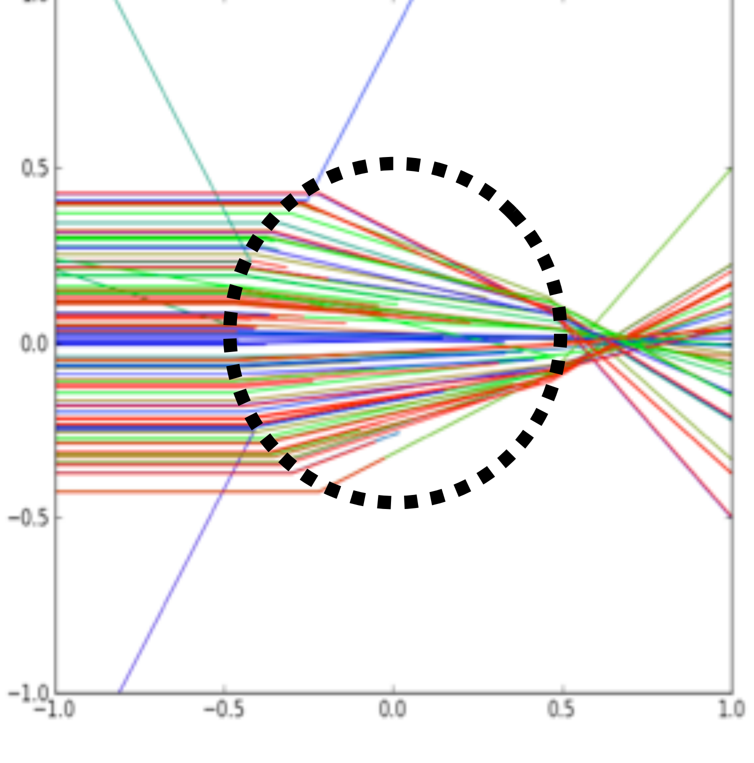
Fast Optical Simulation and Reconstruction with *Chroma*



Anthony LaTorre
Stan Seibert
University of Pennsylvania

Advances in Neutrino Technology Conference
October 12, 2011

Two Ideas (+URL)

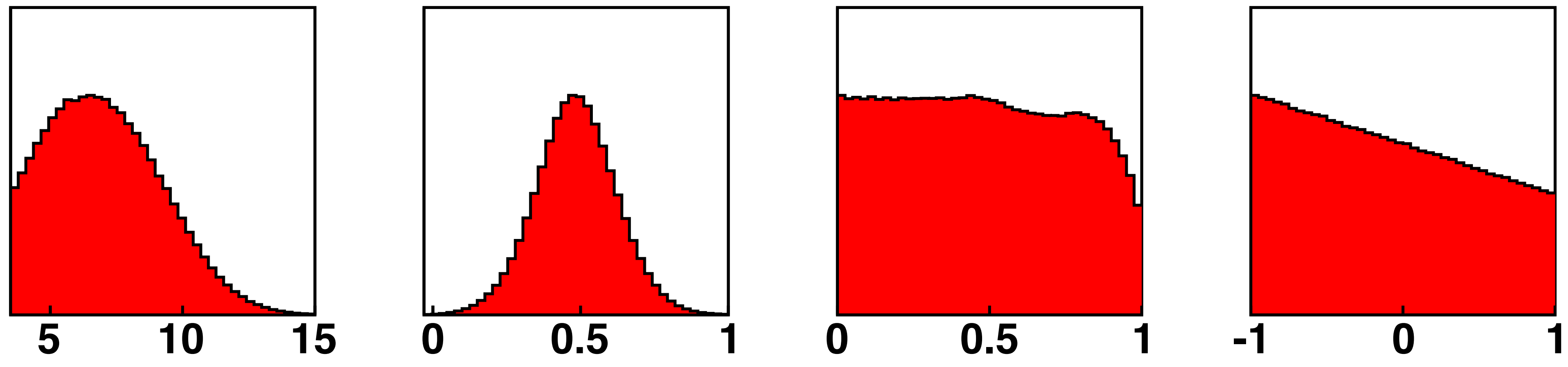
A sufficiently fast and accurate Monte Carlo is the primary ingredient to *the best maximum likelihood reconstruction you can make.*

We have the technology to do Monte Carlo photon transport in our detectors *hundreds of times faster than GEANT4.*

http://chroma.bitbucket.org

Maximum Likelihood Parameter Estimation

- We use the ML method all the time. Very effective and asymptotically optimal.
- Hard part: Creating the PDFs for all of your observables.



Maximum Likelihood Parameter Estimation

- Typical technique:
  - Cut all the hard to model / "low" information data. (Ex: late scattered photons)
  - Make analytic approximations to detector geometry and physics processes.
  - Integrate over internal degrees of freedom.

$$P_i^S = \lambda_{\text{Norm}} \int_{\lambda_1}^{\lambda_2} \frac{d\lambda}{\lambda^2} P_1^s \times P_2^s \times P_3^s \times P_4^s \times P_5^s \times P_6^s$$

$$\rho_i^{scat} = \frac{1}{V} \int\limits_0^{2\pi} \int\limits_0^{\pi} \int\limits_0^{m(\theta,\phi)} P_i^S \sin\theta \; r^2 \; dr d\theta d\phi$$

Ideal ML Fitter

- If you have a Monte Carlo that reproduces the real detector geometry, and all the physics, *why not use it?* *(Answer: speed)*
- In the limit of infinite CPU processing power, for each point in parameter space you could produce a PDF for every observable of interest.
- Very natural to include scattered light, realistic geometry effects, DAQ, etc.
- Easy to track changing detector conditions.
- If your MC is accurate, then *this is the best you could possibly do.*

An Analogy

- "Analytic" reconstruction algorithms typical work like a perturbative expansion over the space of photon histories.

  Finite computing resources (and physicist time) force you to truncate the series expansion and accept some degree of *bias* in the PDFs.
- Monte Carlo based reconstruction is a non-perturbative approach, analogous to something like lattice QCD.

  Finite computing resources force you to limit your statistics and accept some degree of *variance* in the PDFs.

Not a new idea...

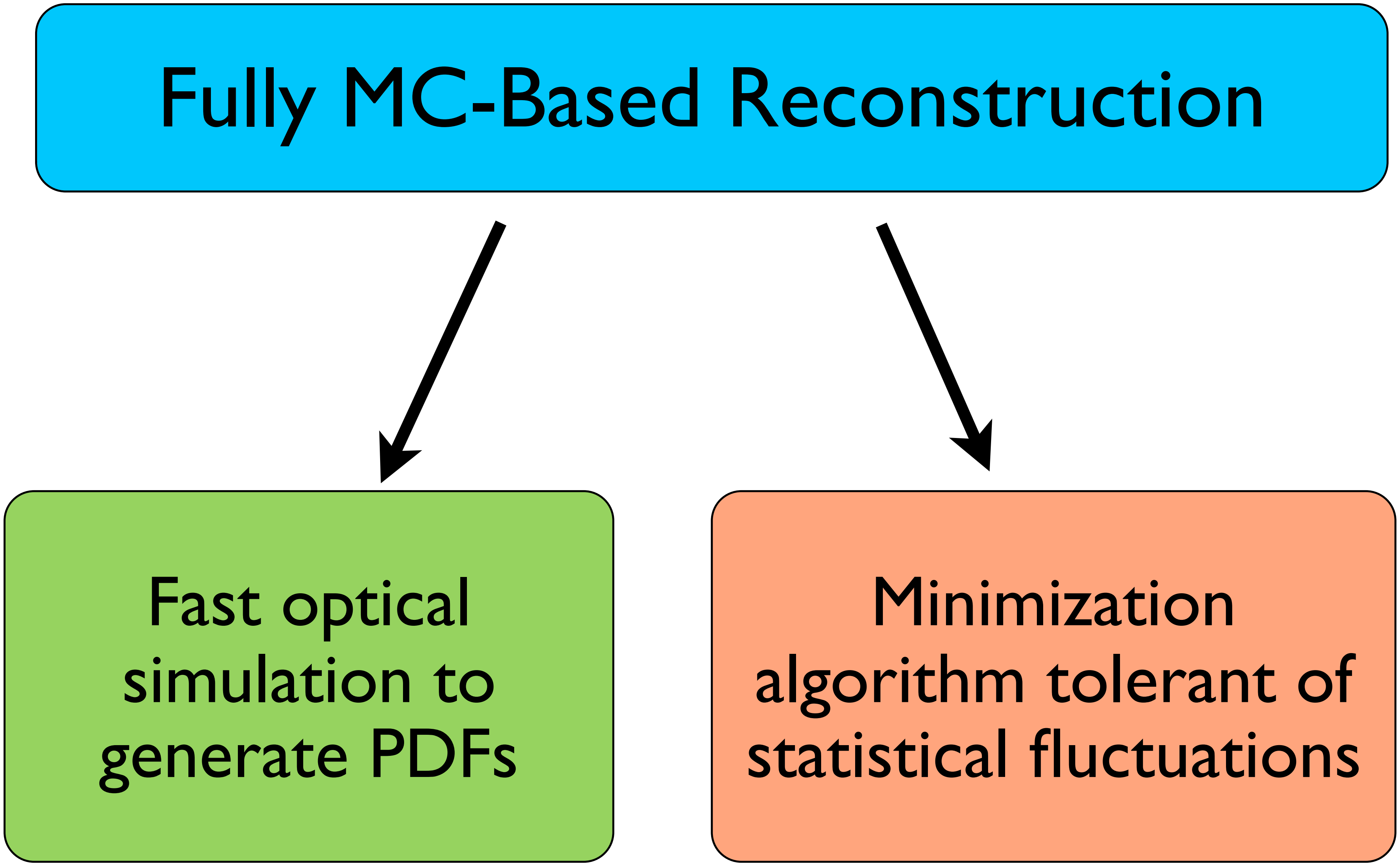
- Many experiments use Monte Carlo to create static tables for reconstruction.
- Ex: SNO and MiniCLEAN have had success with hybrid MC-analytic fitters that use fragments of the Monte Carlo simulation "on the fly" to integrate over some parts of the detector response.
    - ➡ SNO treated refractive magnification of PMTs viewed through the acrylic vessel this way, as well as reflected and scattered light.

Computing power is rapidly reaching the point where we could do the entire PDF calculation in Monte Carlo!

Obstacles

- GEANT4 is far too slow, structurally resistant to parallelization/acceleration, and very hard to use outside of a full simulation.

- A Monte Carlo-derived likelihood has an intrinsic uncertainty due to Poisson fluctuations in the PDF bins. Even a small amount of statistical variation will confuse MIGRAD, and pretty much any other gradient descent fitter.


Fully MC-Based Reconstruction
Fast optical simulation to generate PDFs
Minimization algorithm tolerant of statistical fluctuations

How do we make a fast optical simulation?

How do we make a fast optical simulation?

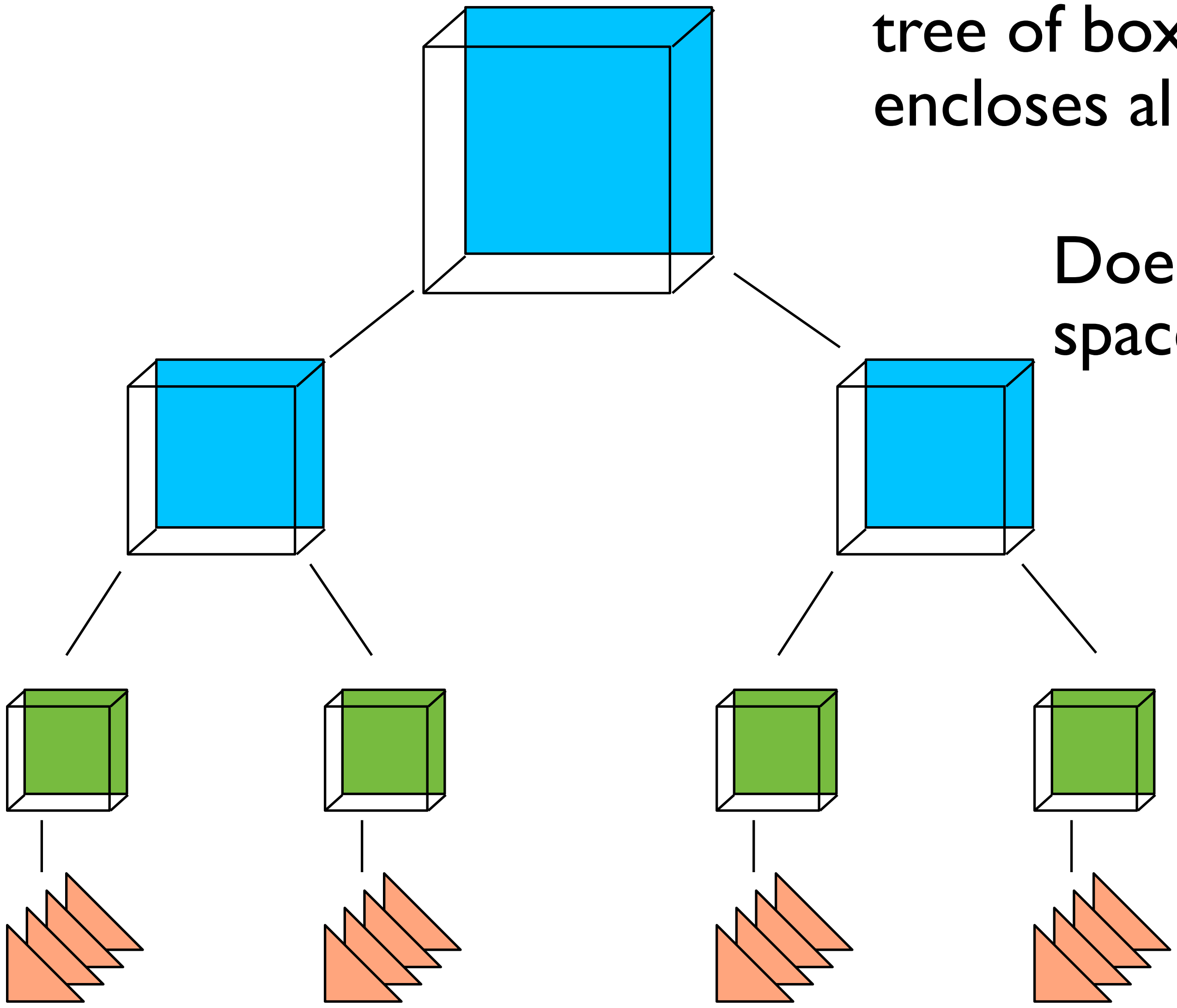
... go back and re-evaluate our assumptions.

Two Approaches to Geometry

Software tools that work with 3D objects, generally fall into one of two paradigms:

- Solid Modeling: Build objects using various 3D solid primitives and constructive solid geometry. (GEANT4)
- Surface Modeling: Build objects using a surface primitive, like a triangle, replicated and connected to form a continuous mesh. (our approach)

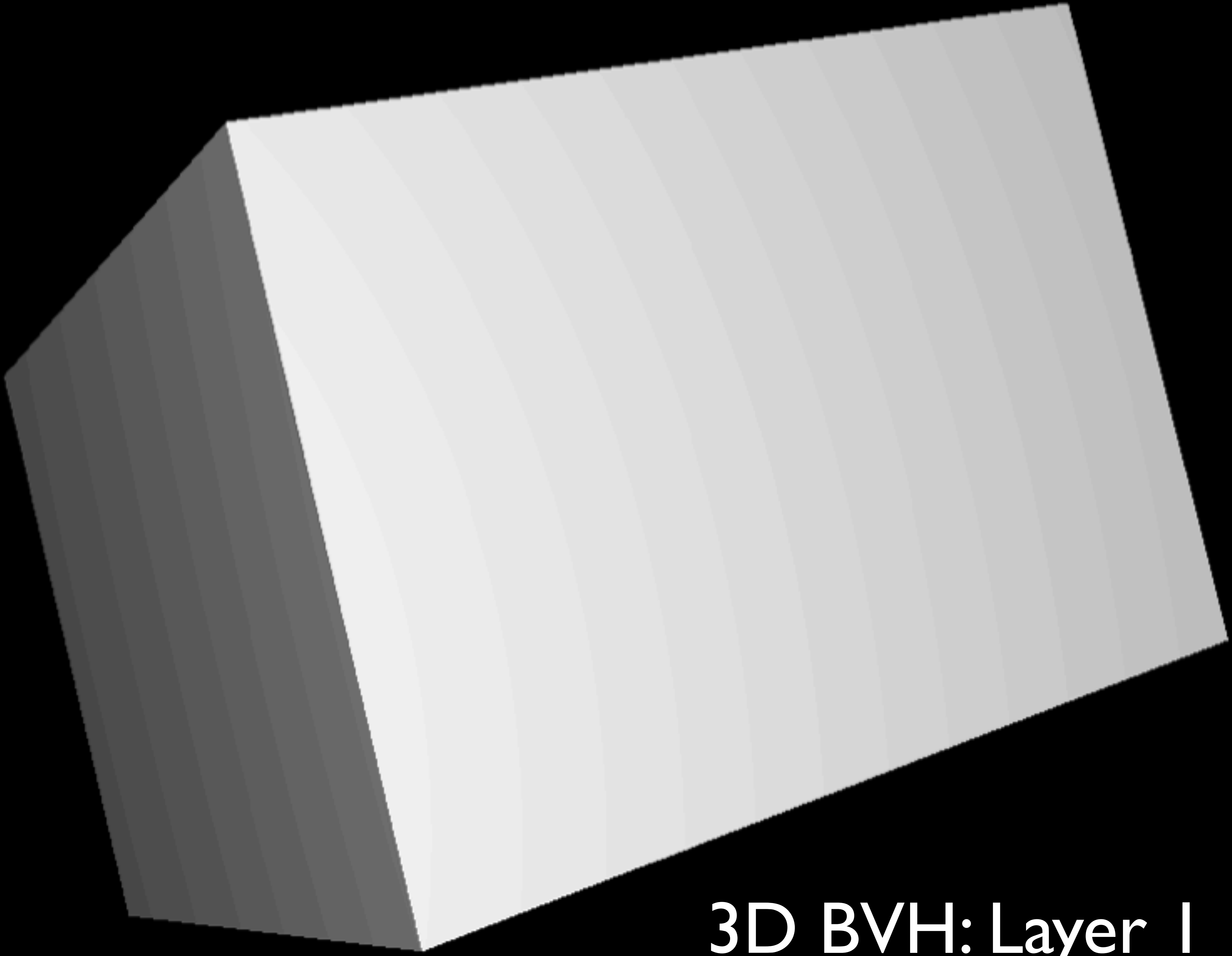
Fast intersection with a mesh



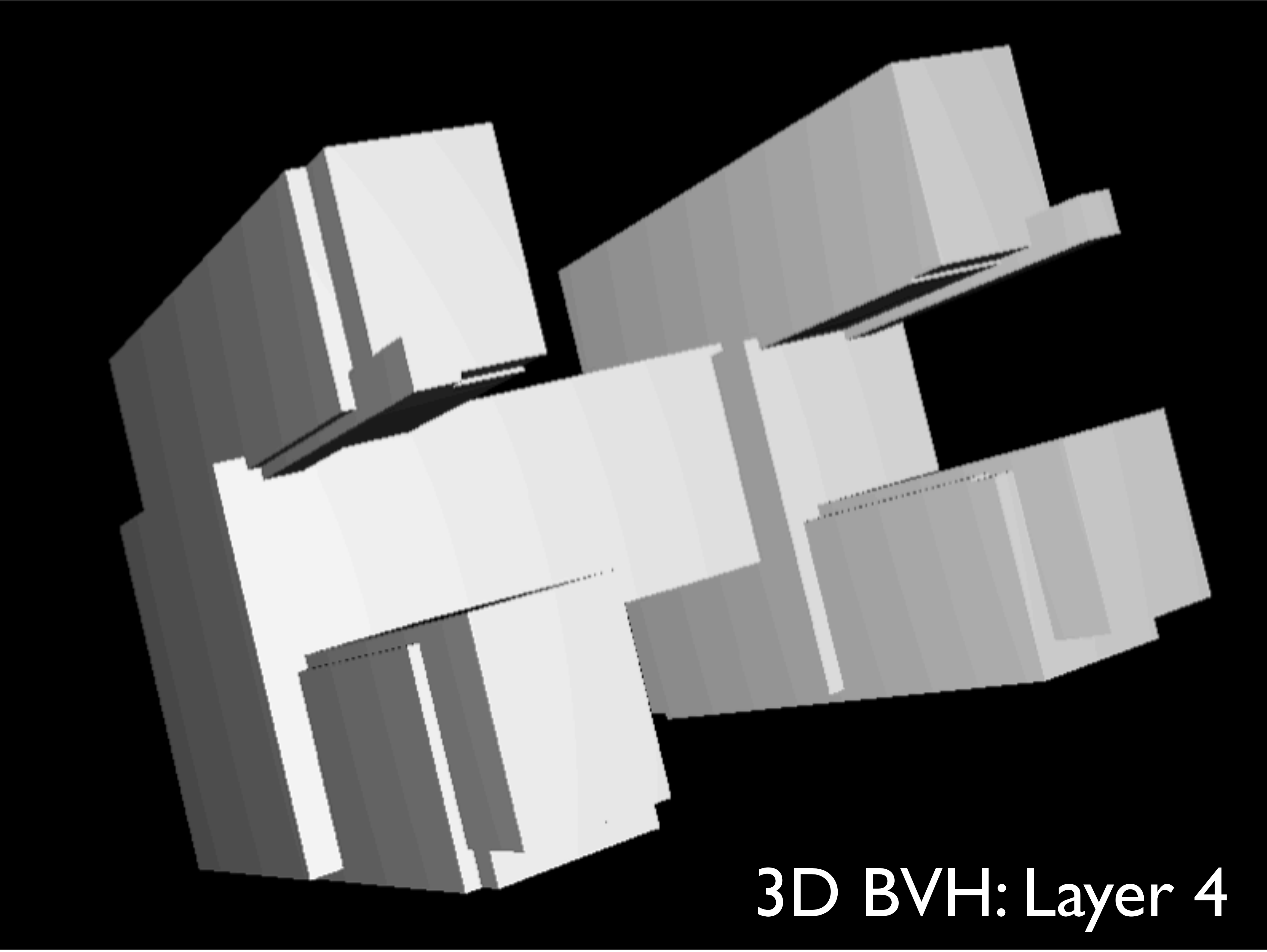
Bounding Volume Hierarchy: A tree of boxes where each node encloses all of its descendants.

Does not need to partition the space and siblings can overlap!

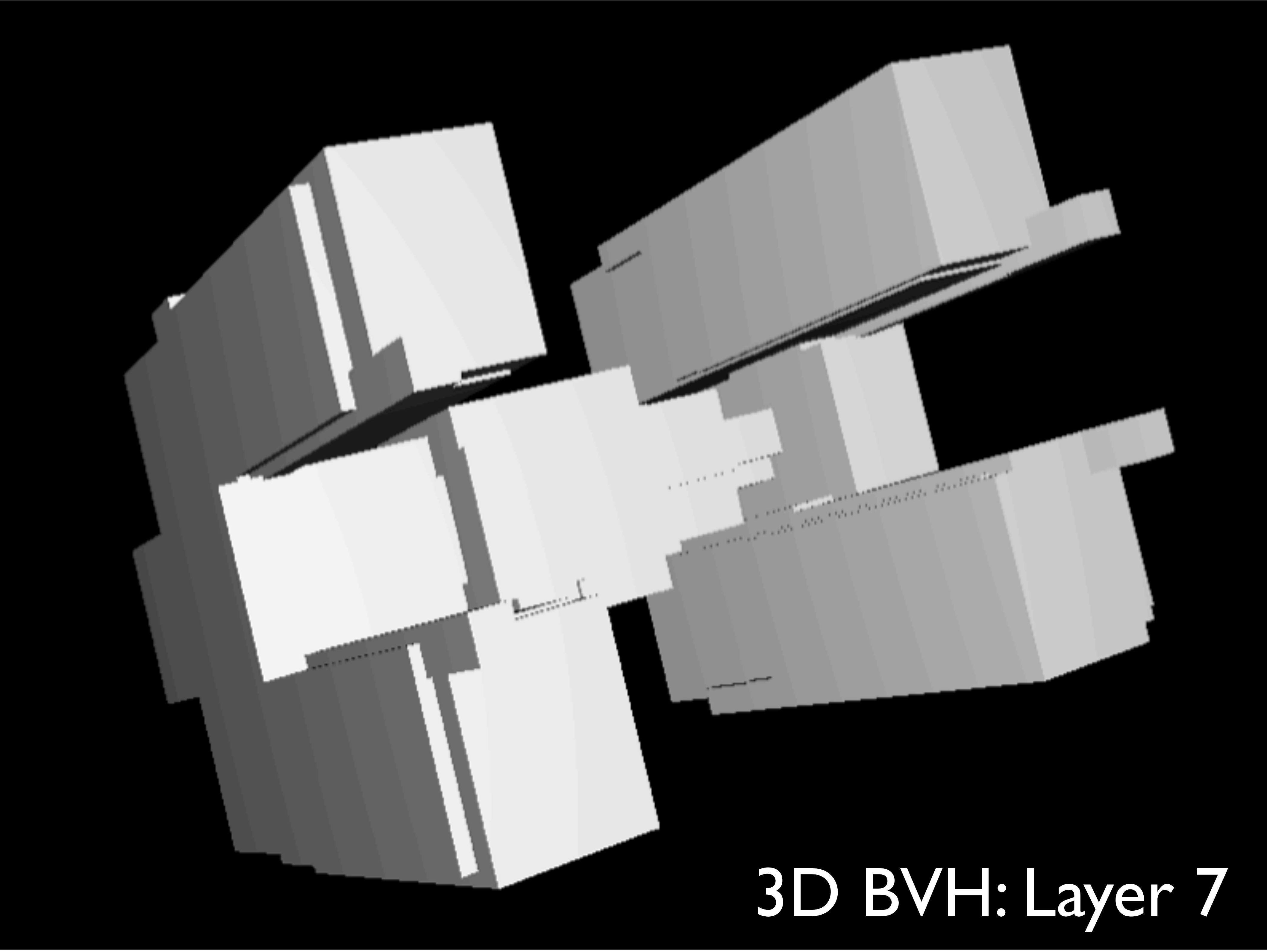
Leaf nodes contain list of triangles



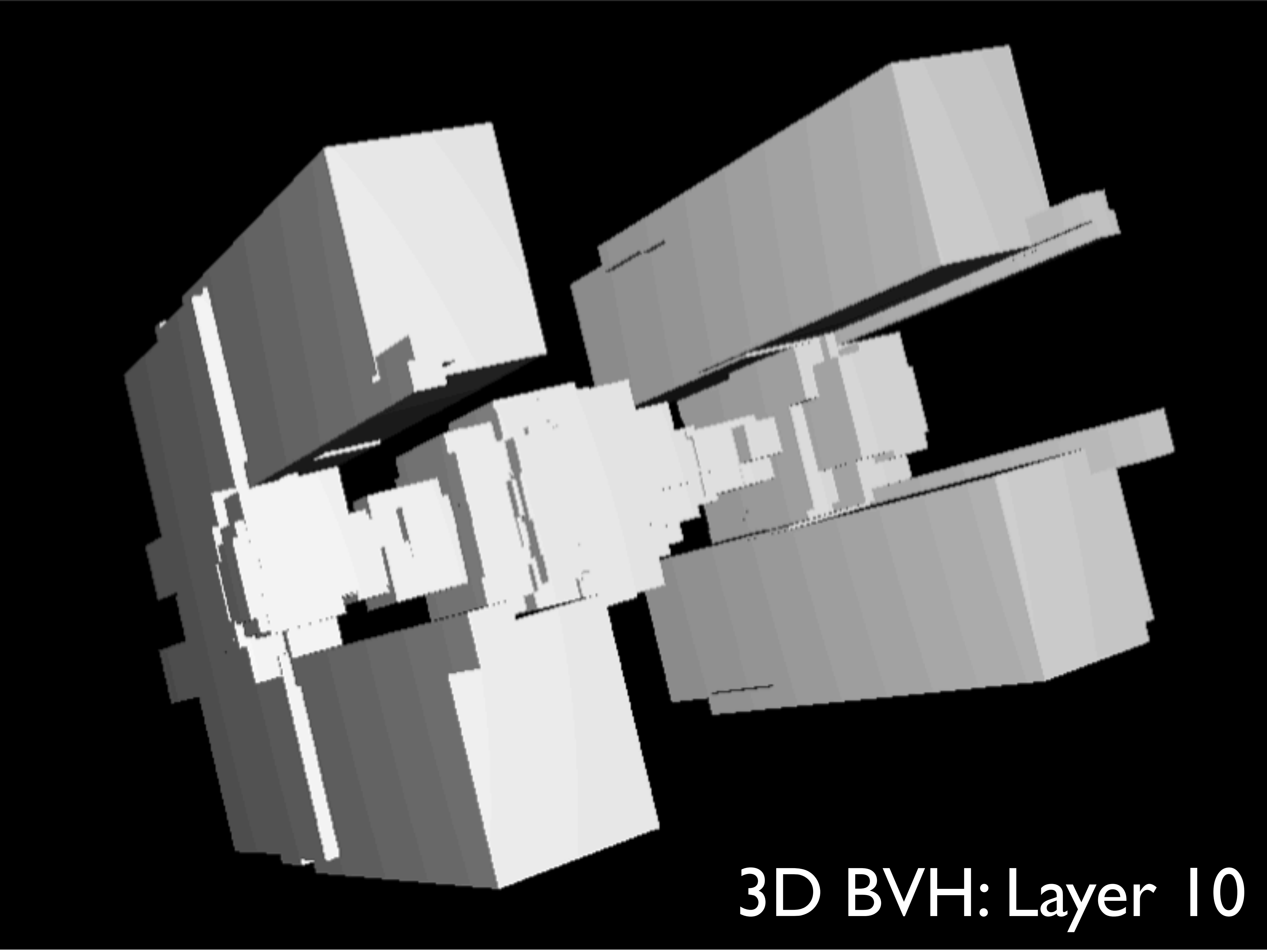
3D BVH: Layer 1



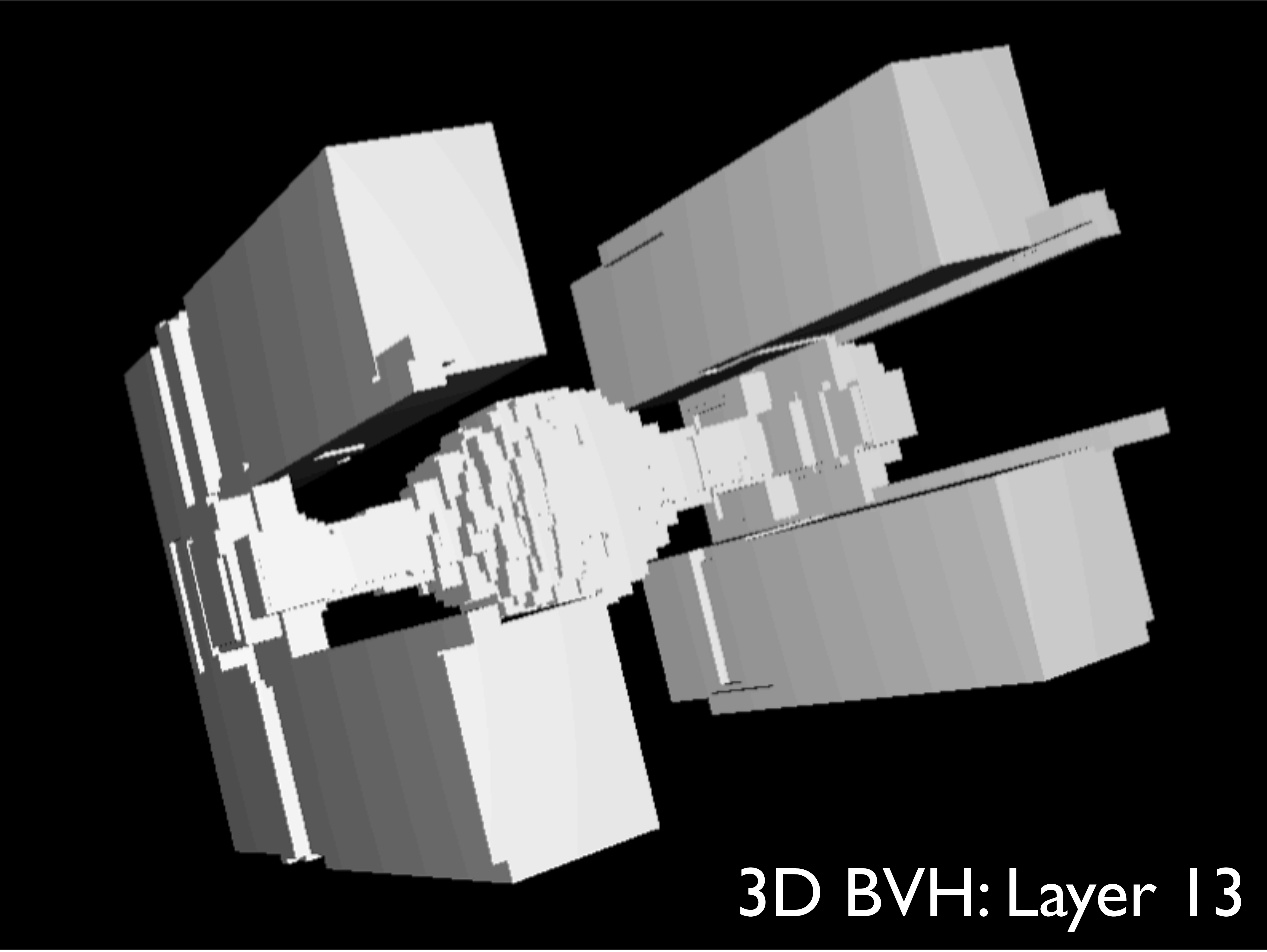
3D BVH: Layer 4



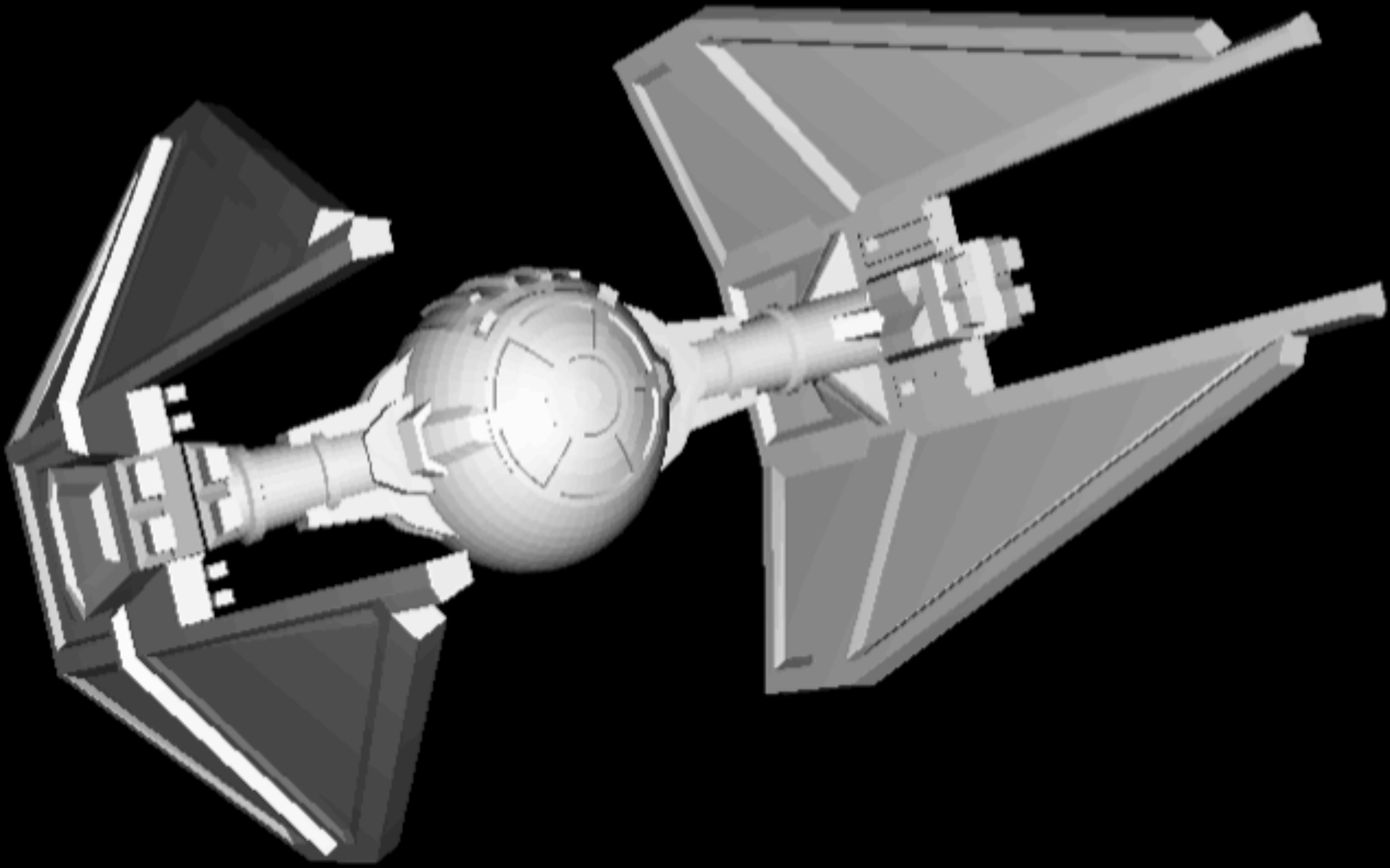
3D BVH: Layer 7



3D BVH: Layer 10



3D BVH: Layer 13

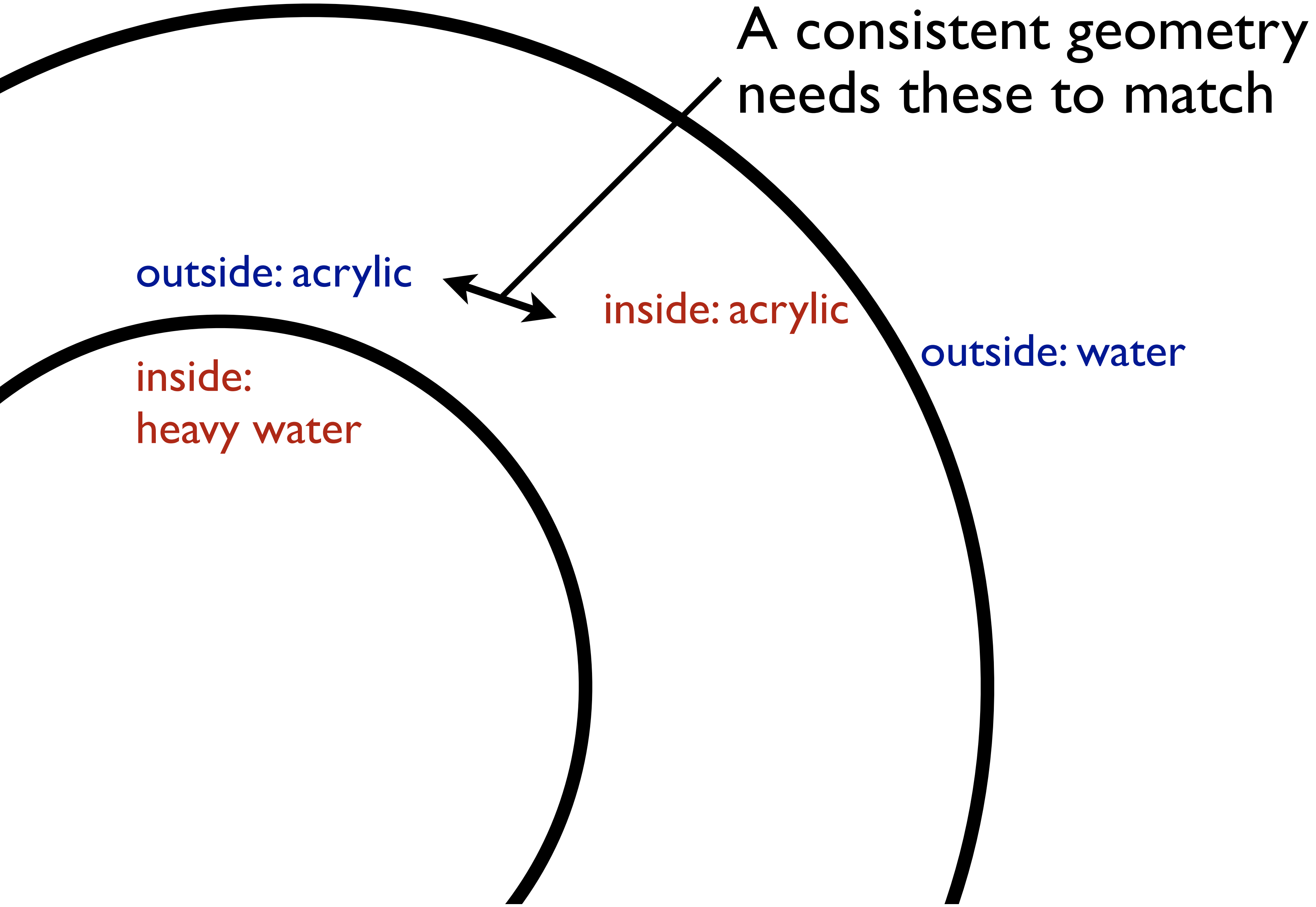


Actual Model

Why a Triangle Mesh?

- GEANT4 is slow because of the overhead of delegating geometric operations to polymorphic methods. *Abstraction prevents simplification and optimization, in this case.*
- A triangle mesh can reasonably approximate most surfaces, and is pure data. No geometry-specific code → one code path to optimize.
- Fast mesh techniques are well-studied in the literature.
- Plenty of tools for manipulating triangle meshes exported from CAD files or constructed numerically.

Surface-based Definition of a Detector



Comparison to GEANT4

- In GEANT4, the detector is constructed with a particular tree structure of mother and daughter volumes.
- GEANT4's "voxelization" technique further subdivides volumes along one axis at a time.

- In Chroma, the tree is constructed dynamically for the user in the form of a BVH. User selects "patience level" only.
- Much more aggressive than voxelization, and can actually improve performance by subdividing a solid.

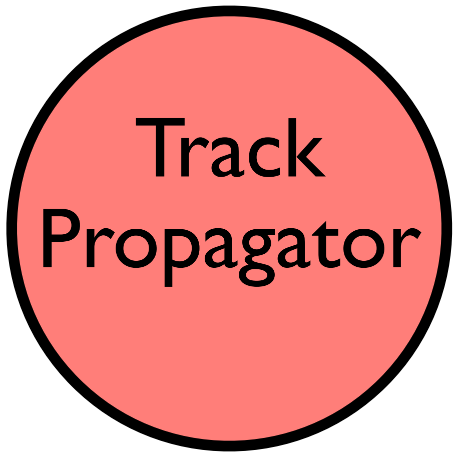
Even faster intersection with a mesh

- Each event has large numbers of photons.
- Perfect application for GPU parallelization!

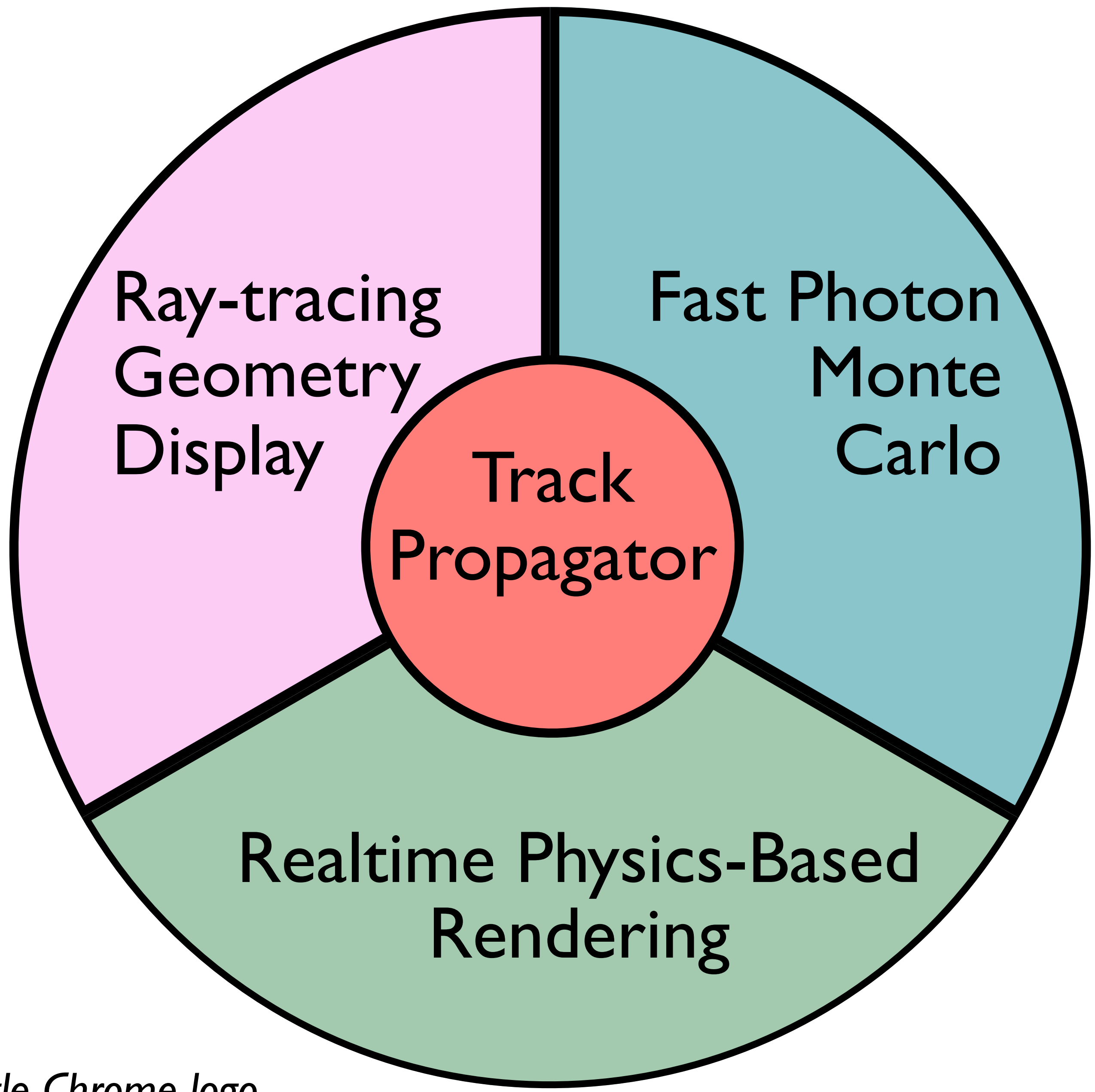
GTX 580

- 512 floating point units
- 1.5 GHz
- 3GB device memory
- 192 GB/sec memory bandwidth
- $550

The Chroma Core

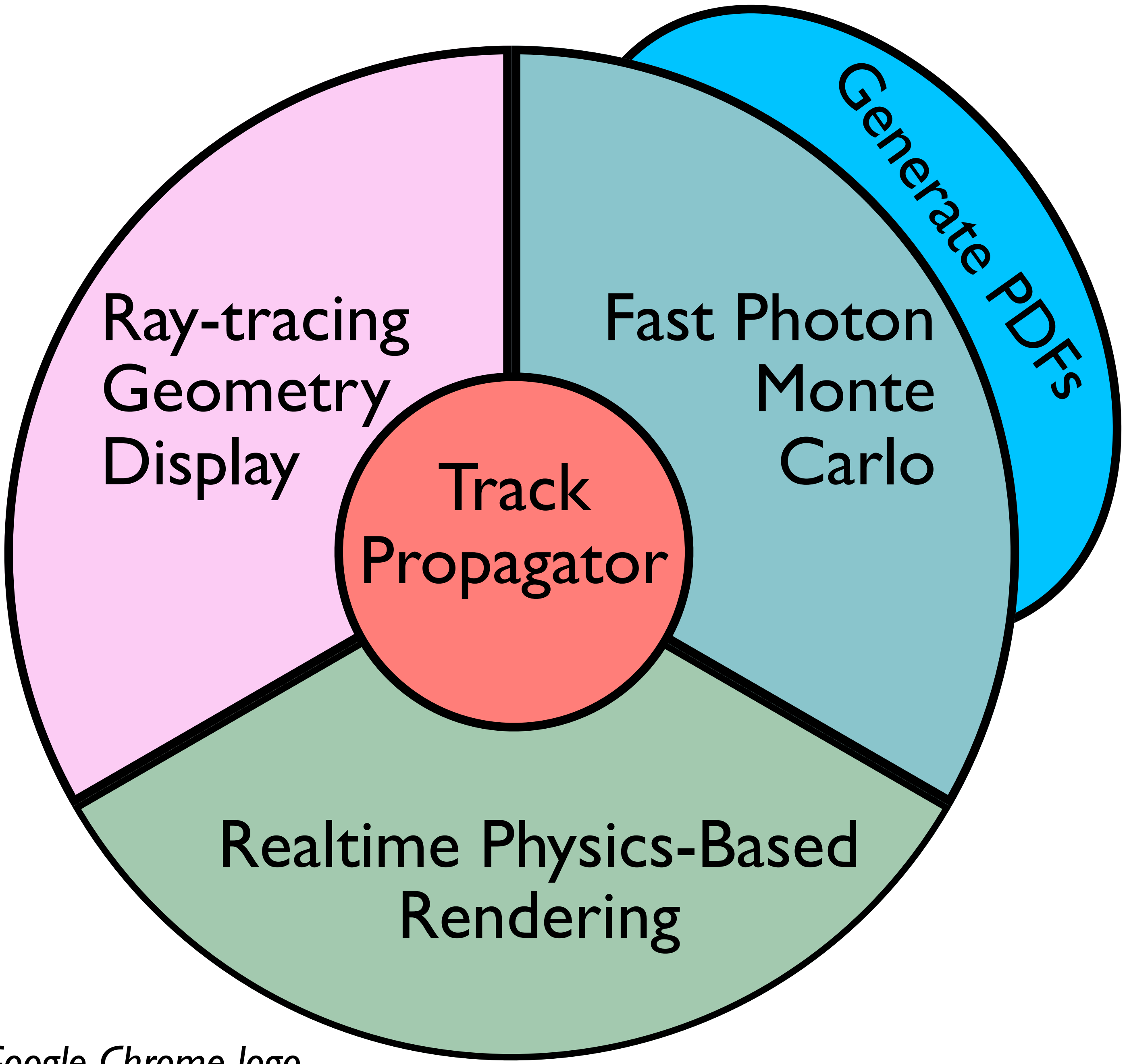


Building on the Chroma Core


Ray-tracing Geometry Display
Fast Photon Monte Carlo
Track Propagator
Realtime Physics-Based Rendering

This is not the Google Chrome logo.

Building on the Chroma Core


Ray-tracing Geometry Display
Fast Photon Monte Carlo
Generate PDFs
Track Propagator
Realtime Physics-Based Rendering

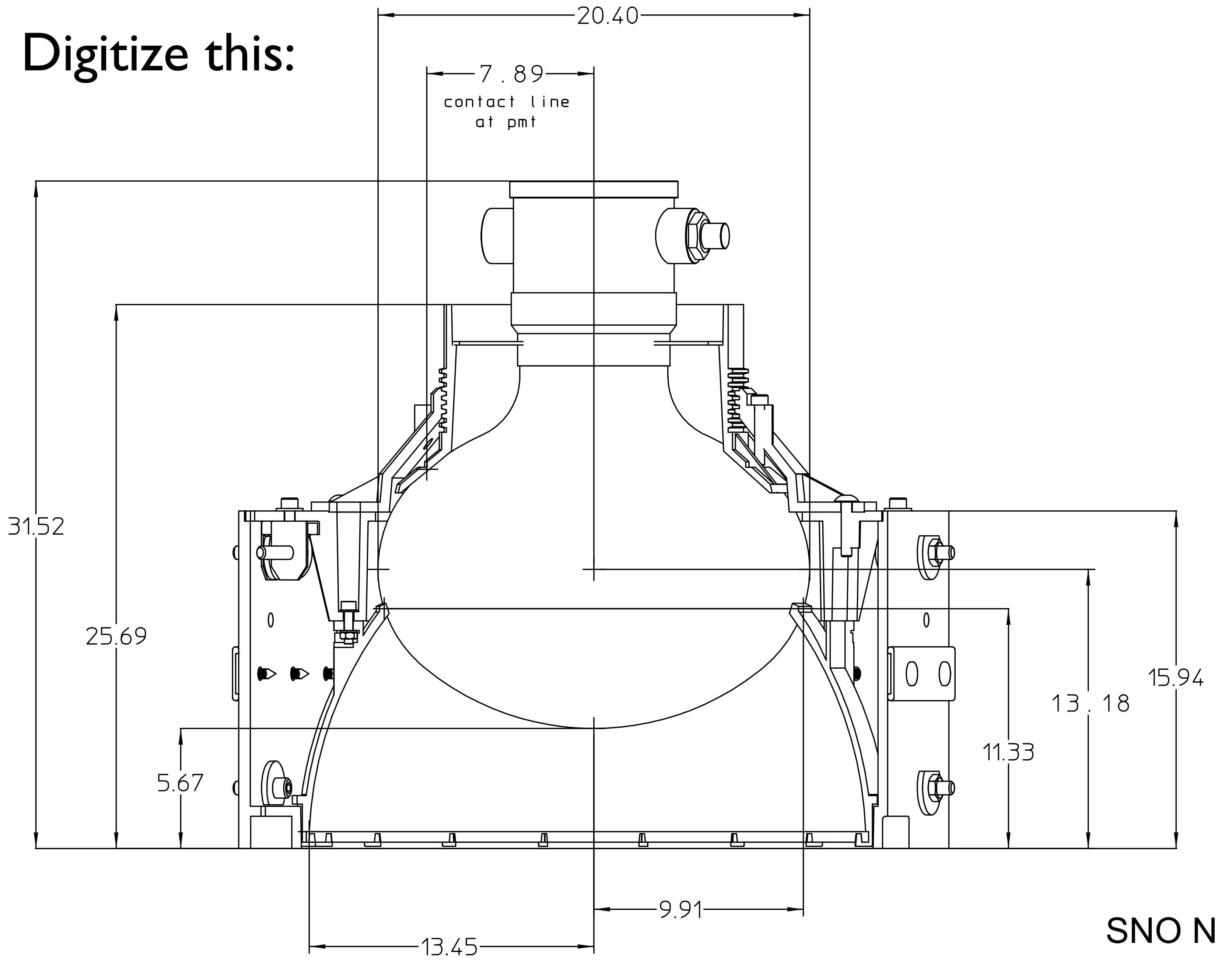
This is still not the Google Chrome logo.

A Quick Tour:

Let’s Make a Detector!

PMT Model

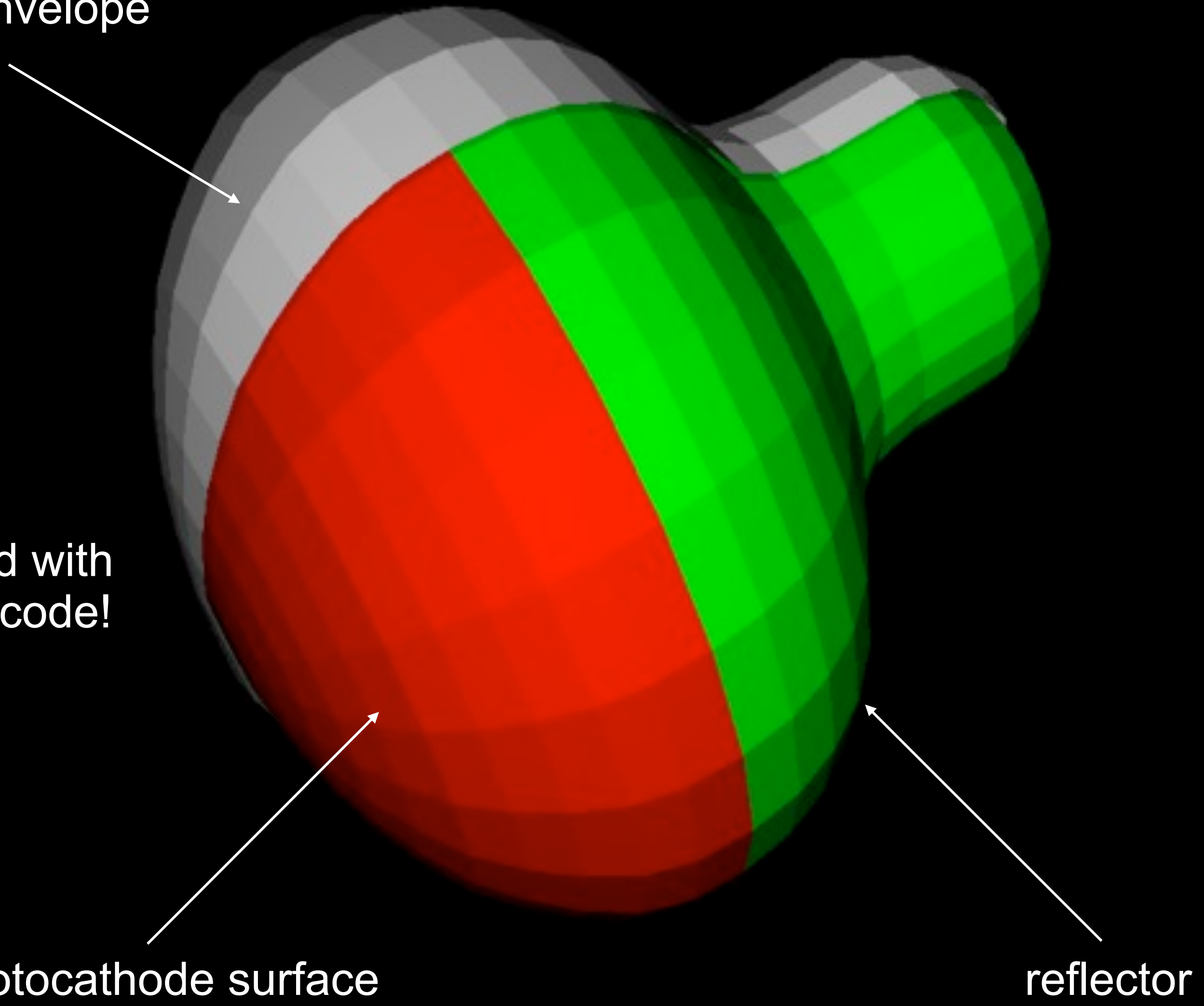
Digitize this:



SNO NIM paper

PMT Model

To get this:



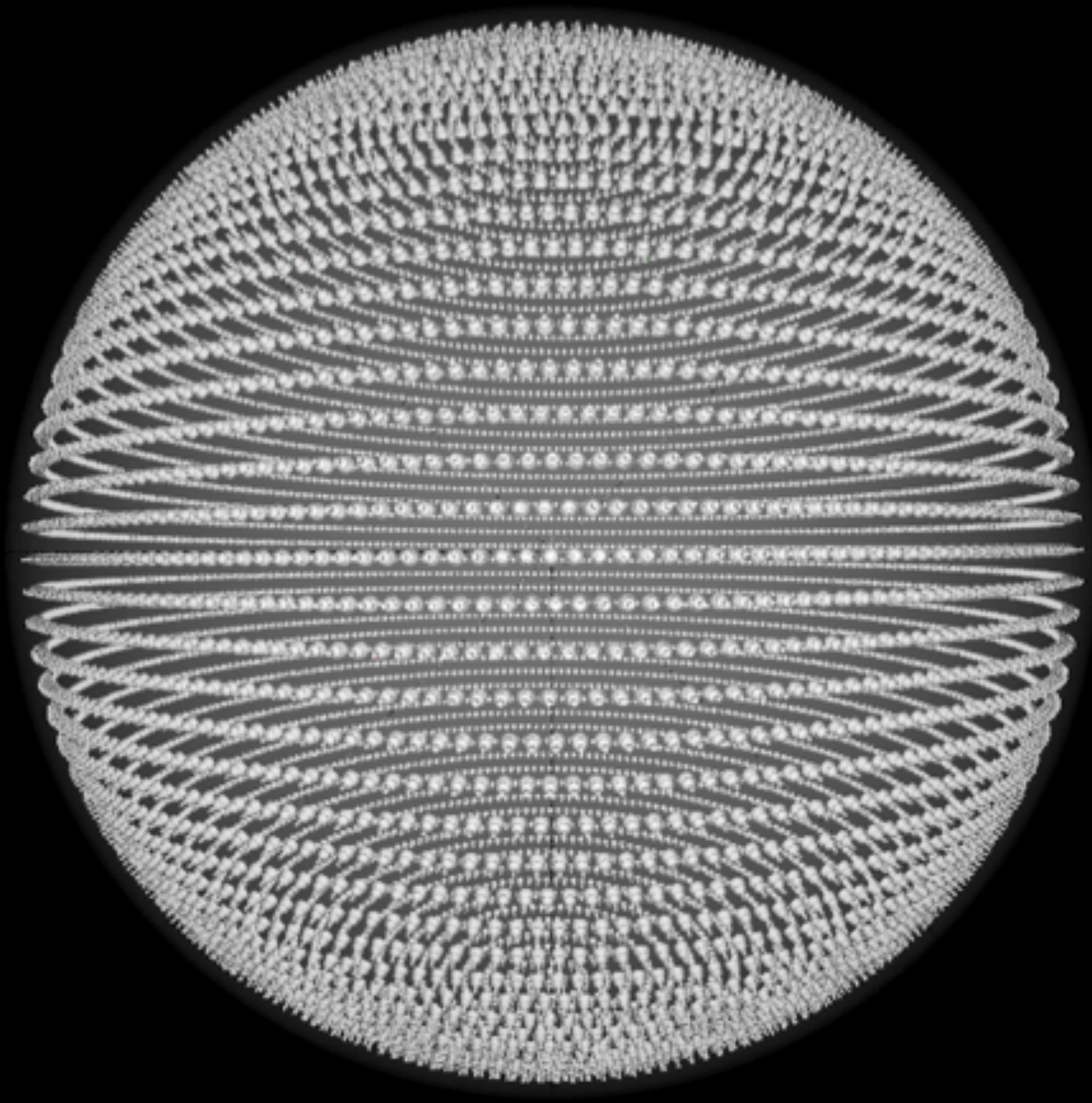
Note: This is rendered with the actual simulation code!

“Hyper-SNO” Demo Detector

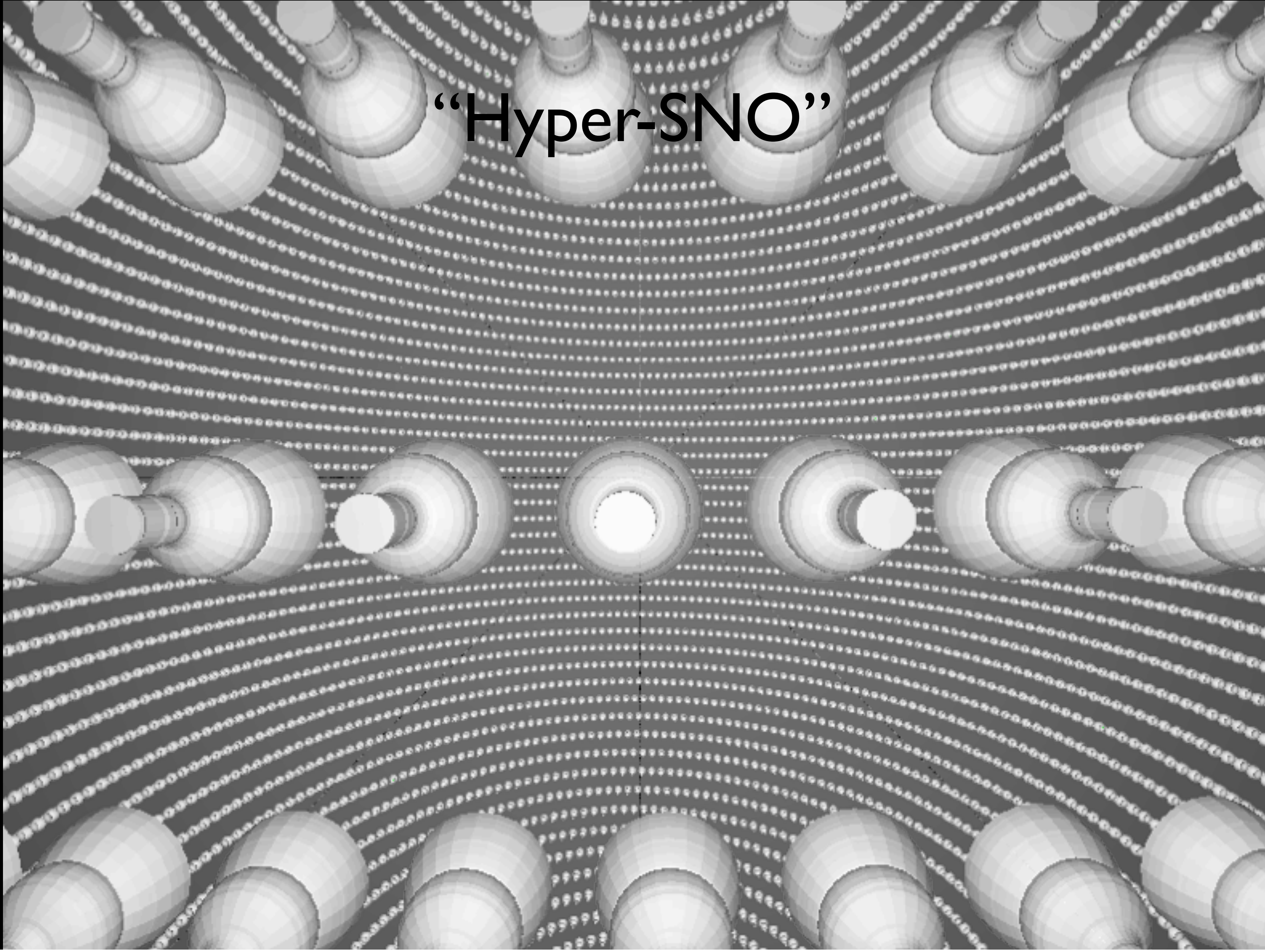
28 M diameter

10,055 PMTs

60M triangles



Bundled with Chroma

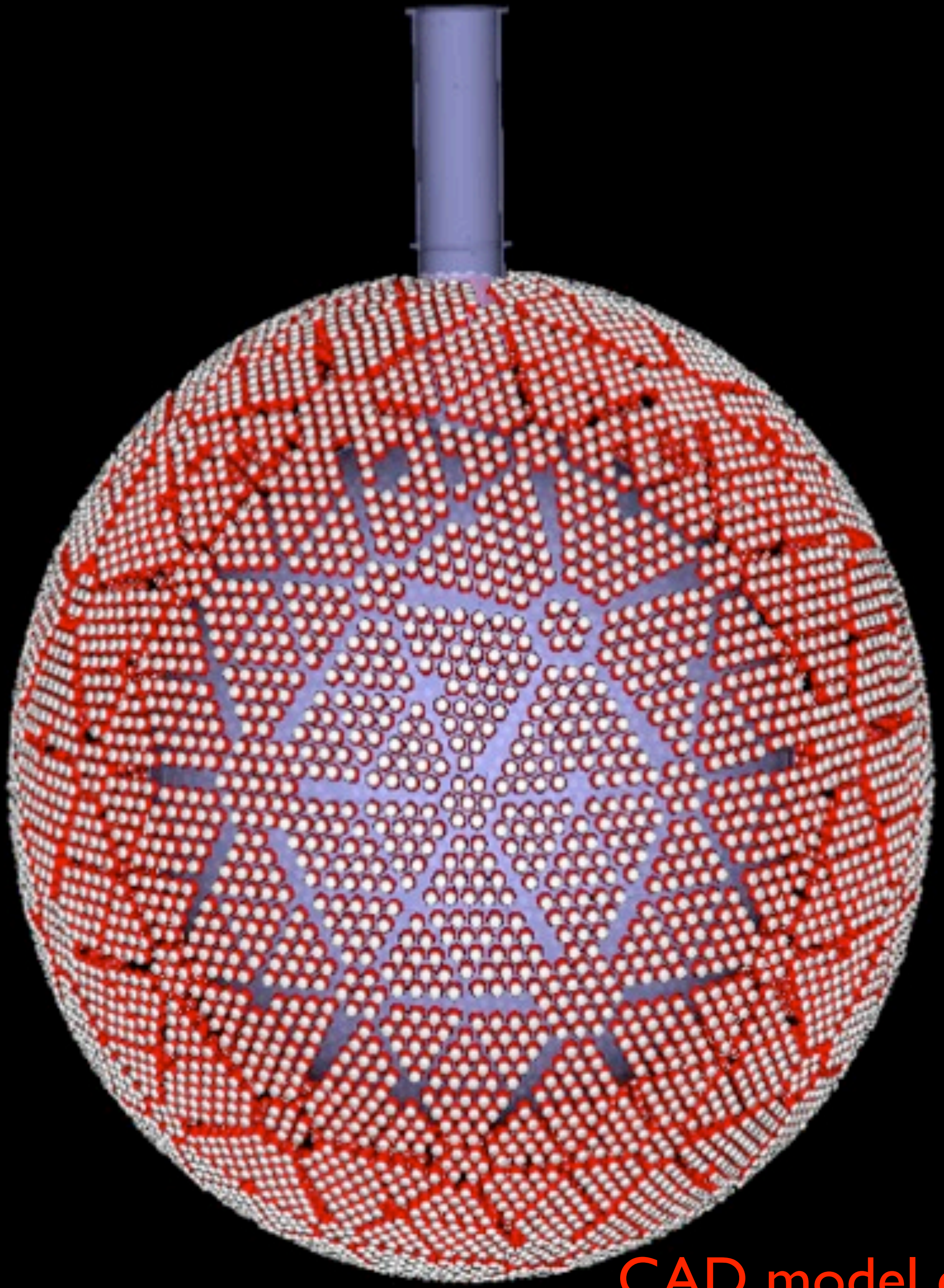


“Hyper-SNO”

SNO Detector in Chroma

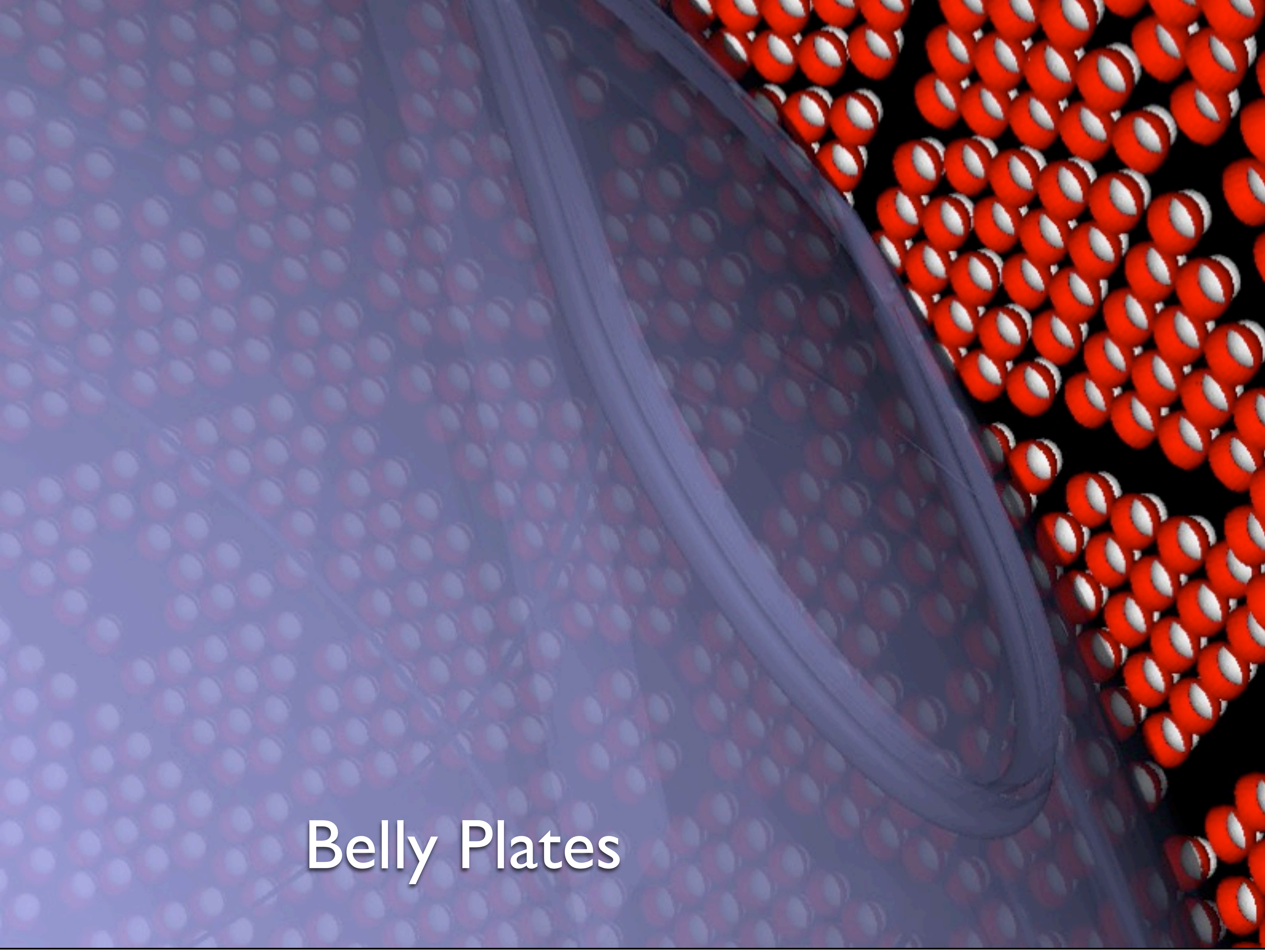
Speed (outside):
3-10 fps
= 1.4M to 4.8M track steps per second

Speed (inside):
? fps
= ?? M track steps per second



22 million triangles in whole detector

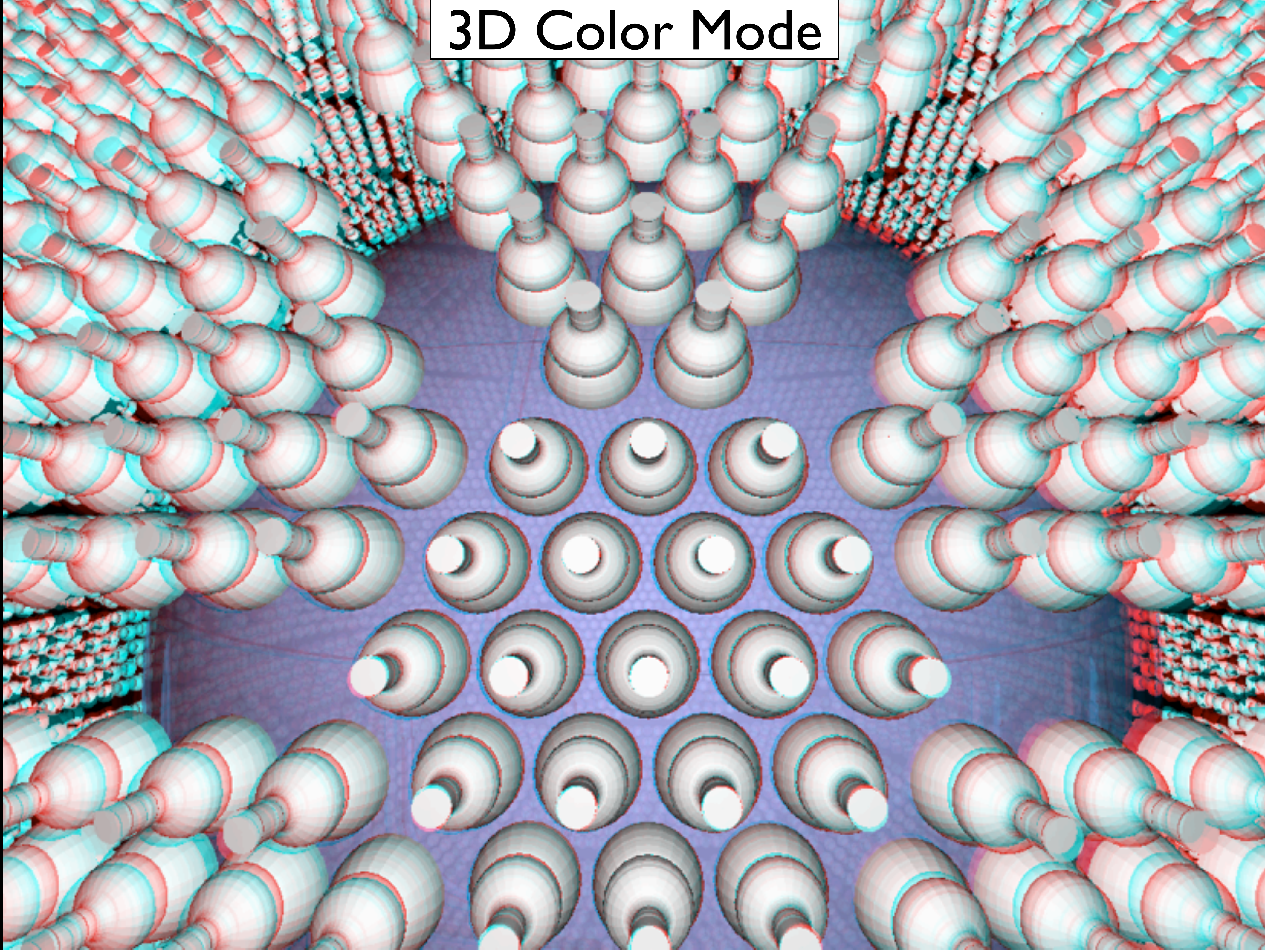
CAD model courtesy of Chris Ng



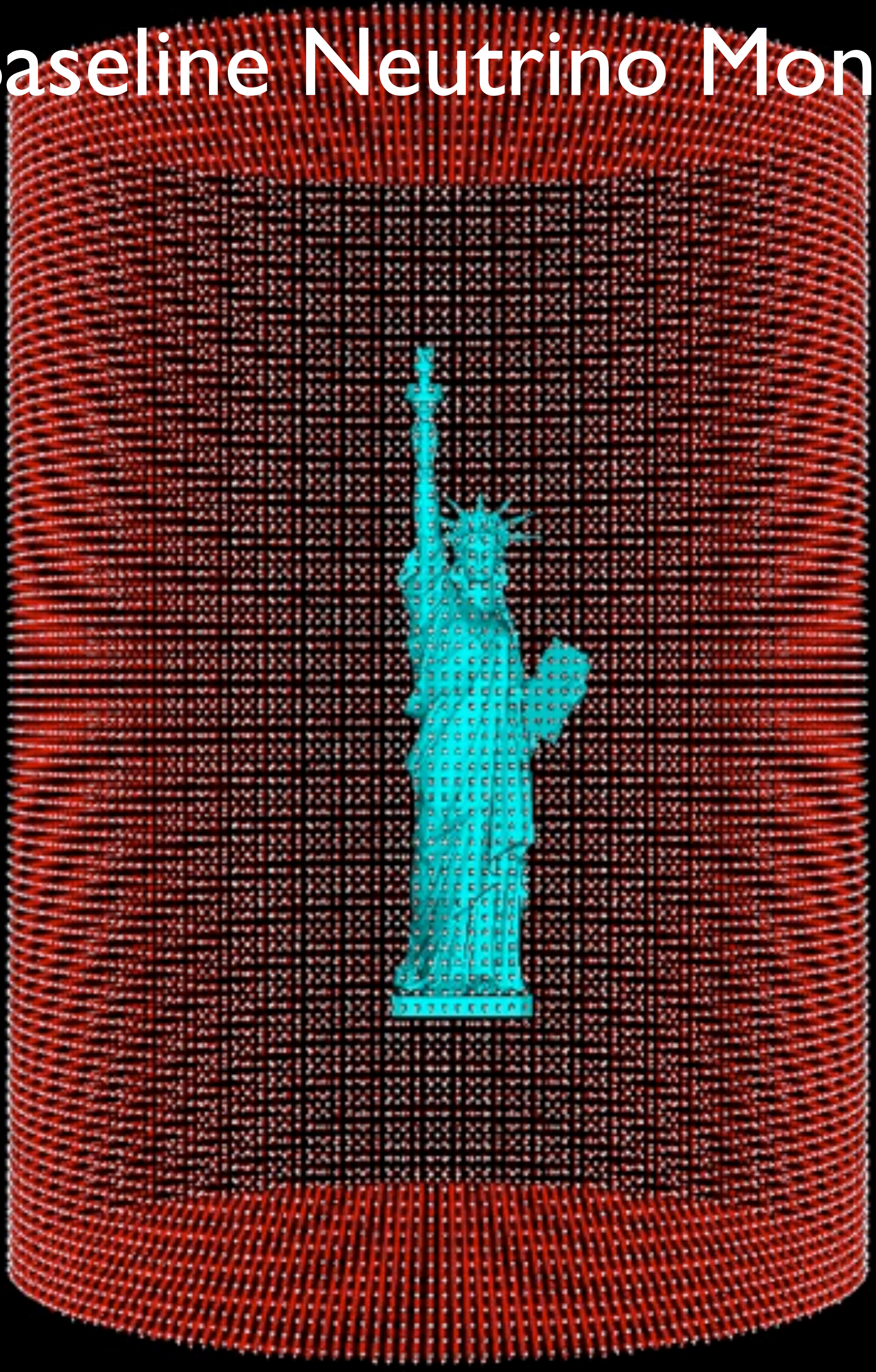
Belly Plates

Inside, looking up at the neck

3D Color Mode



Long Baseline Neutrino Monument



Most time consuming part: finding Statue of Liberty STL file and determining scale factor for accurate height.

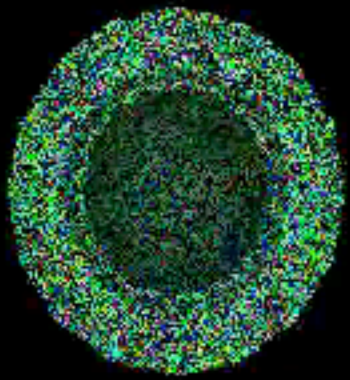
Nearly every CAD program can dump STL files.

Quickly add complex shapes to detector *without* massive performance loss.

Physical Rendering

LBNE candidate 12” PMT + Light Collector in water

Physical Rendering

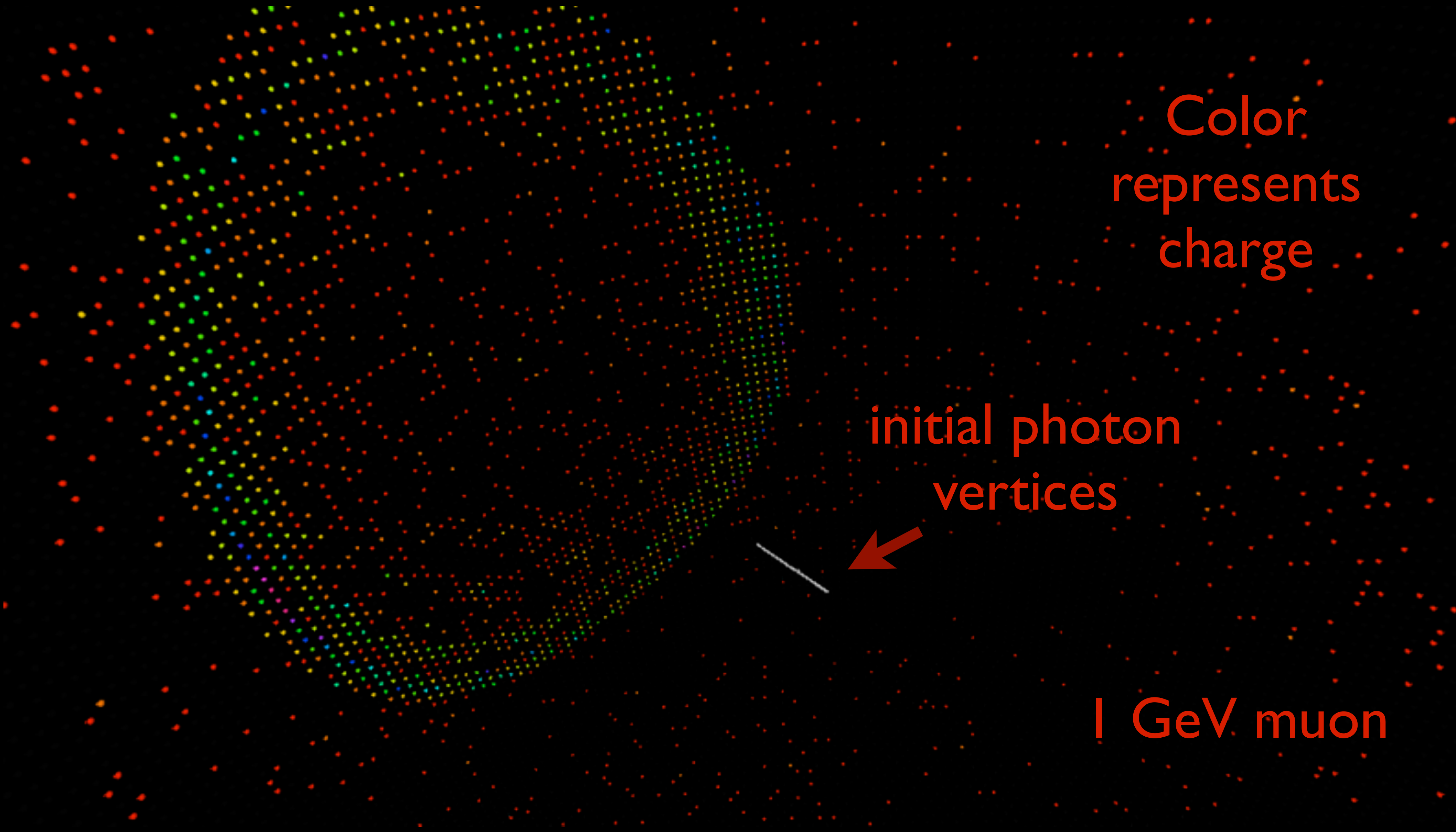


LBNE candidate 12” PMT + Light Collector in water

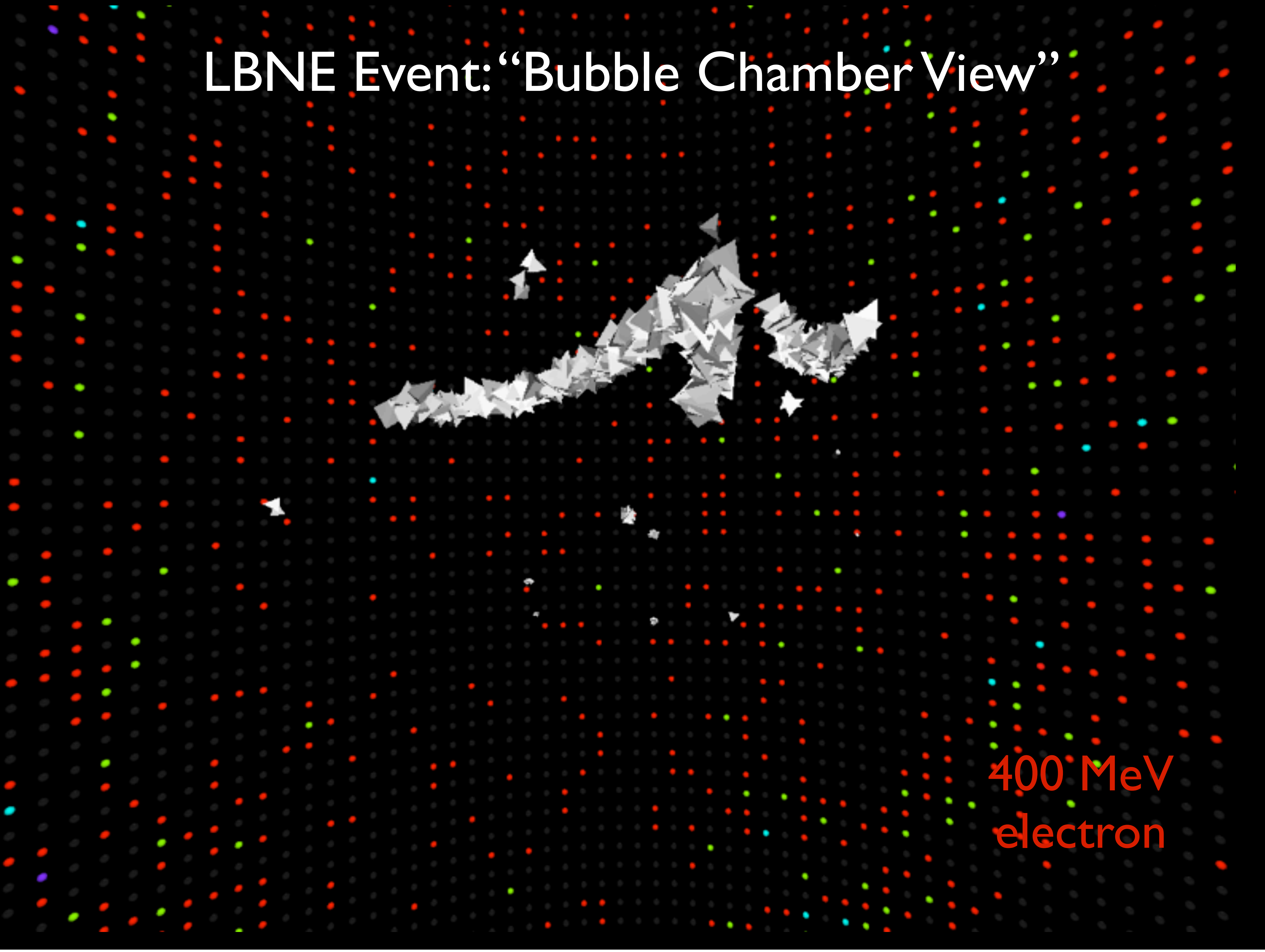
Simulation

- Chroma propagates only photons (refraction, diffuse and specular reflection, Rayleigh scattering, easy to add others)
- Our Simulation class can spawn multiple GEANT4 processes in the background to propagate primary particles and harvest the optical photons for propagation by Chroma.
- If you produce starting photon vertices some other way, you can feed those directly to Chroma.

LBNE Event


Color represents charge
initial photon vertices
1 GeV muon

LBNE Event: "Bubble Chamber View"



Performance on LBNE (and Hyper-SNO)

- 600 thousand photon vertices per second per GEANT4 process (saturates at ~2M per Chroma instance)
- ~7 million track segments per second (no physics)
- ~3 million photons per second (incl. physics)
- Requires ~2 GB of GPU memory and ~5 GB of host memory for setup.
- Ex: 1 GeV electrons = 4.1 events per second

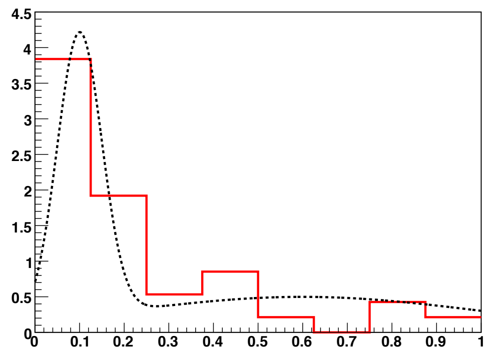
Likelihood Calculation

- Fitting with a particular particle hypothesis reduced to a problem of generating events with the appropriate distribution of initial photon vertices.
- *Turning an electron fitter into a muon fitter (or a pi0 fitter) is literally just changing "e-" to "mu-" in the generator function.*
- You are free to decide what to integrate over (e- track histories) vs. fit for (initial particle direction)
- The purpose of Monte Carlo (both GEANT4 and Chroma) is to integrate over these uninteresting (or unobservable) degrees of freedom.
- This distinction matters when fitting π_0, where you probably want to fit for the direction of a gamma, rather than integrate over the space of all π_0 decays.

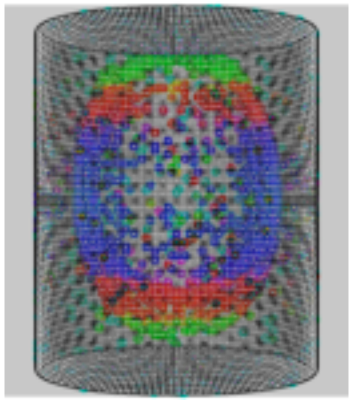
Evaluating PDFs with Limited Monte Carlo

Usual technique:

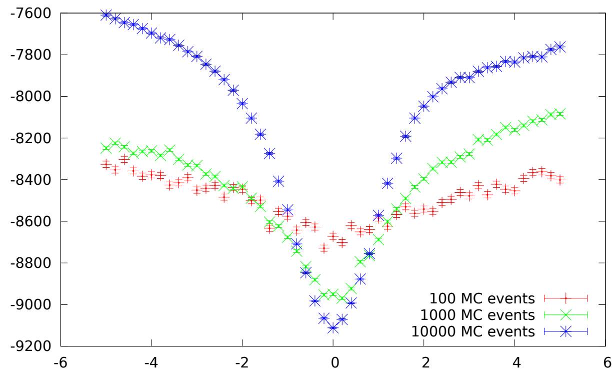
Run Monte Carlo



Bin Events into PDFs



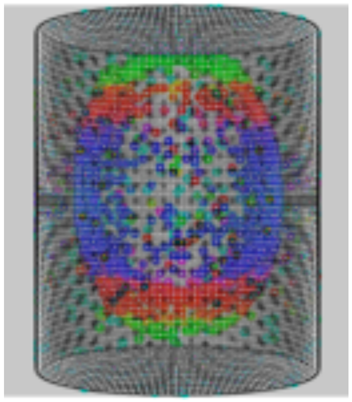
Load data event to fit



Evaluate PDFs for Likelihood

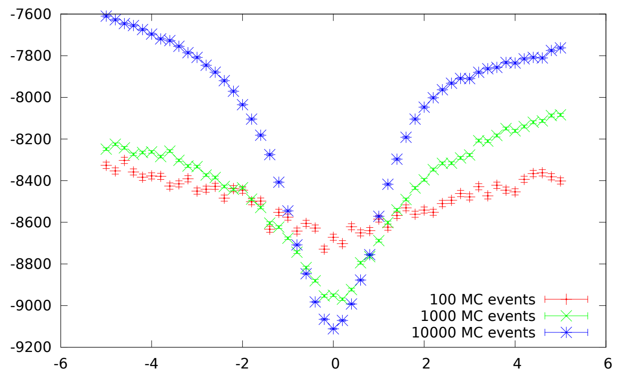
Evaluating PDFs with Limited Monte Carlo

Chroma does it the other way

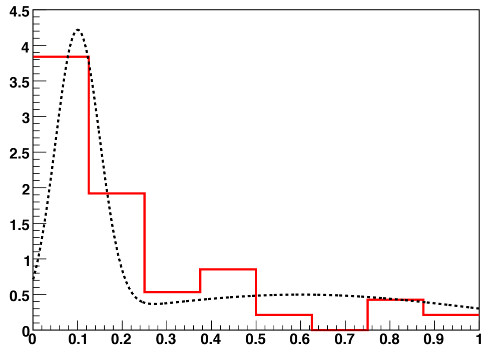


Load data event to fit

Run Monte Carlo



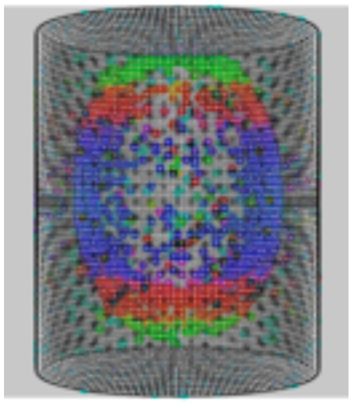
Evaluate PDFs for Likelihood



Bin Events into PDFs

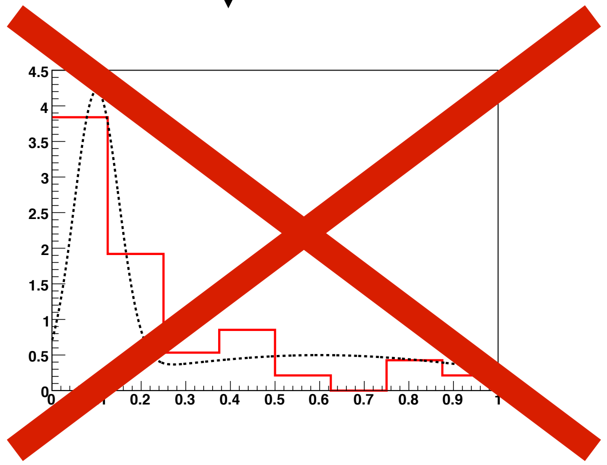
Evaluating PDFs with Limited Monte Carlo

Even better!

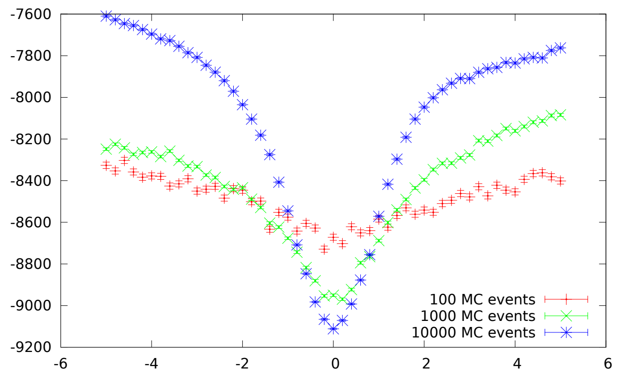


Load data event to fit

Run Monte Carlo

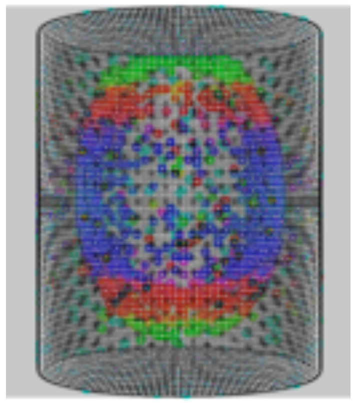


Bin Events into PDFs



Evaluate PDFs for Likelihood

Evaluating PDFs with Limited Monte Carlo

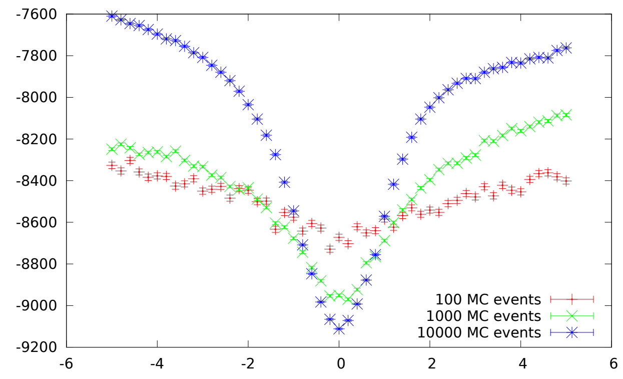


Load data event to fit

$\longrightarrow$

Run Monte Carlo

$\downarrow$



Evaluate PDFs for Likelihood

Save storage space by not keeping around all the bins you just going to throw away!

“Oversampling”

We are using Monte Carlo methods to evaluate a “triple” integral:

$$P_i(t) = \int \text{Particle Physics} \int \text{Optics} \int \text{Detector Response}$$

But not all parts of the code have the same throughput:

$$P_i(t) = \int \text{GEANT4} \int \text{Chroma} \int \text{DAQ}$$

2M photons/ sec (50 events/sec @ 100 MeV)	3.5M (or 10M) photons/sec (250 events/ sec)	4k events/ sec

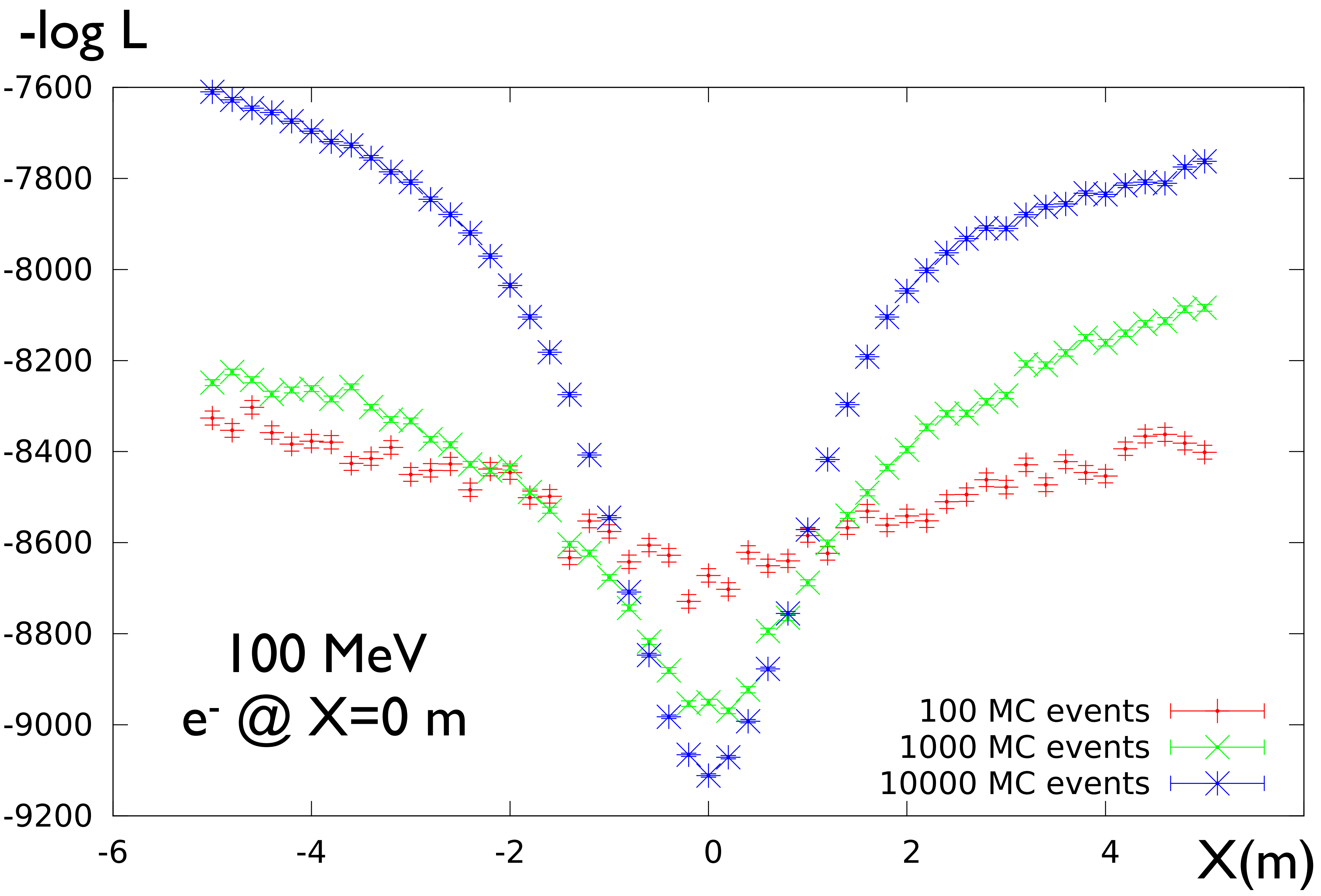
Setup to Calculate a Likelihood

```python
from chroma import Simulation, Likelihood, \
        constant_particle_gun
import chroma.demo
from itertools import islice
import numpy as np

detector = chroma.demo.detector()
sim = Simulation(detector)

gen = constant_particle_gun('e-',
                            pos=(0,0,0),
                            dir=(1,0,0),
                            ke=100)

event = sim.simulate(islice(gen,1)).next()
likelihood = Likelihood(sim, event)
```

Scan Likelihood Space

```
for x in np.linspace(-1.0, 1.0, 20):
    hypothesis = constant_particle_gun('e-',
                                       pos=(x,0,0),
                                       dir=(1,0,0),
                                       ke=100)
    L = likelihood.eval(hypothesis, neval=800,
                        nreps=4, ndaq=32)
    print x, L
```

Example Likelihood Scan



Future Plans

- Photon propagation core is mostly stable, but always looking for performance opportunities.
- Add re-emission of absorbed photons
- Understand how best to leverage Monte Carlo output to balance bias and variance in the channel PDFs. (Variable size histogram bins, Kernel estimation? Group infrequently hit, but adjacent channels.)
- Refine our strategy (not in this talk) for minimization of a stochastic likelihood function.
(AKA: 7D is hard)
- Do some likelihood maximization!

Conclusion

- Chroma is a really fast photon Monte Carlo, and Moore's Law is on our side!
- Chroma lets you do realtime visualization, fast simulation, and likelihood estimation.
- Many things still to do, but the derivative is positive!

Get it at:

http://chroma.bitbucket.org

One more thing...