OpenVDB Course:

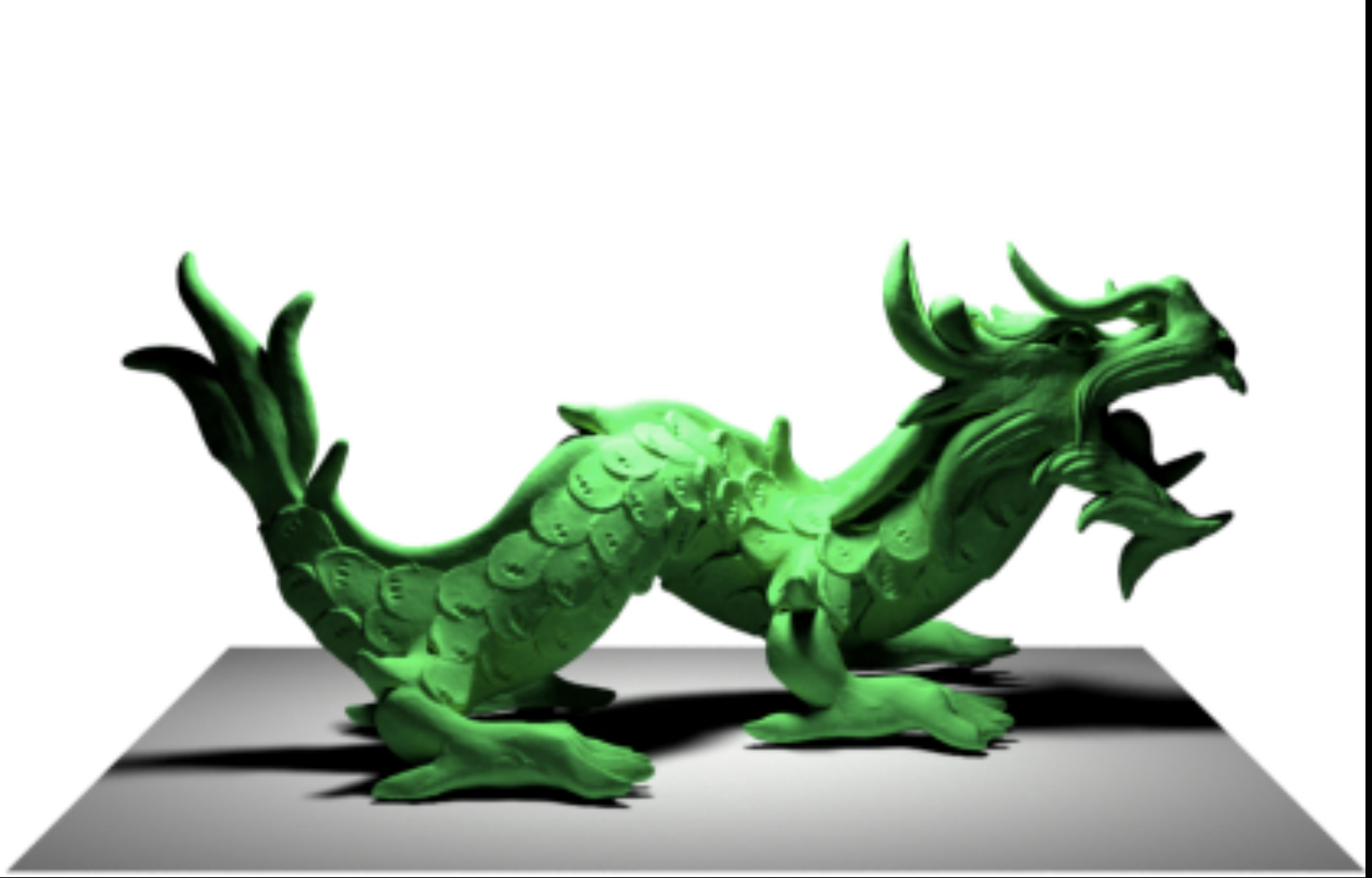
Advanced Applications of OpenVDB in Production

double negative visual effects
dan bailey (dan@dneg.com)

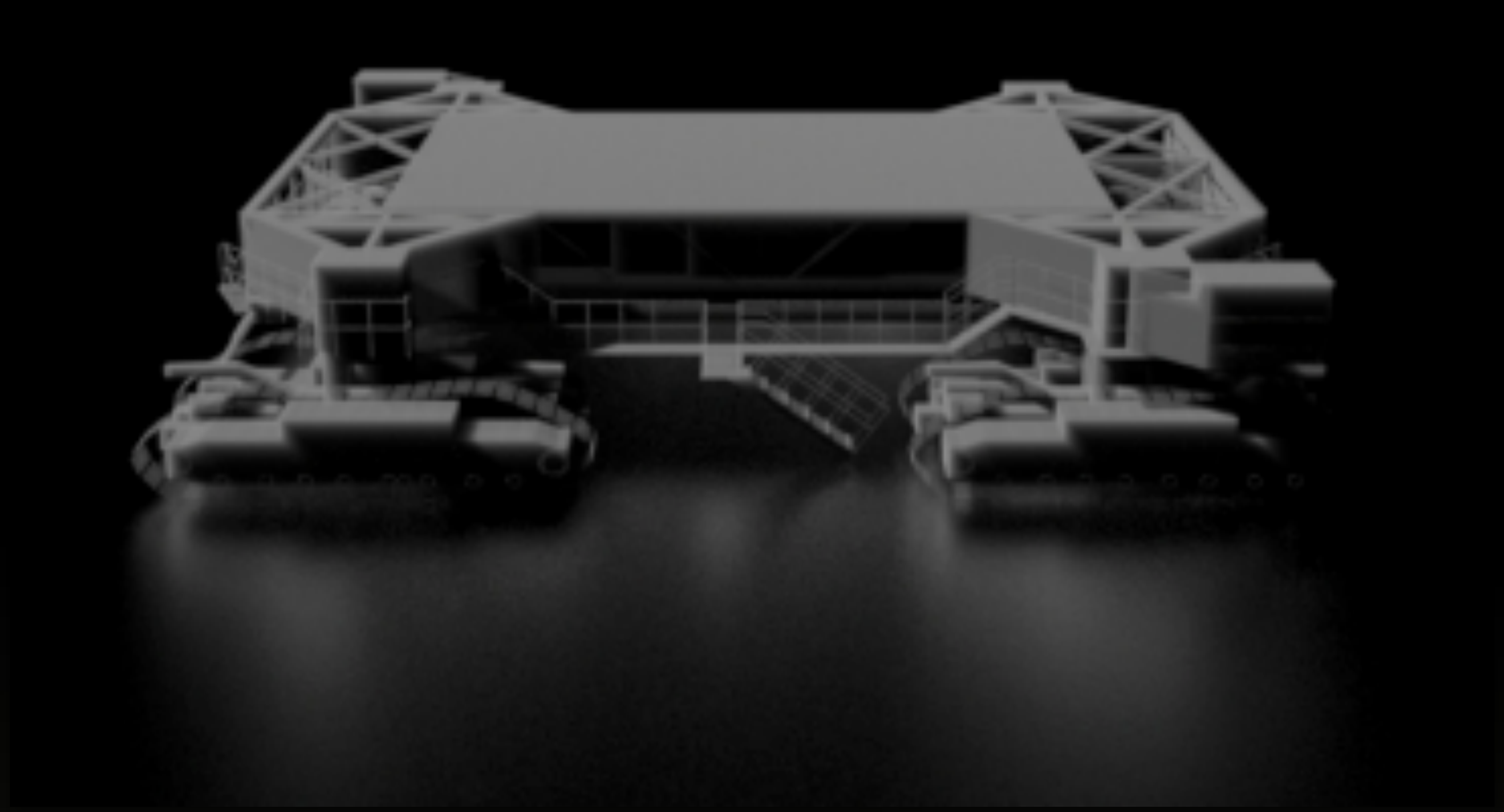
OpenVDB

OpenVDB Uses

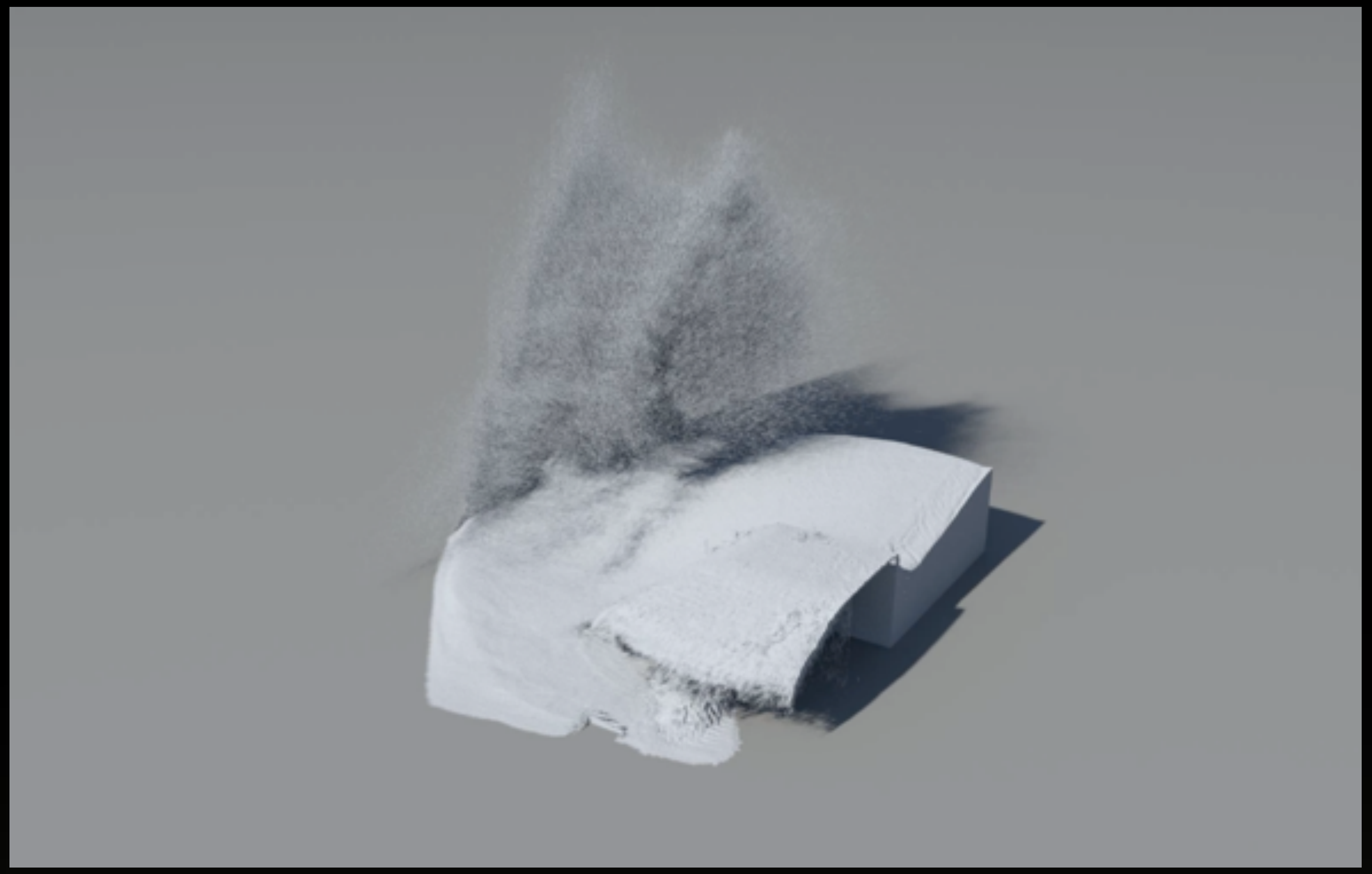
double negative visual effects



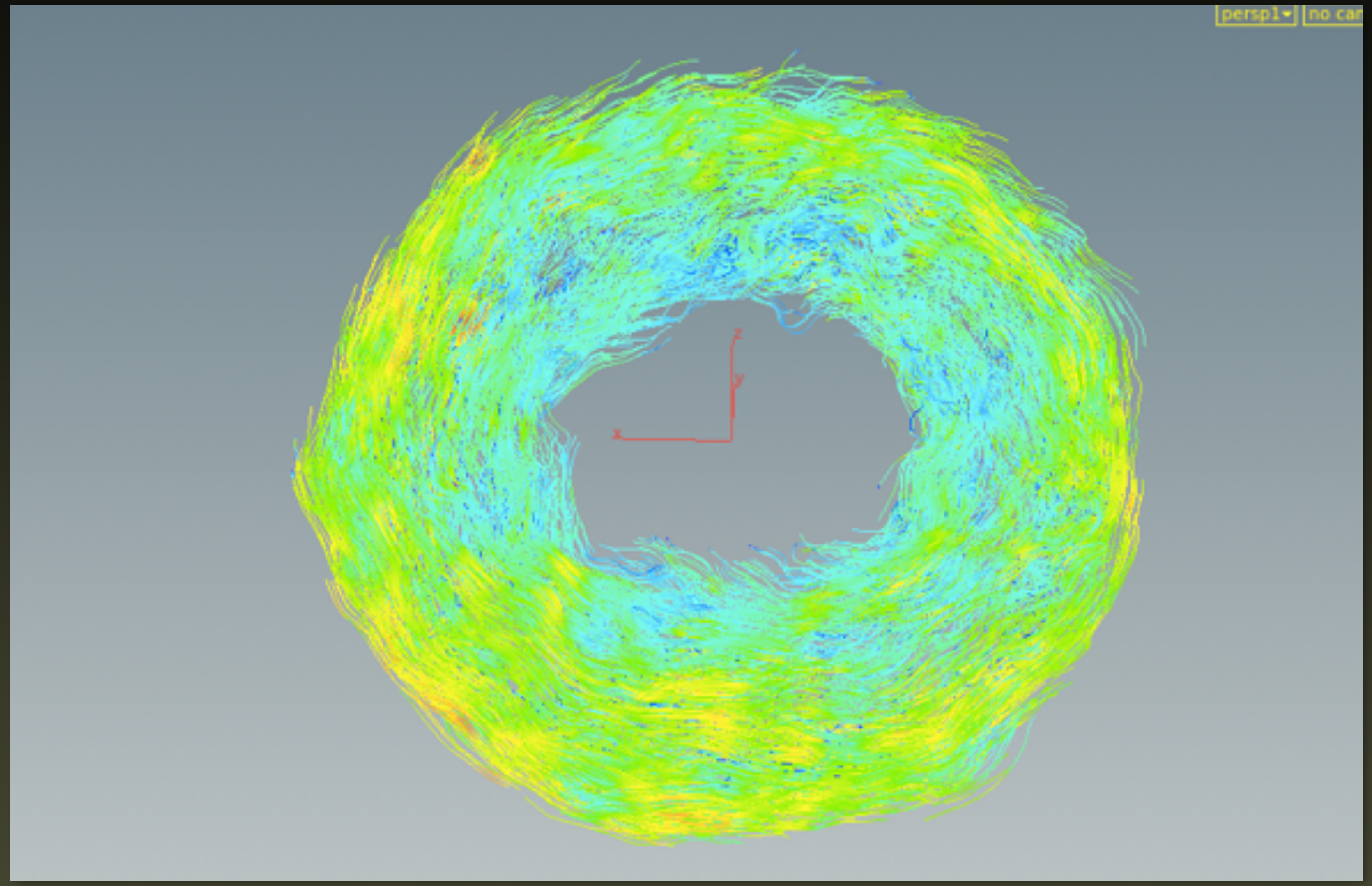
Level Set



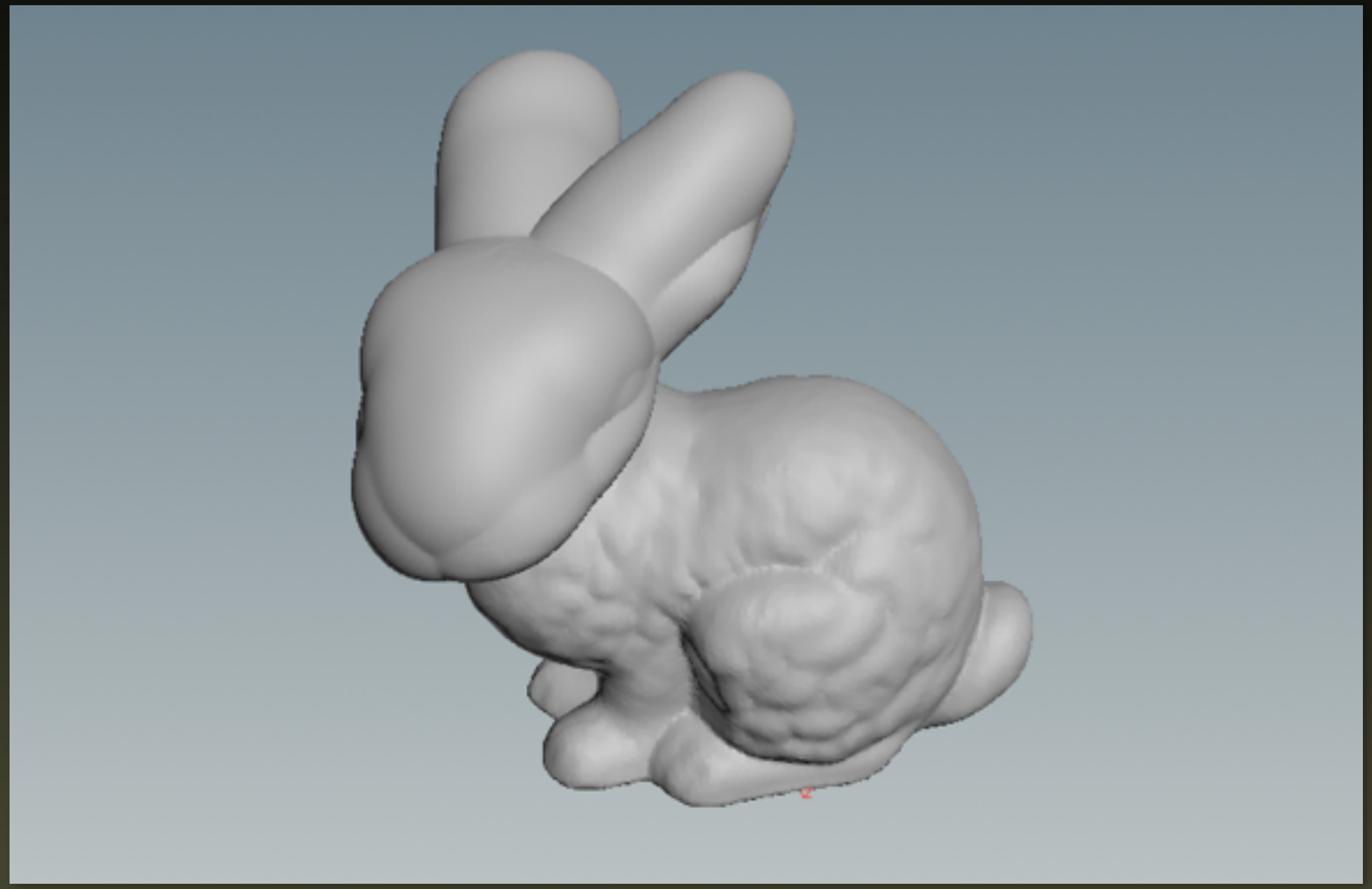
Fog Volume



Points (New)



Vector Field



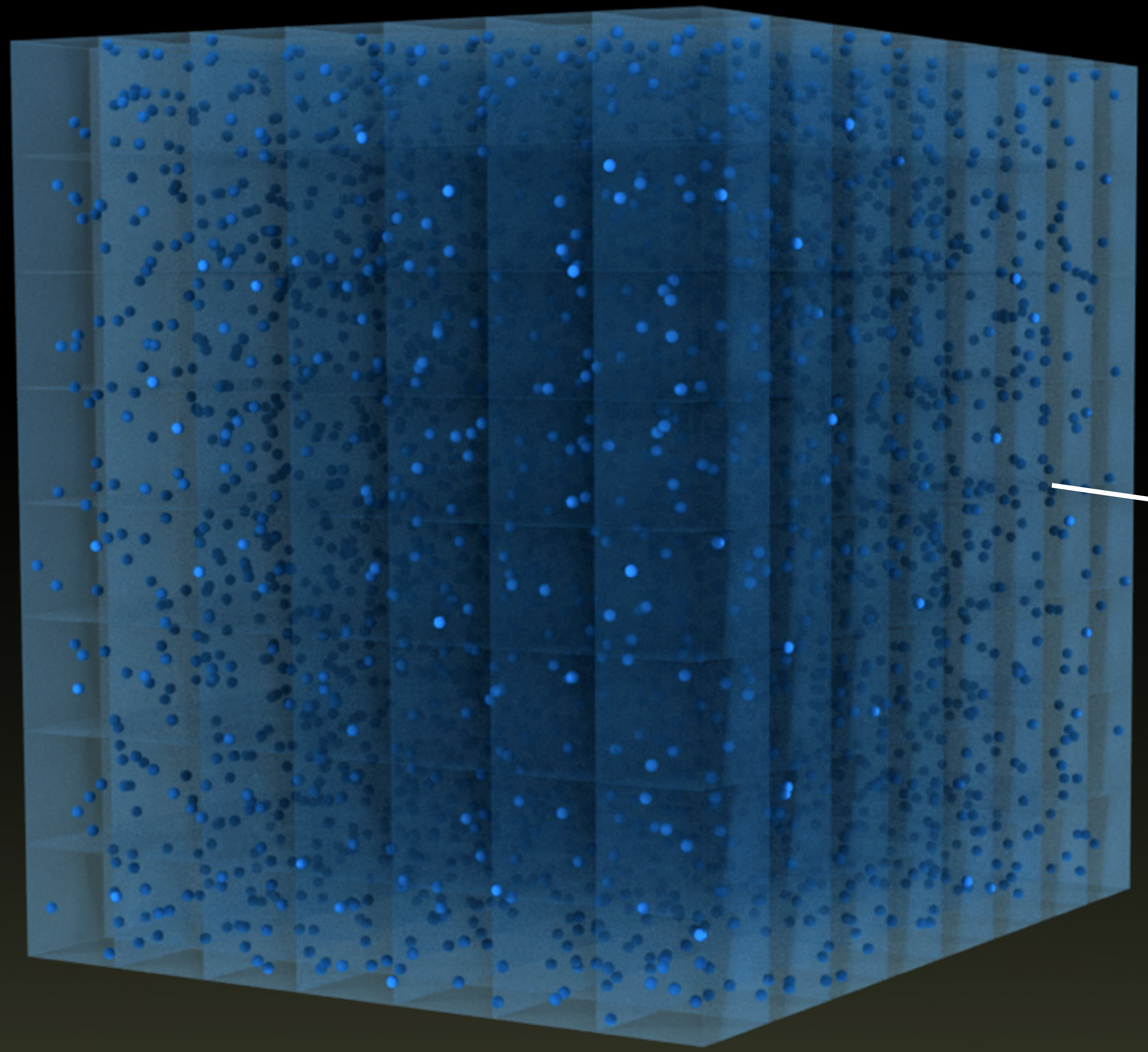
Alpha Mask

Attributes

double **negative** visual effects

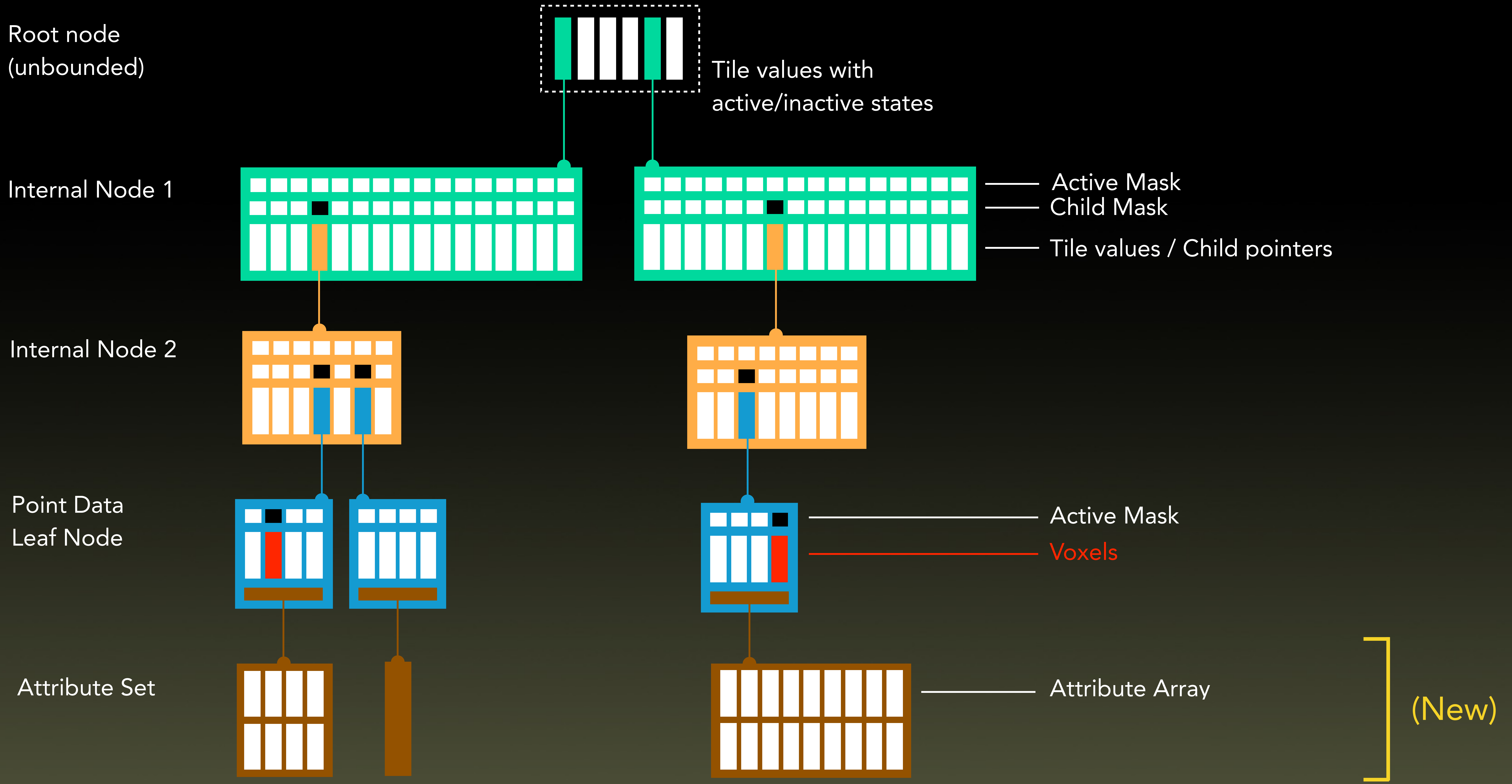
- OpenVDB extended with new Attribute API
- AttributeArray.h and AttributeSet.h

VDB Points Leaf Node



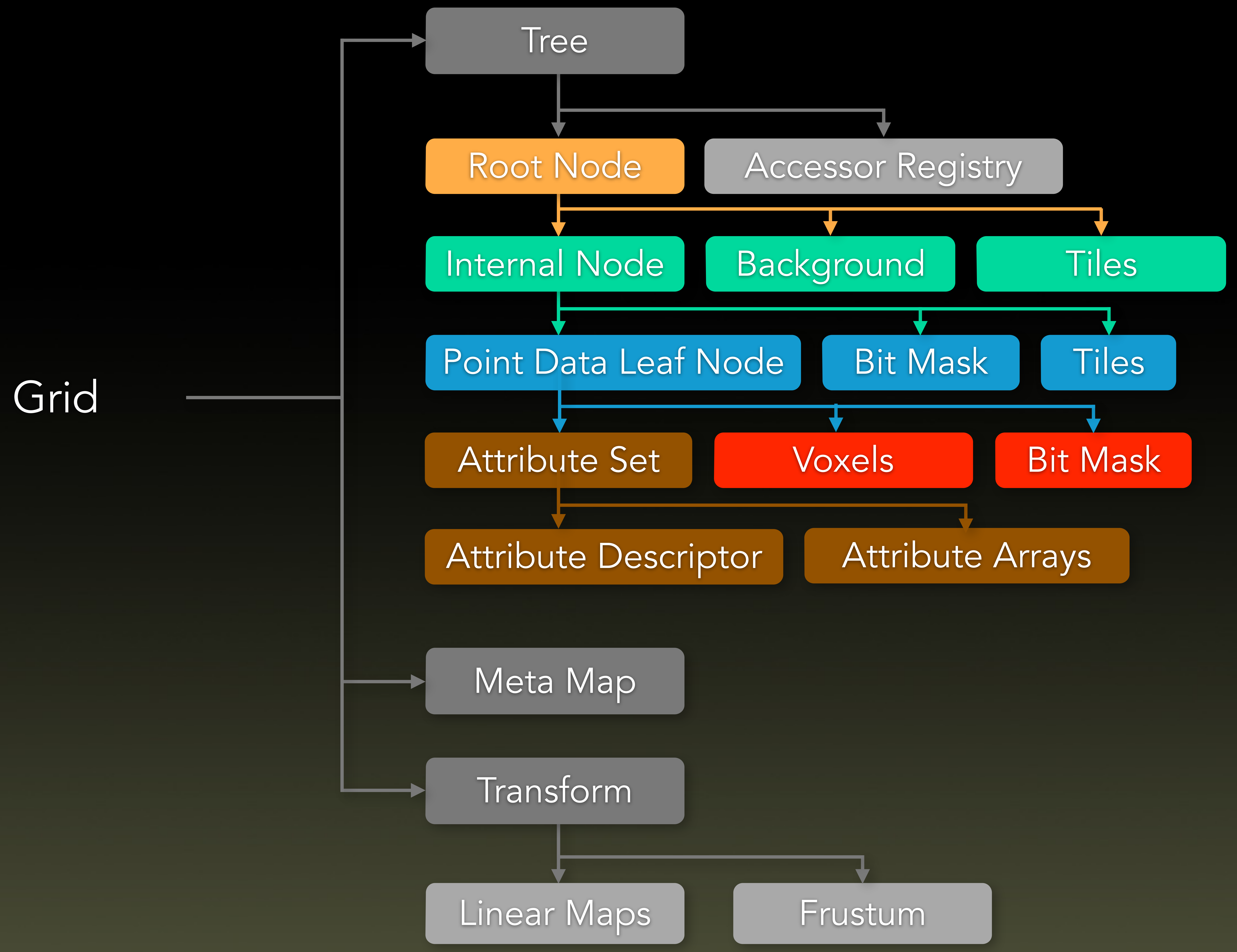
Points located within a Leaf Node are owned by that Leaf along with all of their attributes

VDB Points Data Structure



Point Data Class Structure

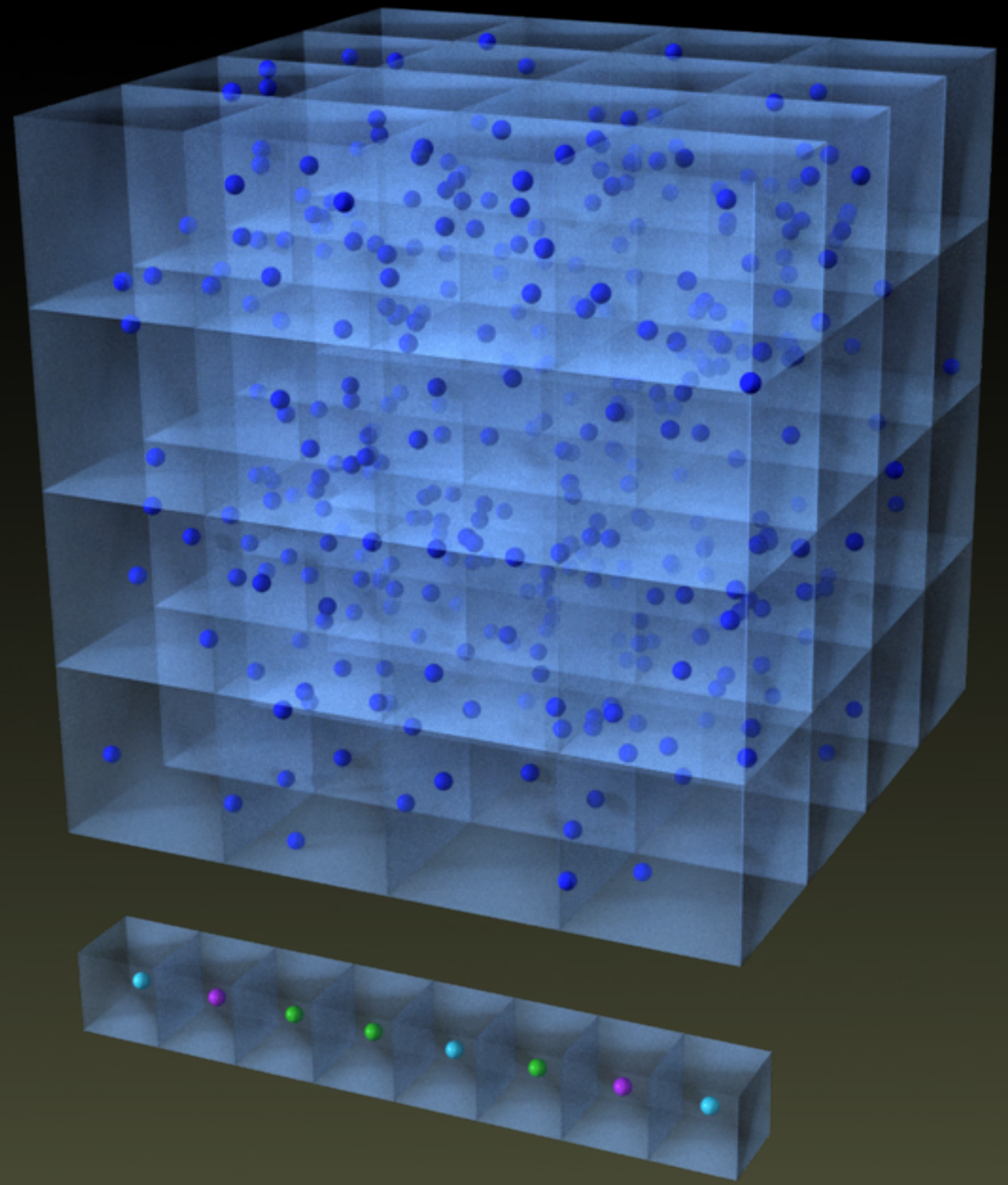
double negative visual effects



Dynamic Attribute Arrays

Leaf Nodes can store different attributes (from each other)

However, not typically supported by tools



OpenVDB

Motivation

double negative visual effects

	Convenience	Performance	Memory	I/O	Distribution
Spatial Organisation	✓			✓	✓
Greater Compression			✓	✓	
Data Locality		✓			
VDB Topology and Tools	✓				

OpenVDB

In-Memory Compression

double negative visual effects

Attribute Compression

(Not Available for Native Houdini Points)

(x, y, z) => (w)

3 x 32-bits 16-bits

Uniform Value Compression

(Available for Native Houdini Points)

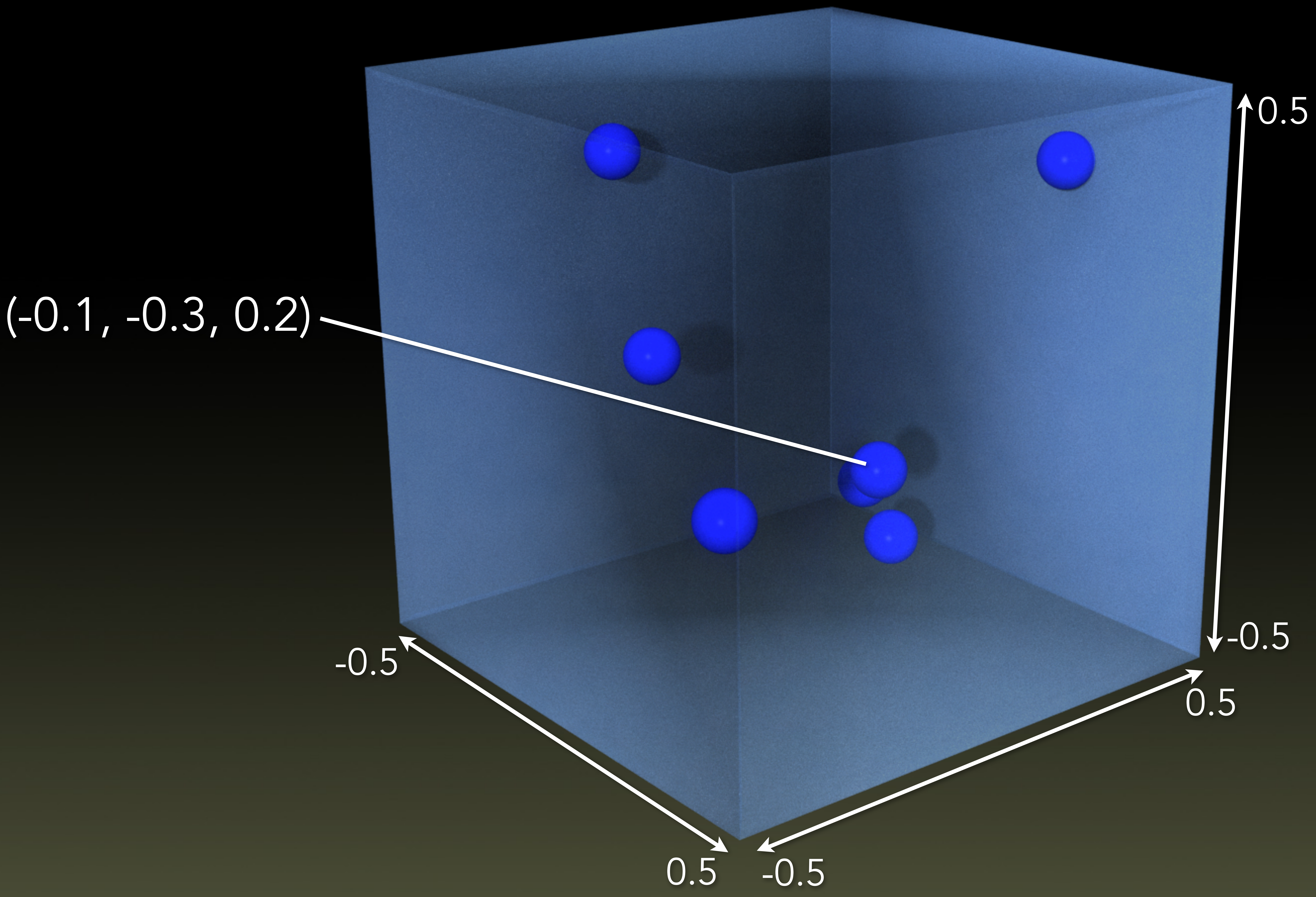
[1, 1, 1, 1, 1, 1, ...] => [1]

Stream Compression

=> =>

(Available for Native Houdini Points in Houdini 14 but only for disk compression)

Position Storage



Floating-point

Exponent

10110111

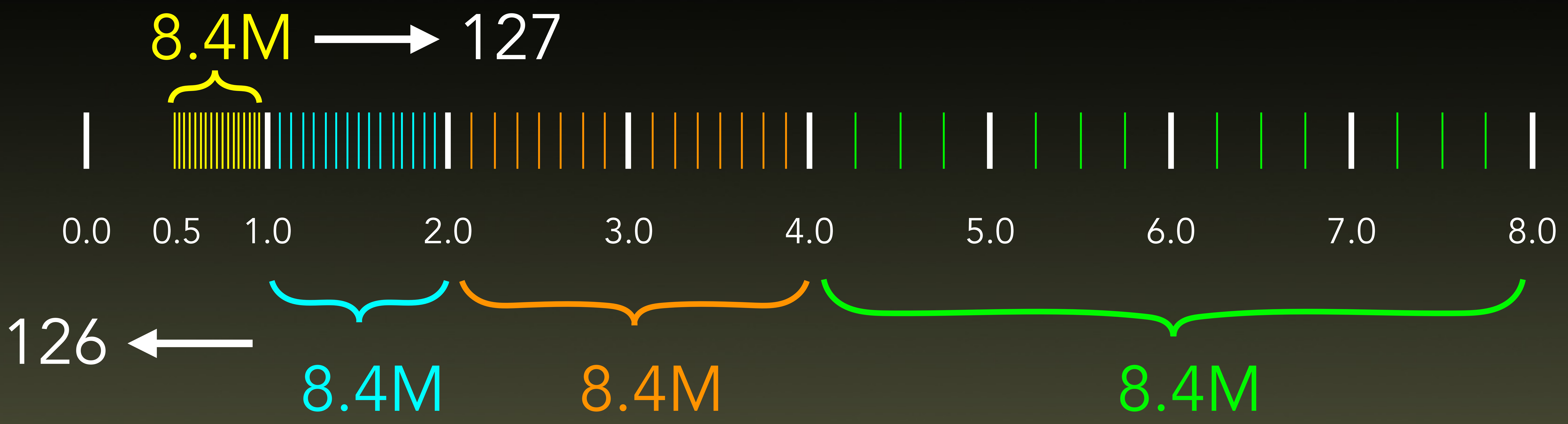
8 bits

Mantissa

010101110101010101010101010

24 bits

$$\boxed{1.\{mantissa\} \times 2^{\{exponent\}}}$$



Quantisation

Encoding

32-bit floating-point → (int) round(x * 2^{16f} + 0.5f) → 16-bit integer

Decoding

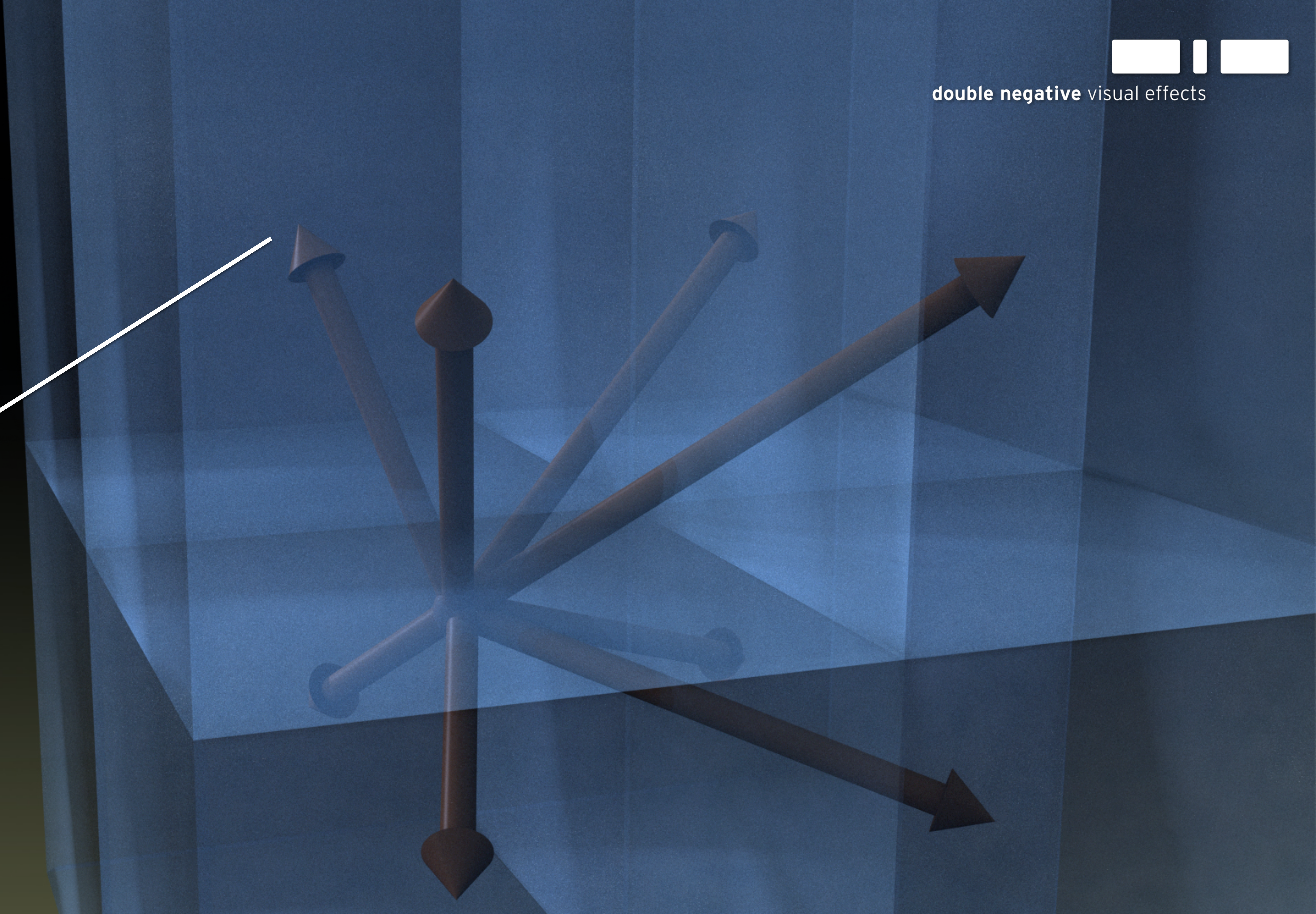
16-bit integer → float(x) / 2^{16f} → 32-bit floating-point

Position Compression

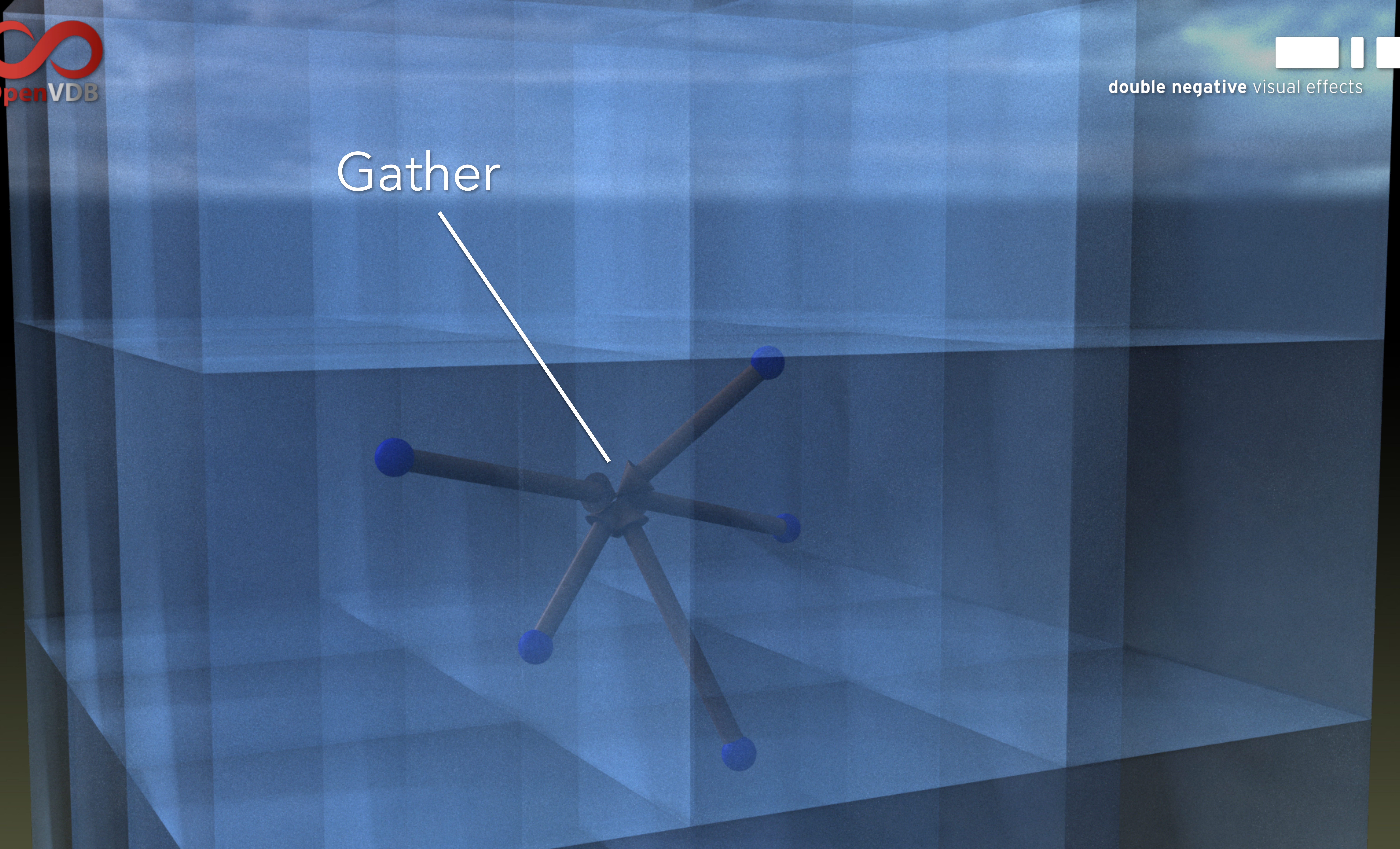
double negative visual effects

Float Array	No Quantisation	16-bit Quantisation	8-bit Quantisation
Memory: 1.92GB	Memory: 1.95GB	Memory: 1.02GB	Memory: 568MB

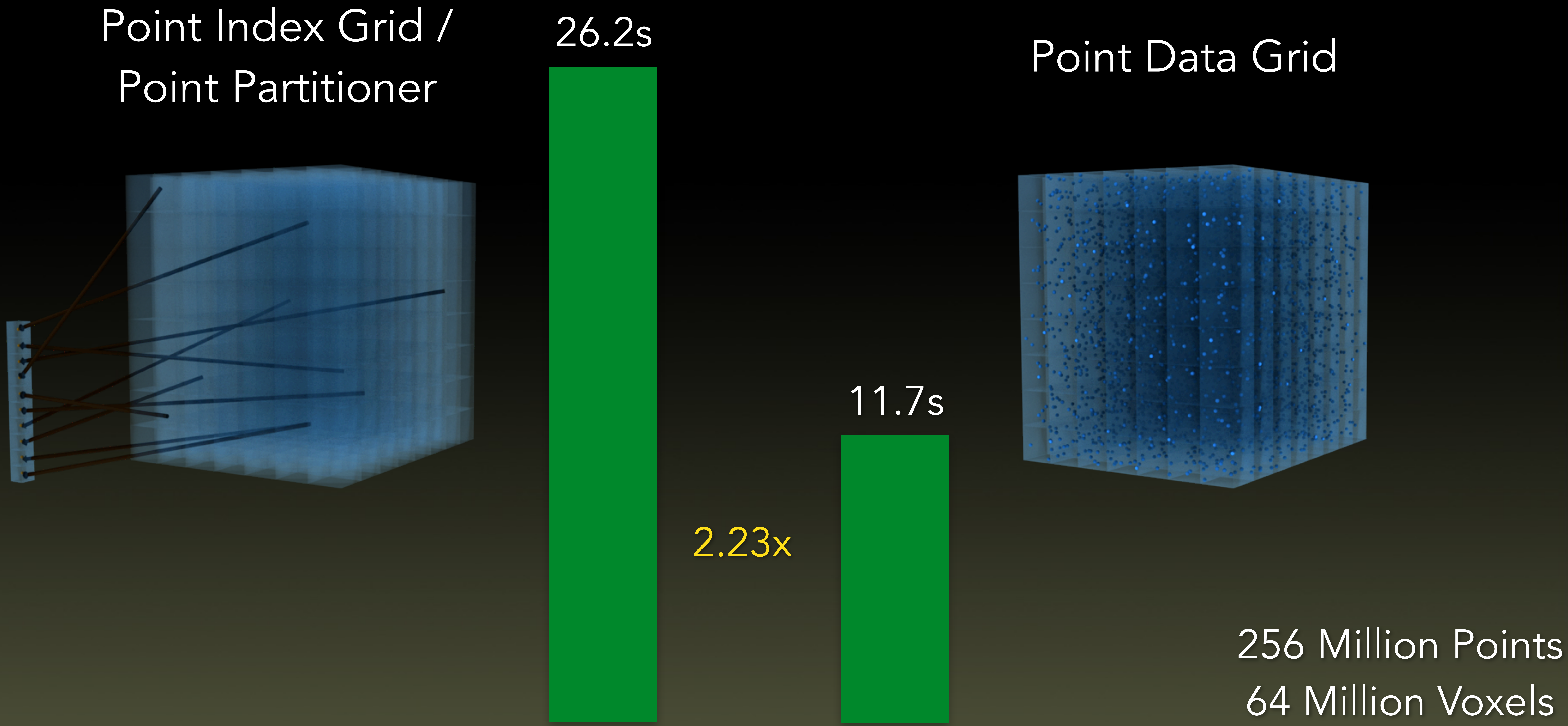
OpenVDB



double negative visual effects


Gather

Gather Rasterisation Performance



In-House Dynamo Liquid Solver

double negative visual effects

Houdini

OpenVDB

Dynamo

Dynamo Nodes

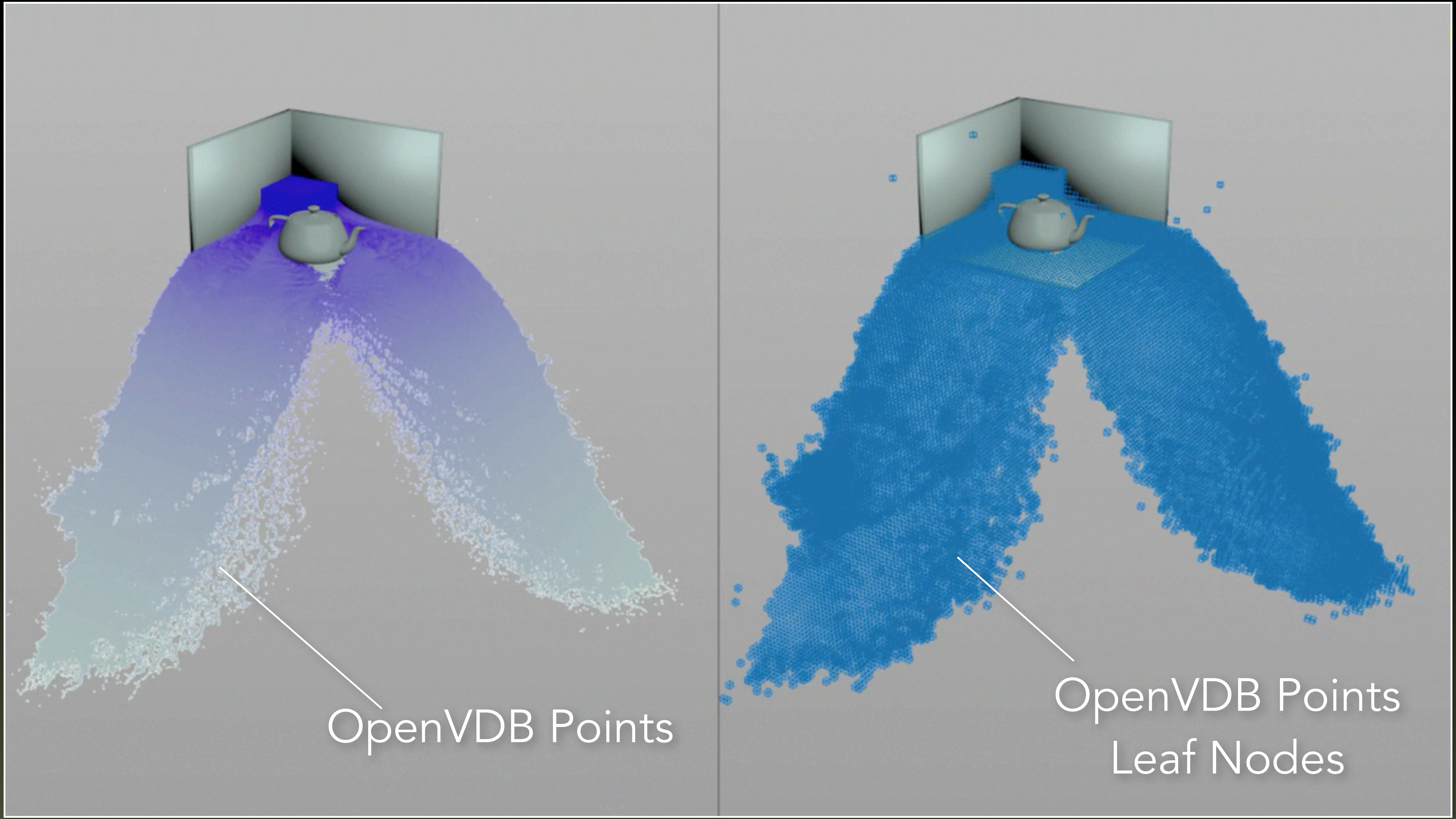
Dynamo
Data Model

Dynamo Distribution

OpenVDB
OpenVDB Points

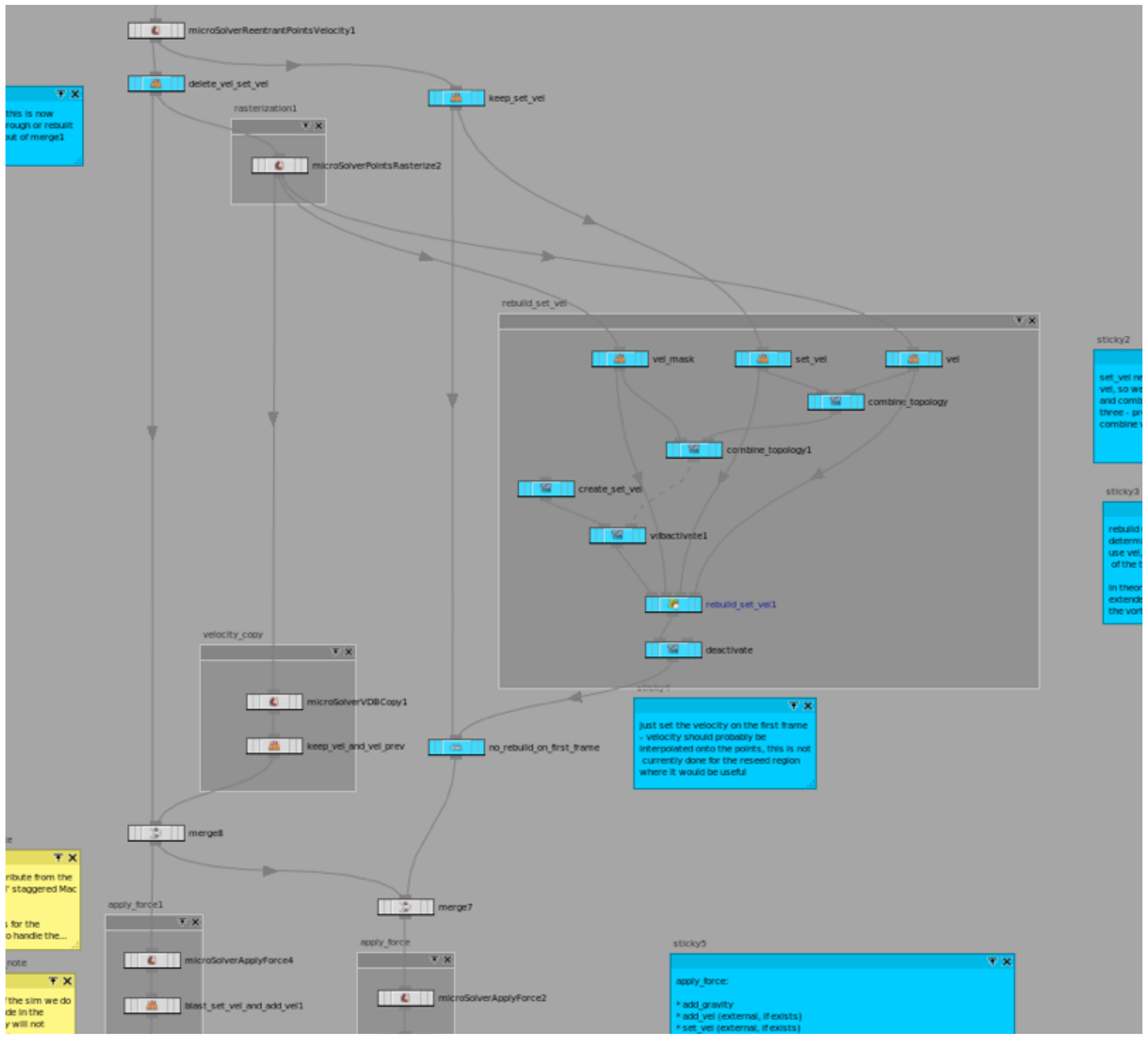
Large Data Sets All Stored using OpenVDB

Dynamo Sparse Solve

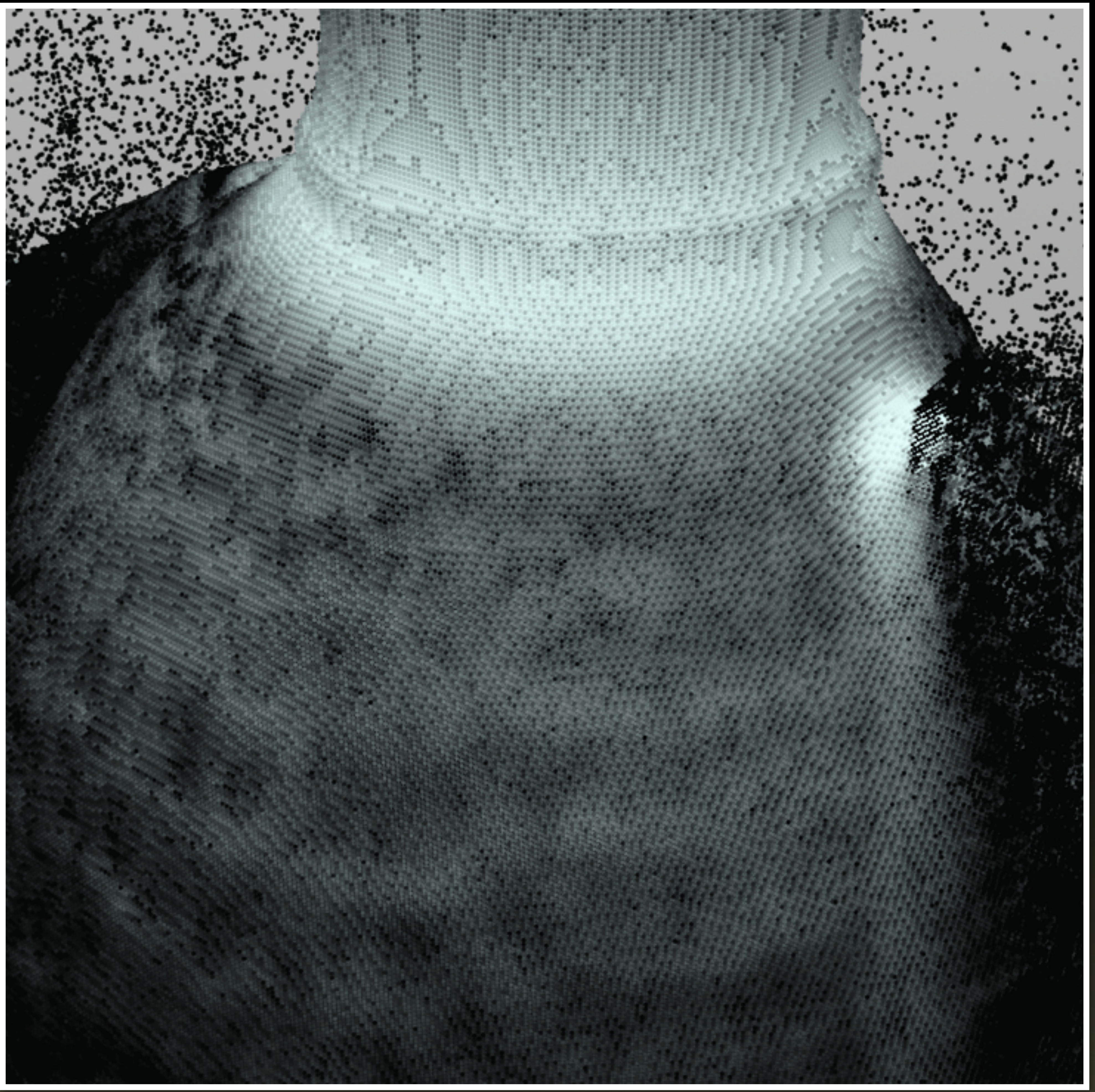


Houdini Integration

double **negative** visual effects



SOP "Micro-Solvers"



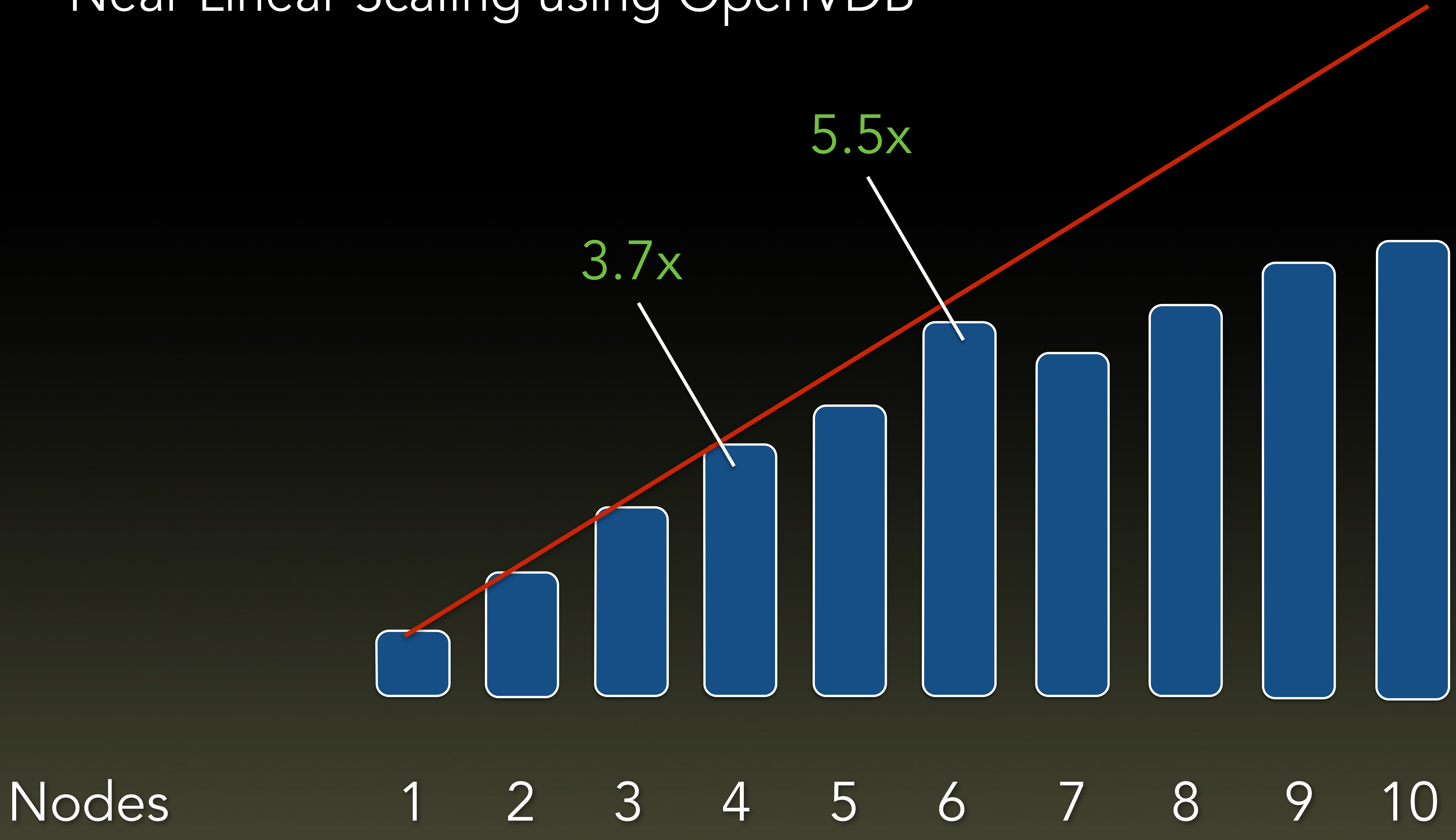
Pressure Visualisation


OpenVDB
double negative visual effects
Dynamo FLIP Liquid Simulation
1 Billion Points +

Distribution Scaling

double negative visual effects

Near-Linear Scaling using OpenVDB



OpenVDB

double negative visual effects

Point Count: 1 billion

Peak Memory: 60GB

Performance: 10-15 mins/frame

Nodes: 1 machine

Render Time: 1 hour/frame

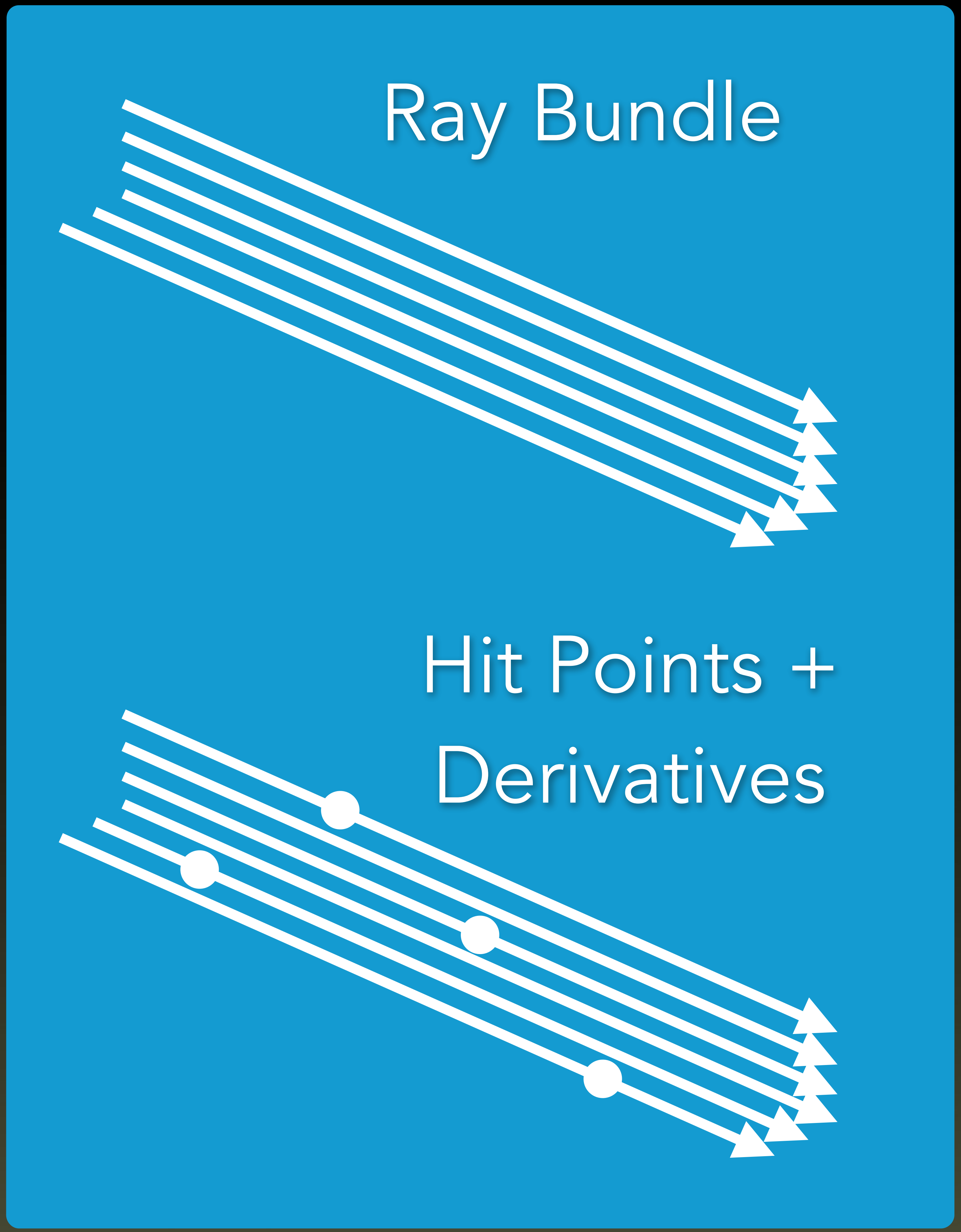
Memory: 11.0 GB

Clarisse Integration

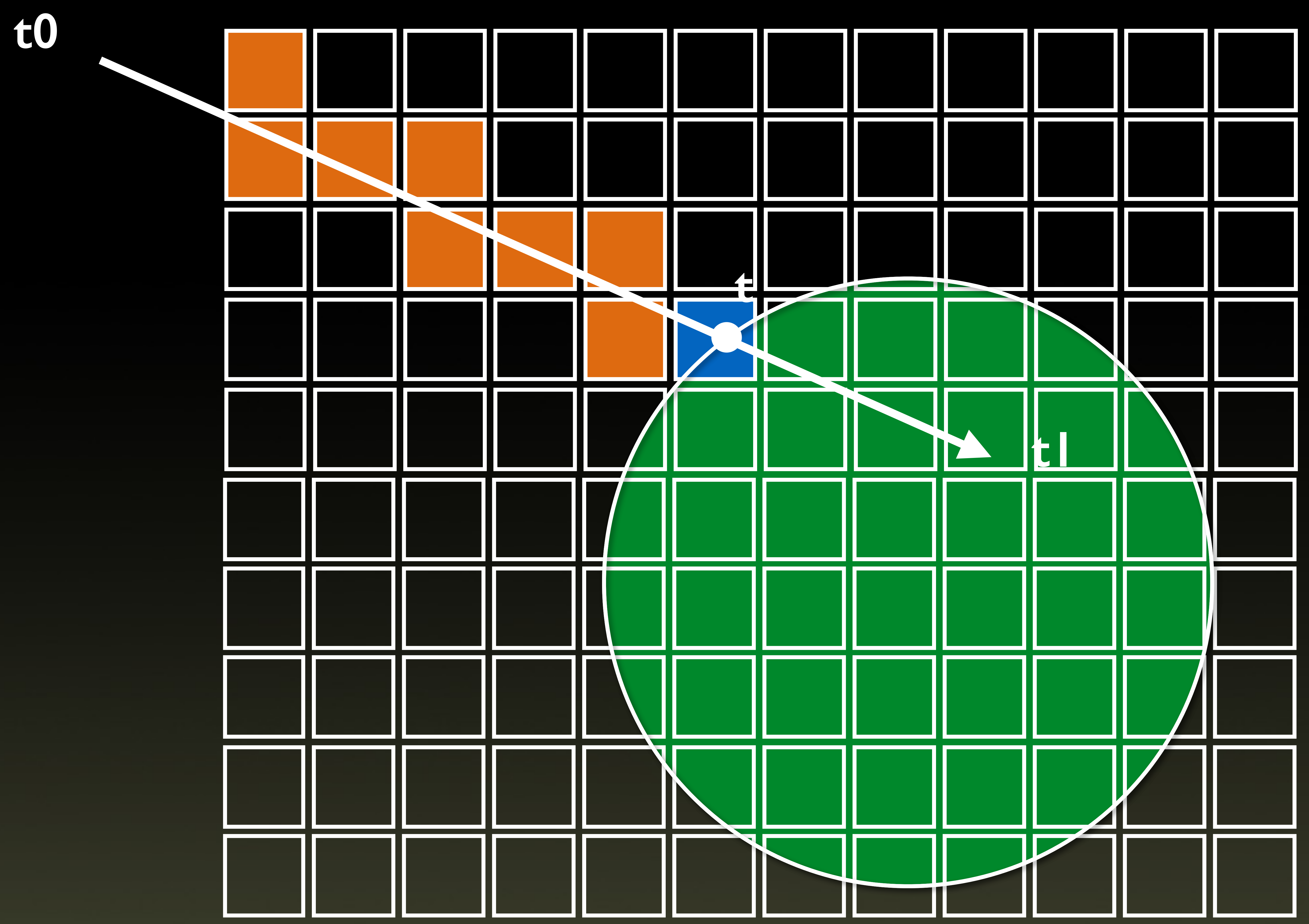
double negative visual effects

clarisse iFX

Intersection Testing



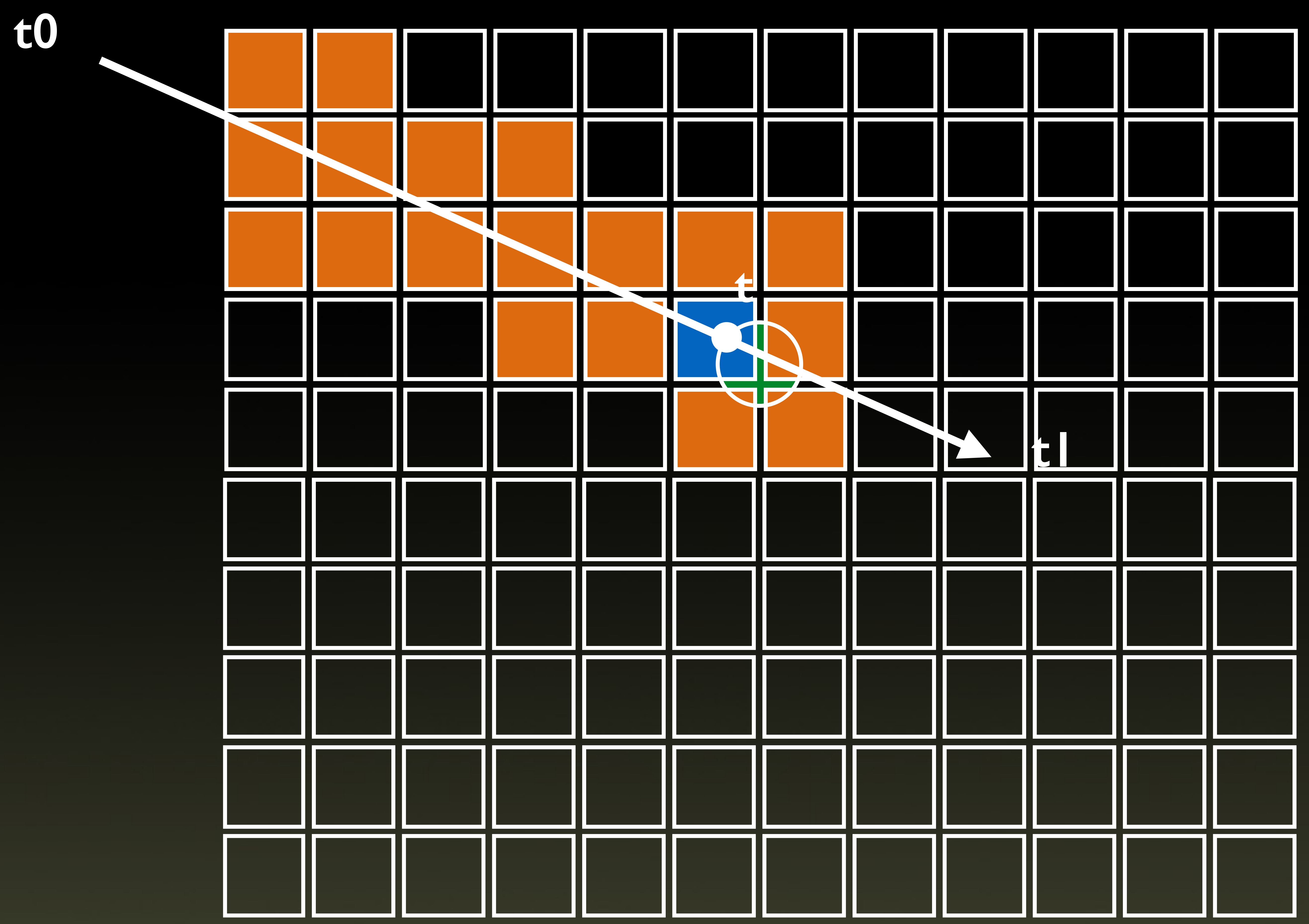
Tracing Rays - DDA for Level Sets



Intersections:
9 voxels

Tracing Rays - DDA for Points

double negative visual effects



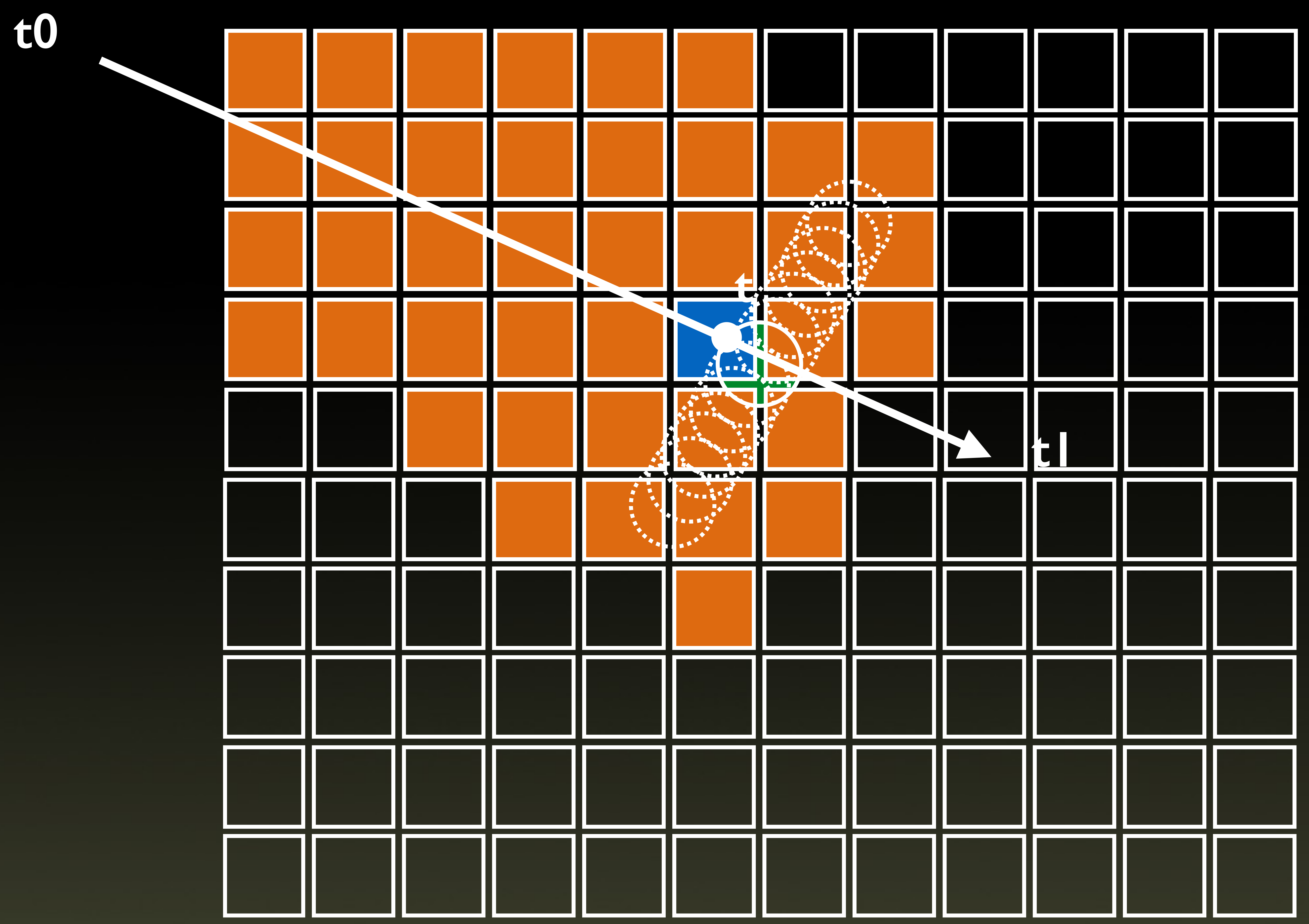
Intersections:
19 voxels

More Intersections
due to Point Radius!

6 Billion Points (position only)
Memory Usage: 17.84GB

Tracing Rays - Motion Blur

double negative visual effects



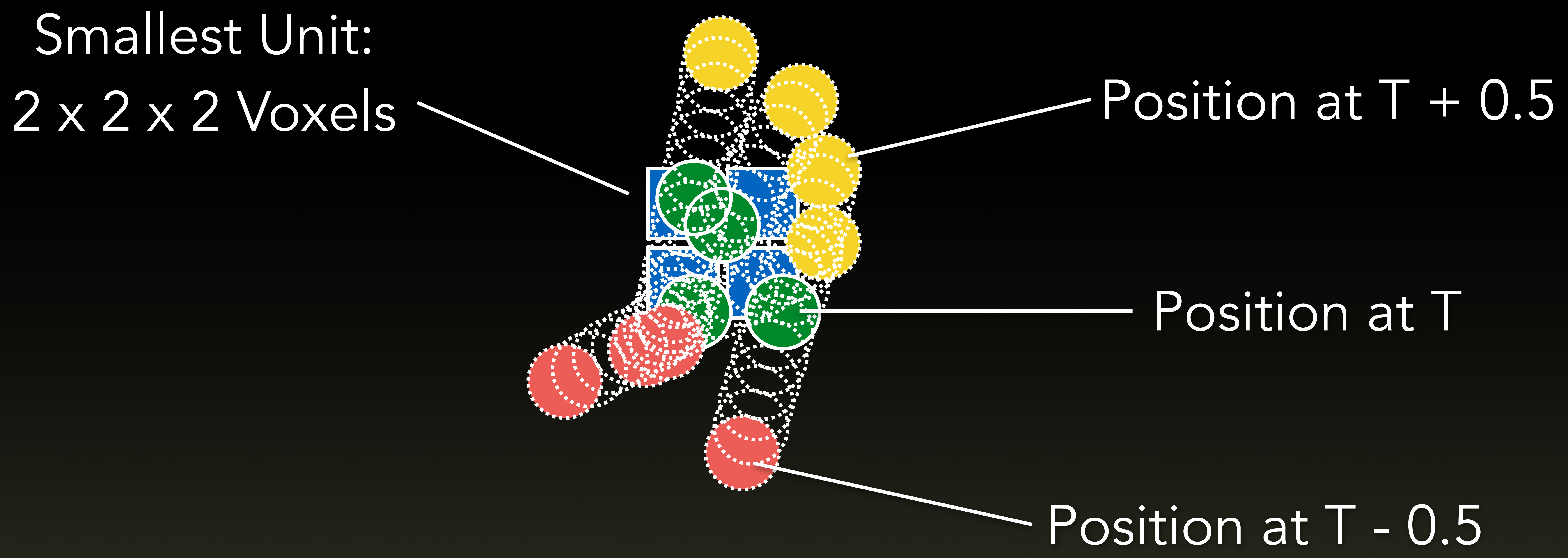
Intersections:

40 voxels

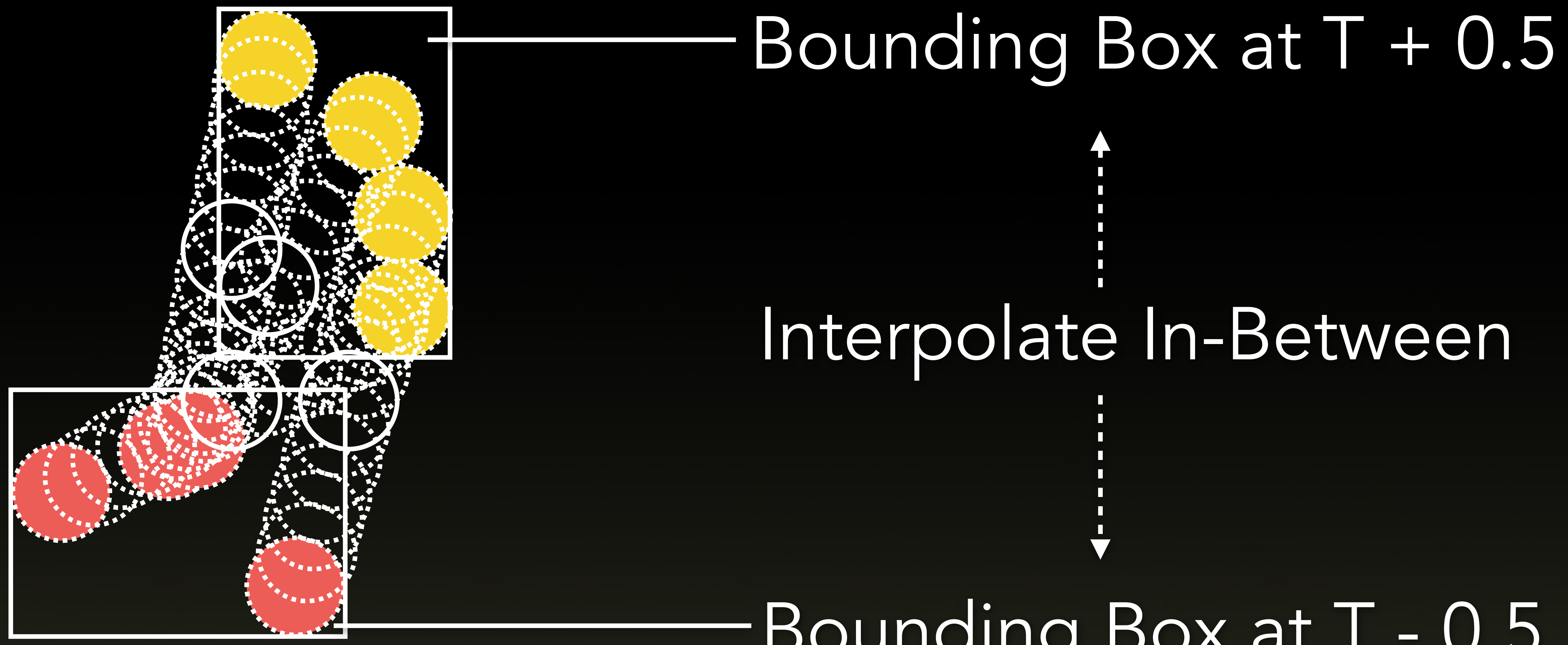
Even More Intersections to Introduce Motion Blur!

BVH Structure

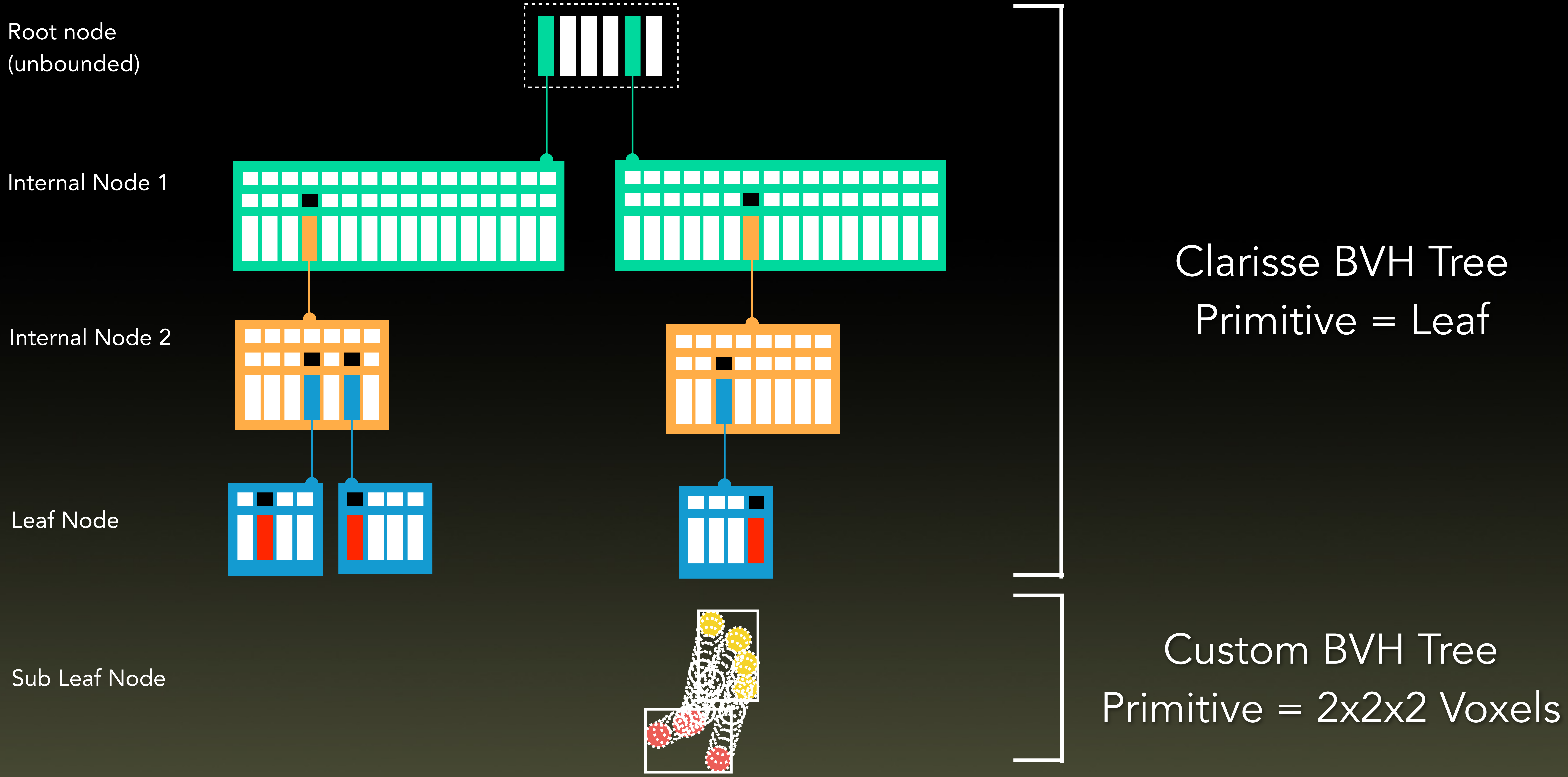
double negative visual effects



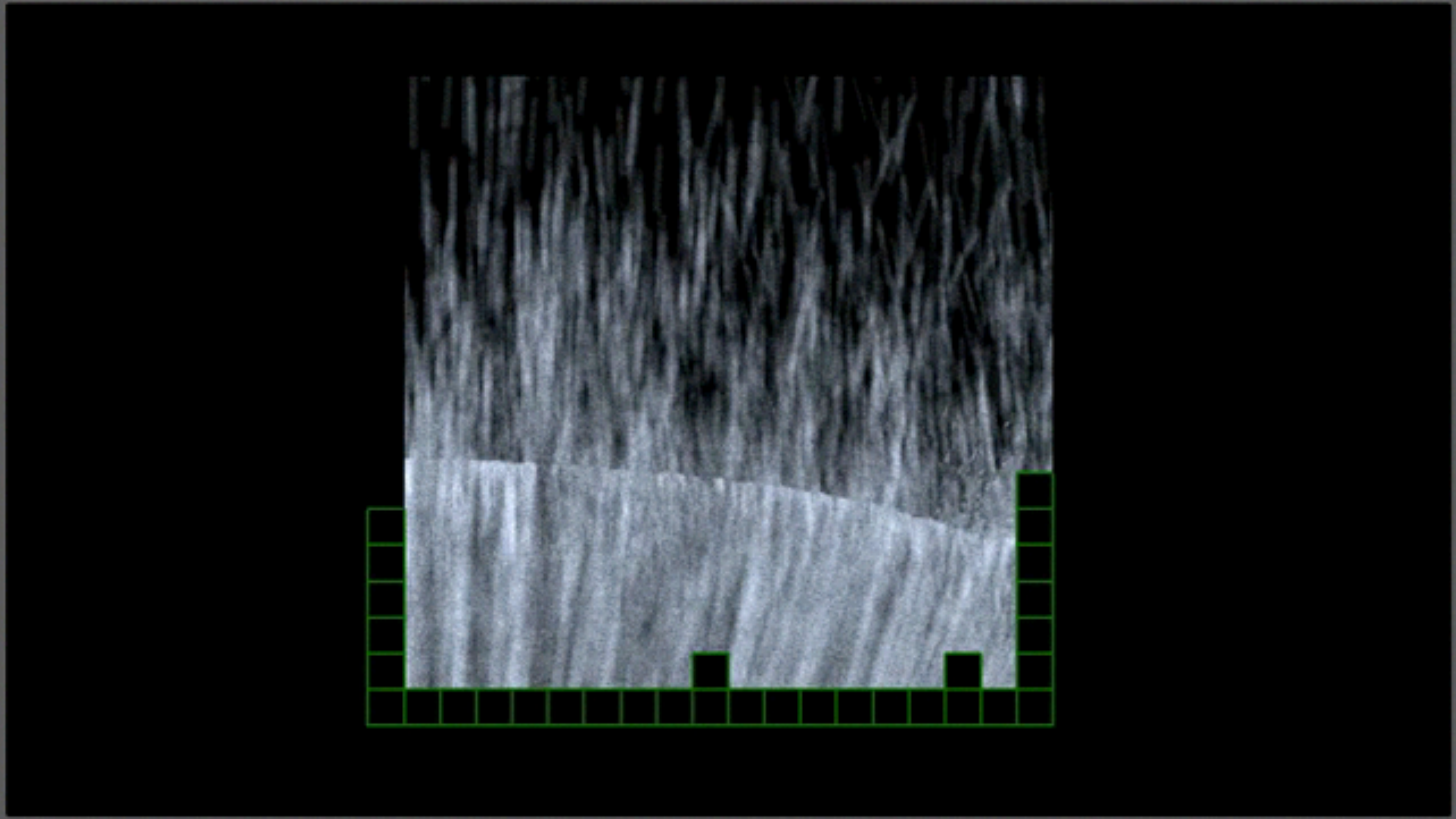
BVH Structure



BVH Structure



Interactive Speed Raytracing 1 Billion Points in Clarisse!



Memory Footprint:
10.4GB (VDB Points)
20% (VDB Grid)
6.4MB (BVH Tree)

Position (16-bit)
Velocity (32-bit + 16-bit)

720p with 6 anti-alias samples
Renders in 200s

Optimum Voxel Size

Voxel Size: 0.25	Voxel Size: 0.05	Voxel Size: 0.01
Leaves: 583	Leaves: 17,200	Leaves: 789,000
Memory: 956 MB	Memory: 999 MB	Memory: 2,829 MB

Slow Performance ← Low Memory, Fast Performance → High Memory

OpenVDB

Open-Source

double negative visual effects

openvdb::points::initialize()

OpenVDB API

AttributeArray
PointDataLeaf
Serialisation
AttributeSet
PointConversion

Houdini 13+

OpenVDB Houdini

OpenVDB Points SOP
Viewport Visualisation

Due to be announced on OpenVDB mailing list very soon!

Email me for more information (dan@dneg.com)