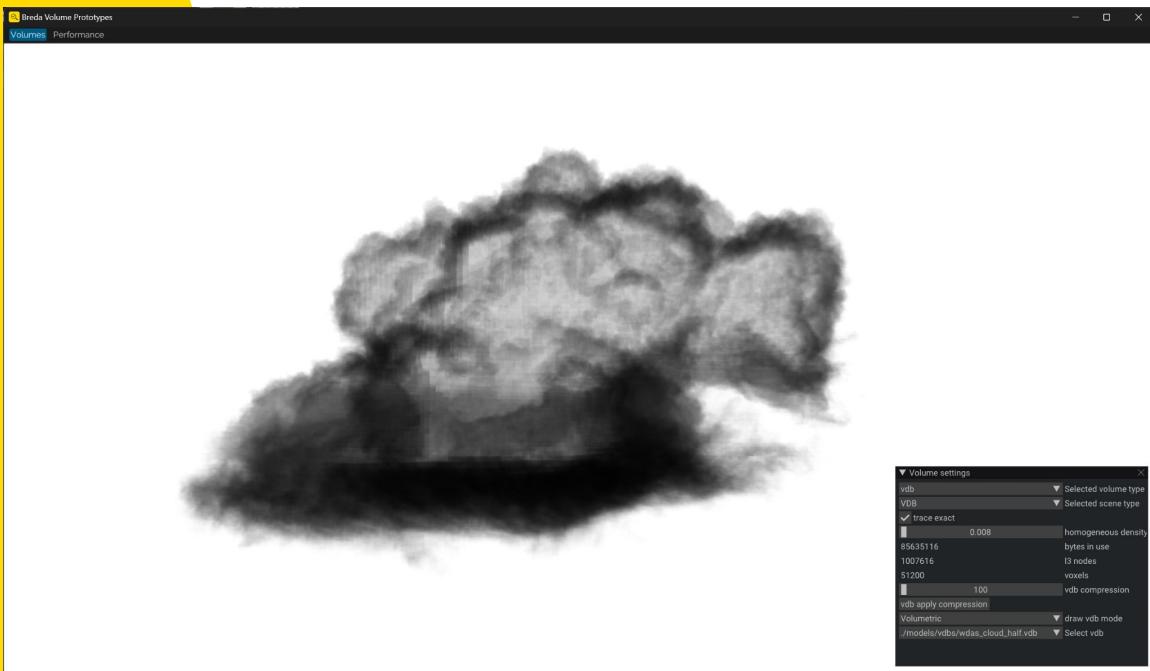
ACTIVE RESEARCH: COMPRESSION

- $k = 1004$
- 87mb
- 12 minutes



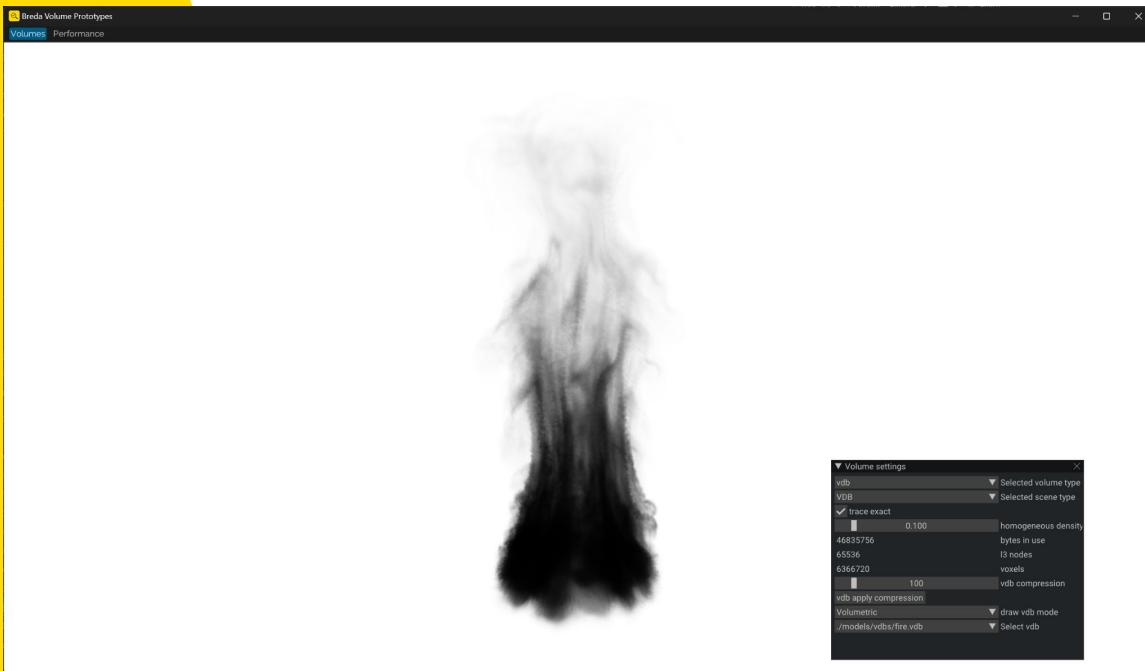
ACTIVE RESEARCH: COMPRESSION

- k = 100
- 86mb
- 6 minutes



ACTIVE RESEARCH: COMPRESSION

- Great for large effect
- Less so for small detailed effect
- 47mb => 22mb



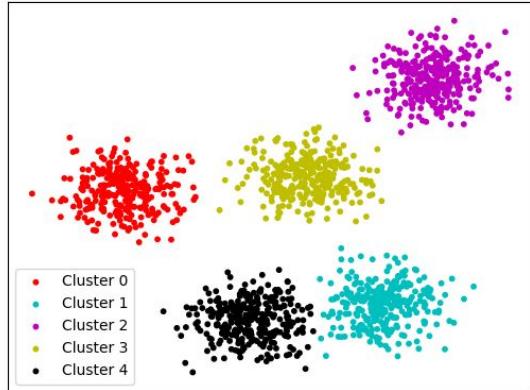
ACTIVE RESEARCH: ANIMATION

- What about the animated storms and smoke
- Flipbook

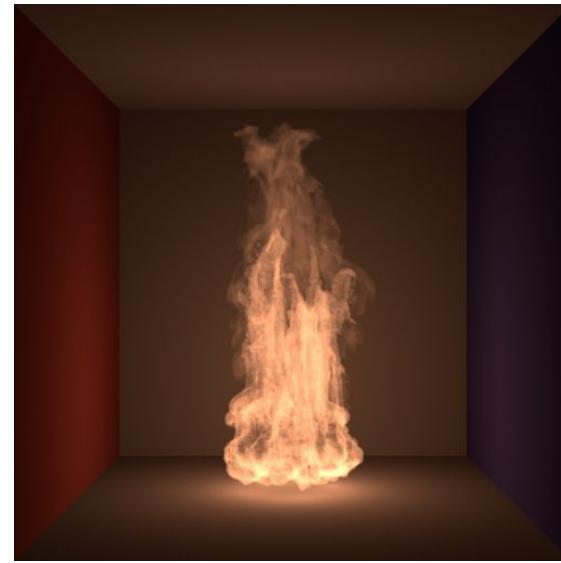


WHAT'S NEXT

ANIMATION COMPRESSION



EMISSION



PERFORMANCE



A wide-angle photograph of a mountainous landscape at sunset or sunrise. A two-lane asphalt road with a yellow center line curves from the foreground towards the horizon. The mountains in the background are rugged with patches of snow and ice clinging to their rocky faces. The sky is filled with large, billowing clouds colored in shades of orange, yellow, and grey.

THANK YOU