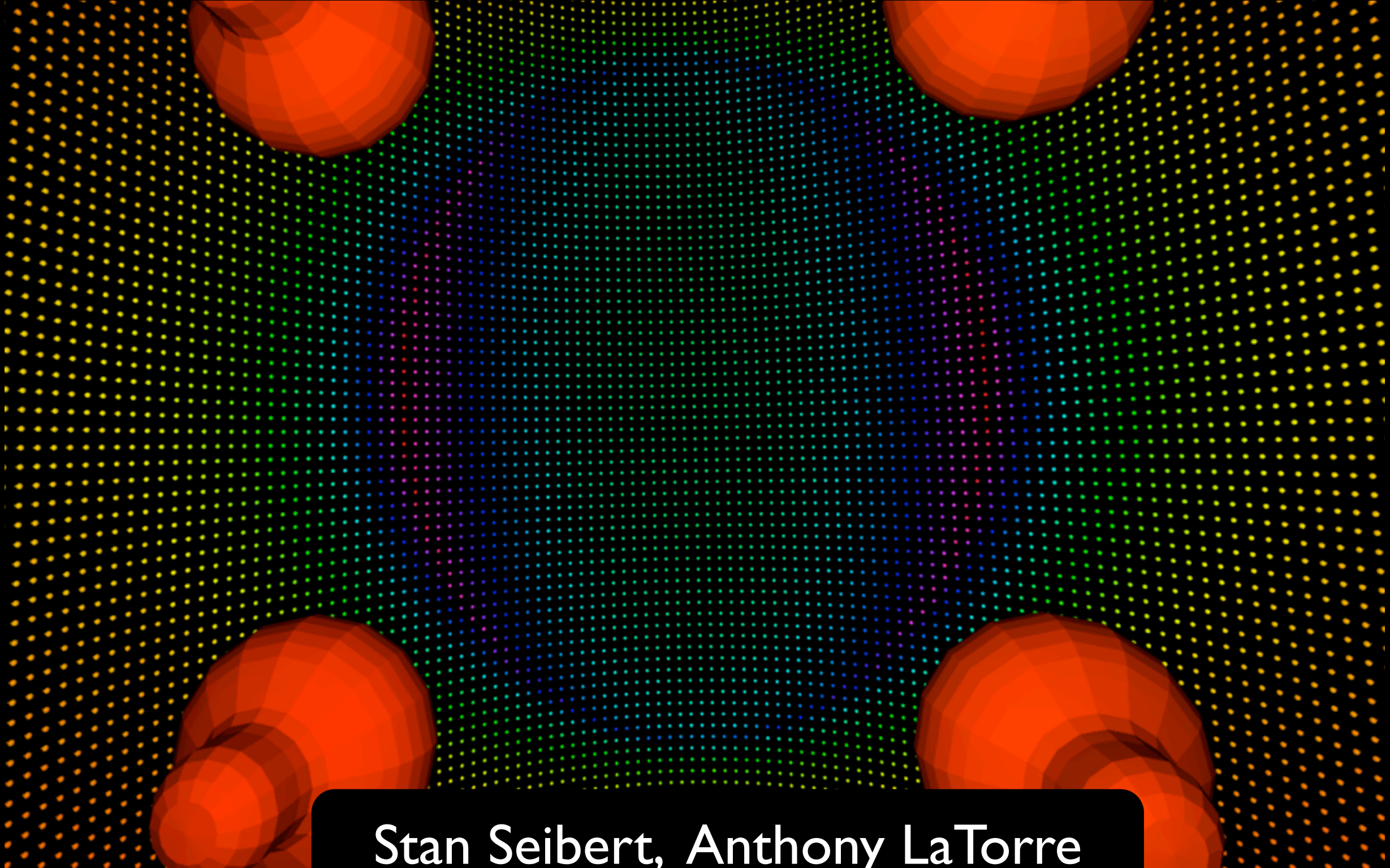


# Chroma: *Reconstruct like it's 2099*



Stan Seibert, Anthony LaTorre  
University of Pennsylvania  
PXPS, June 18, 2012

# Motivations

- Simulation of large photon-based detectors has been very successful (SNO, SuperK, MiniBooNE, etc.) with percent-level precision achieved.
- As a result, future detector design efforts rely heavily on Monte Carlo simulation for design optimization.
- We have found from experience that reconstruction also is leaning more and more heavily on Monte Carlo techniques.
- GEANT4 is ill-suited for optical photon simulations.
- So we embarked on a project last year to see what was possible, which called *Chroma*...

# Interlude: Reconstruction is Important!

- A 200 kton water Cherenkov detector had CPV sensitivity comparable to a 34 kton liquid argon TPC. Why?
- Based on two assumptions:
  - Can only accurately reconstruct CC QE  $\nu_e$  interactions (42% of  $\nu_e$  interactions from 1.5-2 GeV)
  - Can only keep 40% of signal to reduce  $\pi_0$  to acceptable level
- *Water Cherenkov detector mass handicapped by factor of 6!*
- Any improvements in reconstruction could relax these assumptions and boost the detector mass.  
(Or inversely, cut the detector costs to achieve the same physics.)

# What is Chroma?

Chroma is a **GPU-accelerated raytracer** and **Monte Carlo photon simulation** written in a mixture of Python and CUDA:

- Uses GEANT4 or your own software to create initial photon vertices. (Included generator does Cherenkov light; bolt on your own for other light production.)
- **Up to 200x faster than GEANT4** at simulating 1 GeV electrons in an LBNE-sized detector.  
(millions of photons steps/sec)
- Designed to be used either as a library for other Python applications, a server for communication with existing C++ programs, or standalone use.

*Disclaimer: We didn't know about the GPU-accelerated lattice QCD program with the same name until much too late to change...*

# Why would you use Chroma?

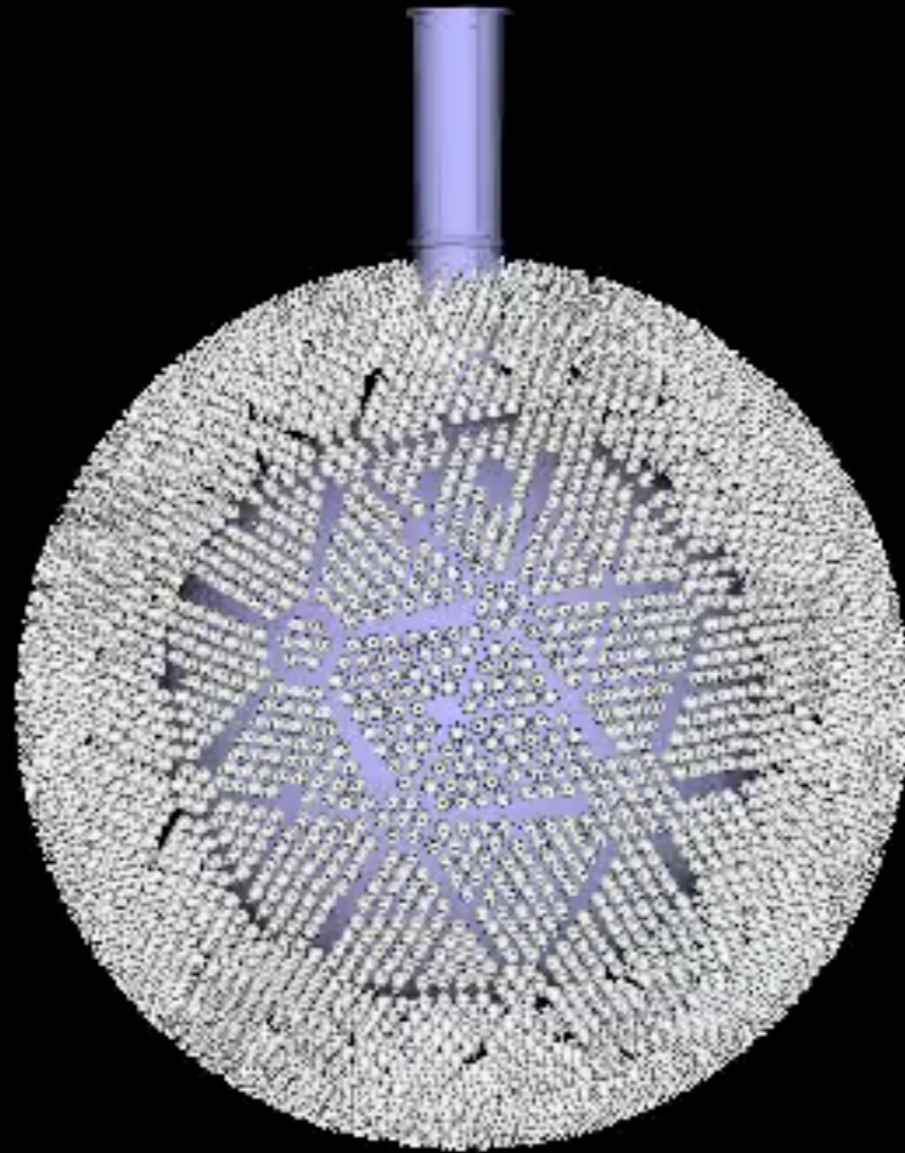
We've learned that Chroma can be a lot of things:

- An interactive 3D geometry exploration and event visualization application.
- A standalone Monte Carlo simulation of electrons, muons, and  $\pi_0$  events.
- A “Monte Carlo photon propagation server” that can be used by existing GEANT4-based simulations.
- A PDF generation tool
- A time/hit likelihood calculator
- Hypothesis testing of hit patterns
- And maybe even a reconstruction algorithm...

# Interactive Geometry Exploration



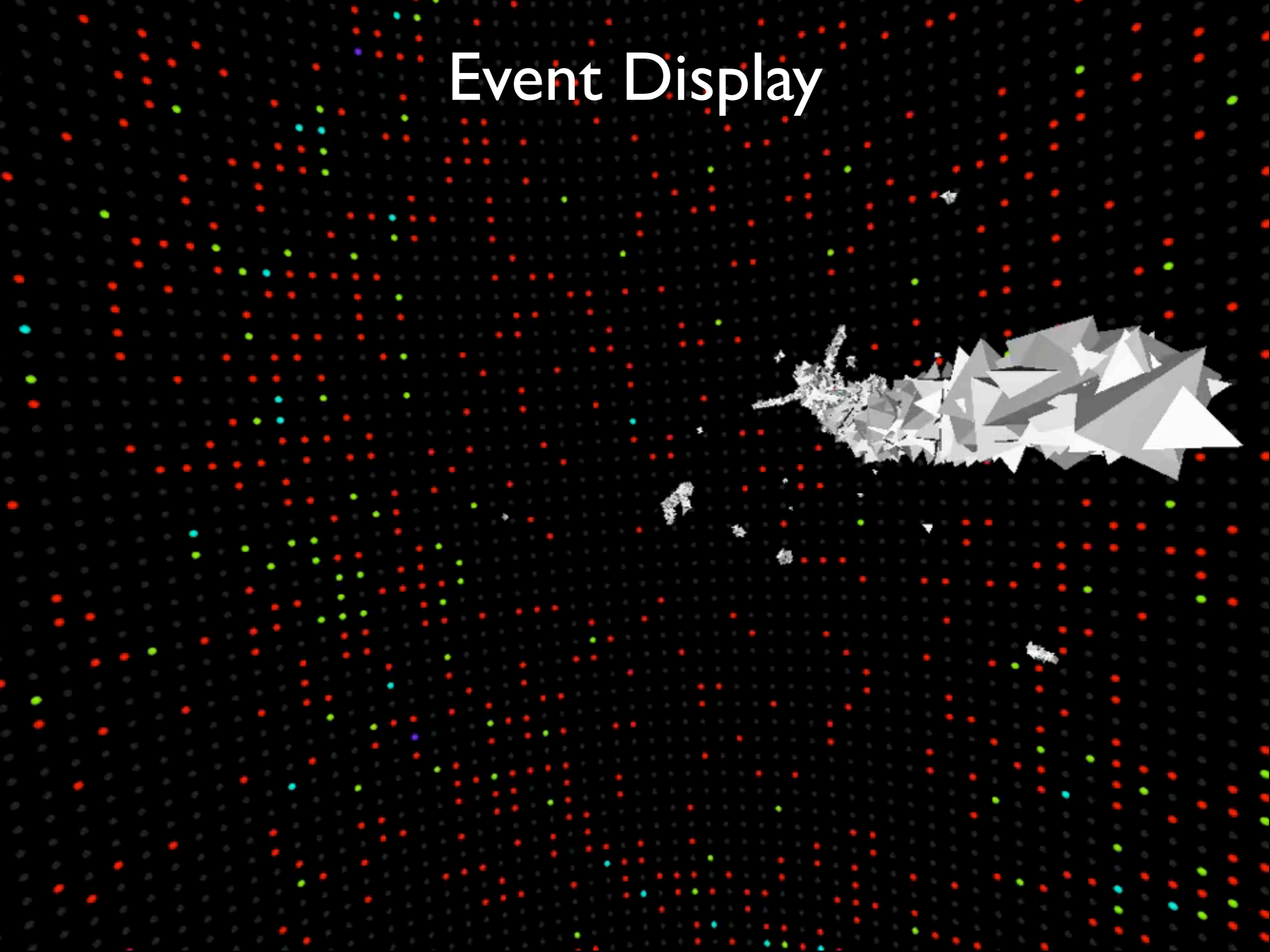
# Interactive Geometry Exploration



# Event Display



# Event Display



# Detector Description

New detectors can be defined rather quickly:

- Basic components are triangle mesh objects which can either be constructed programmatically or exported from your favorite CAD software as STL files.
- Bulk and surface material properties need to be specified, much as in GEANT4.
- *Every triangle can have a different surface material!*

Simplifies description of objects which have a coating placed only on part of a surface.

Most detectors can be created with 100-200 lines of Python + externally created solid shapes.



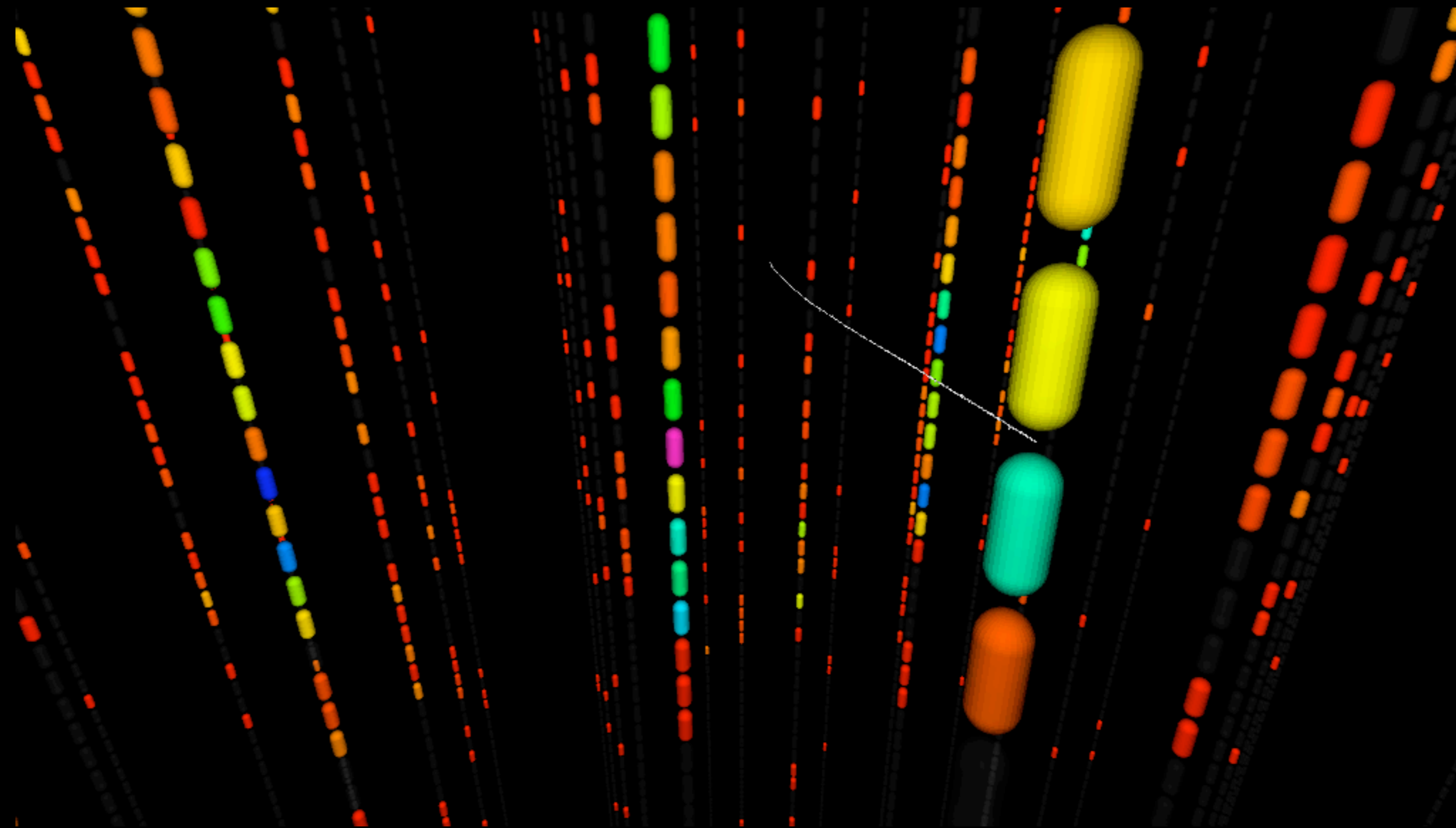
# Water-MICA

1 m

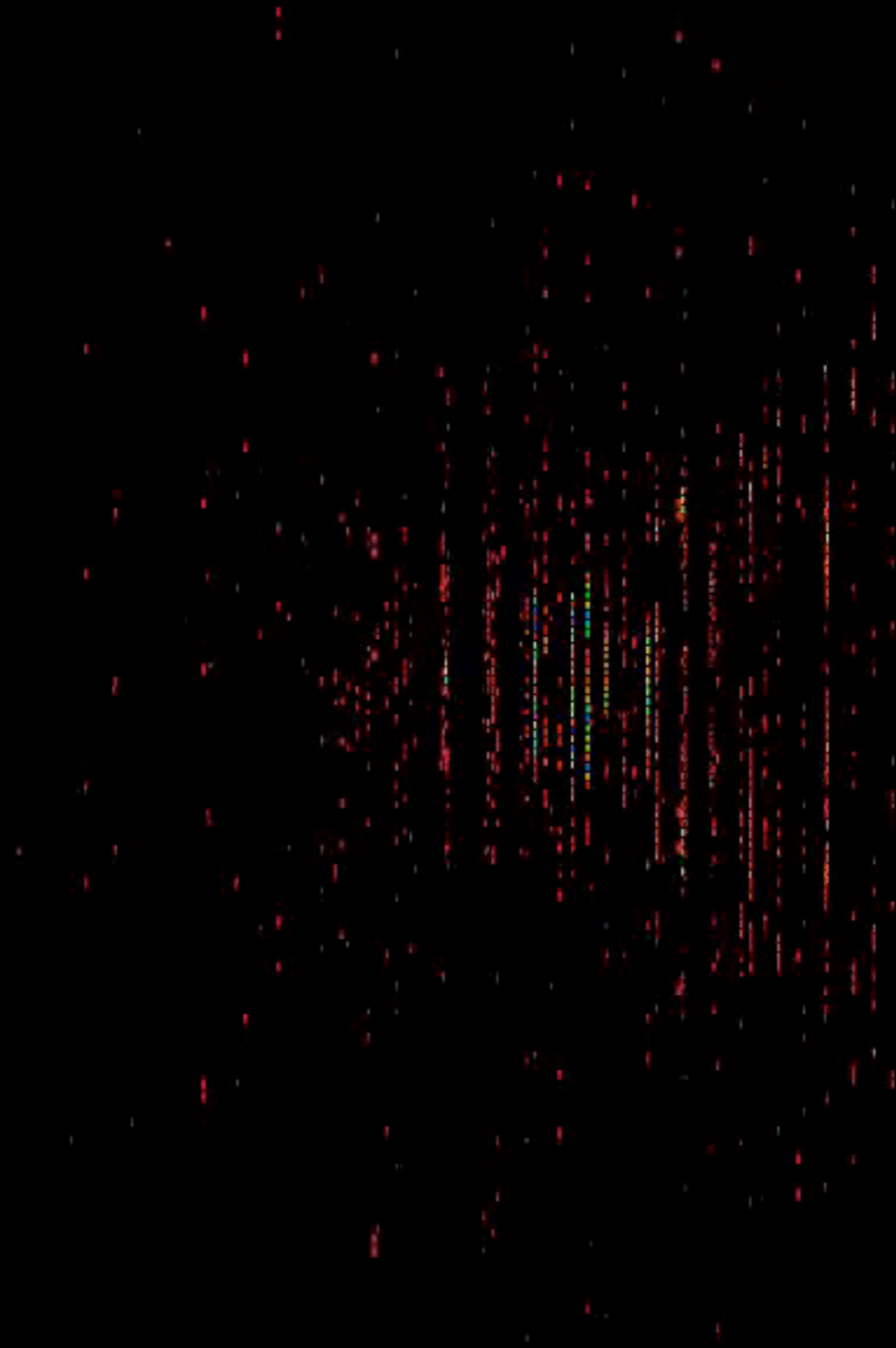
The image displays a series of vertical cross-sections of a material structure, likely a membrane, showing a transition from a dense, layered structure on the left to a more porous, grid-like structure on the right. A scale bar on the left indicates 1 meter. The structure consists of multiple layers of interconnected, elongated, yellowish-green components, possibly representing water molecules or MICA particles, arranged in a regular, repeating pattern. The overall appearance is that of a highly ordered, porous material.



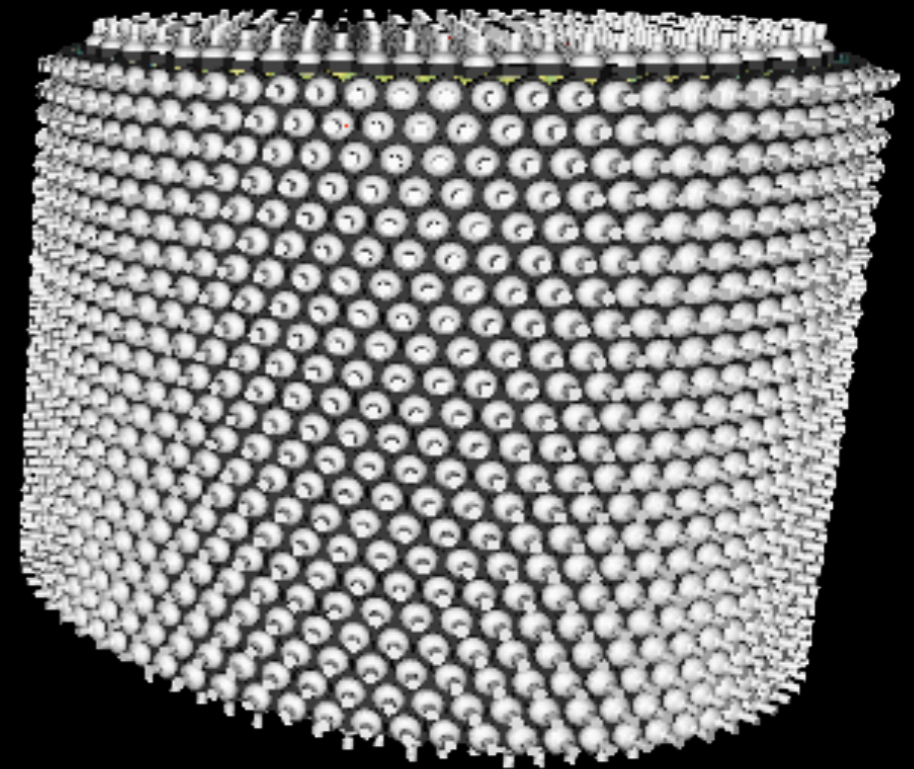
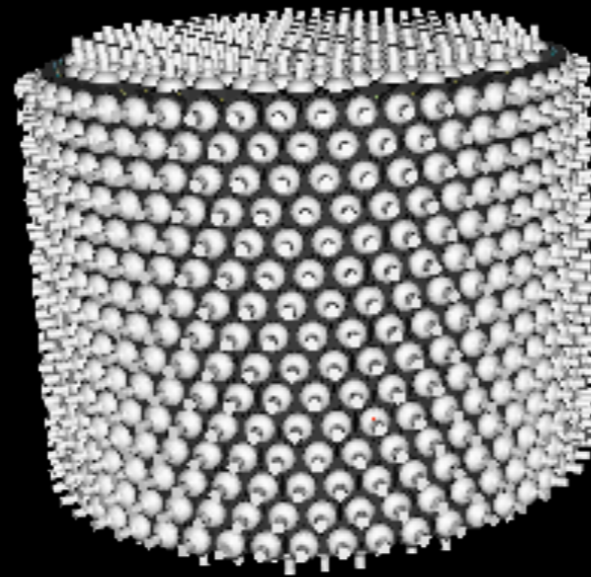
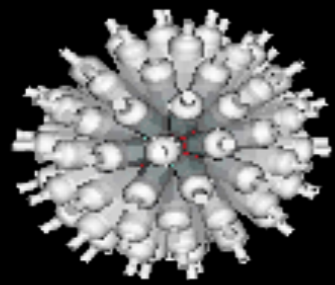
# Water-MICA



# Water-MICA



# Single Phase Liquid Argon Dark Matter Detectors



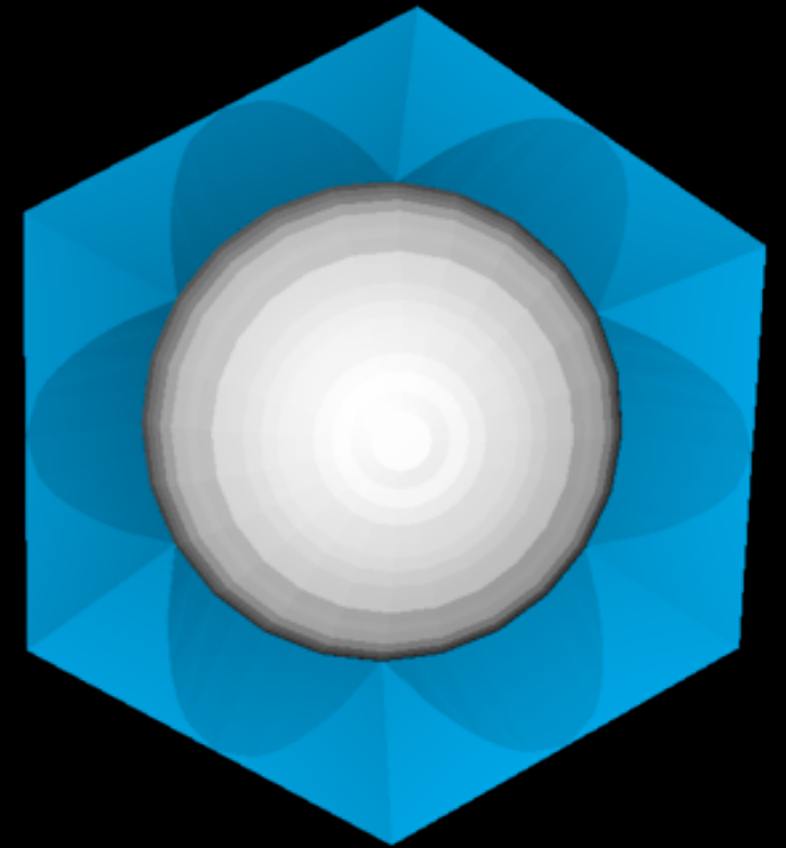
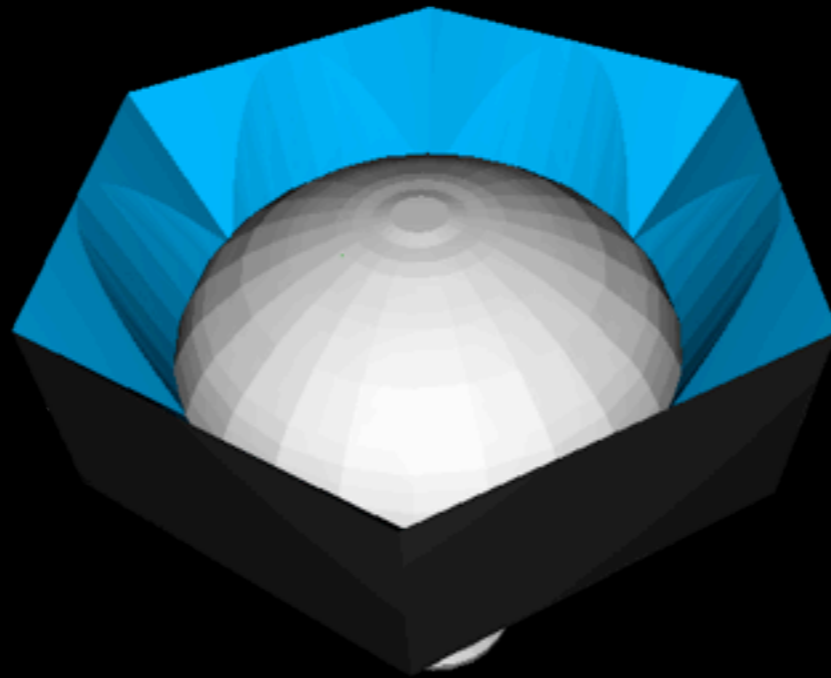
**MiniCLEAN**  
(~500 kg total)

**CLEAN-40**  
(44 tons total)

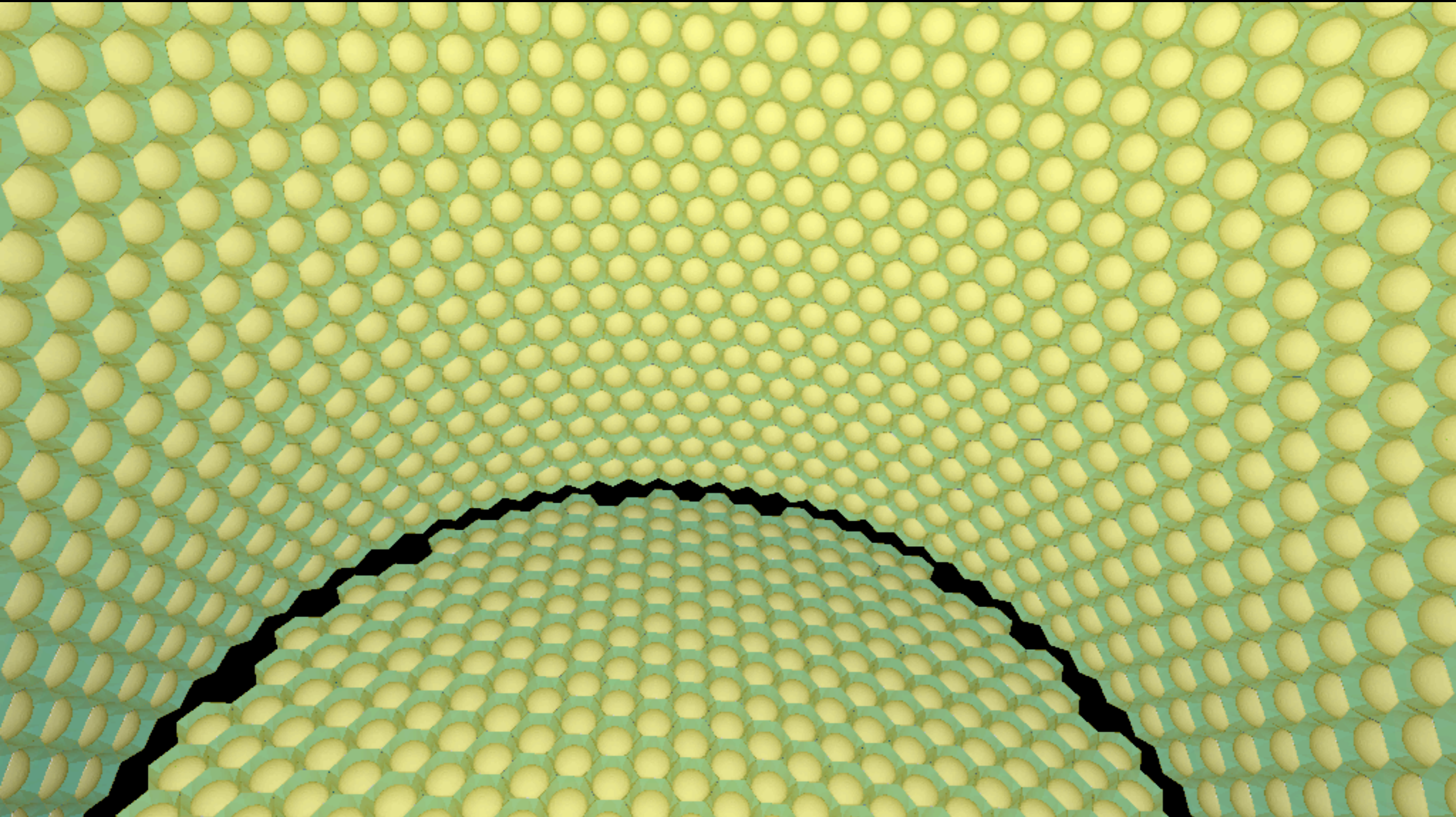
**CLEAN-140**  
(140 tons total)



# Single Phase Liquid Argon Dark Matter Detectors



# Inside of 140 ton Detector

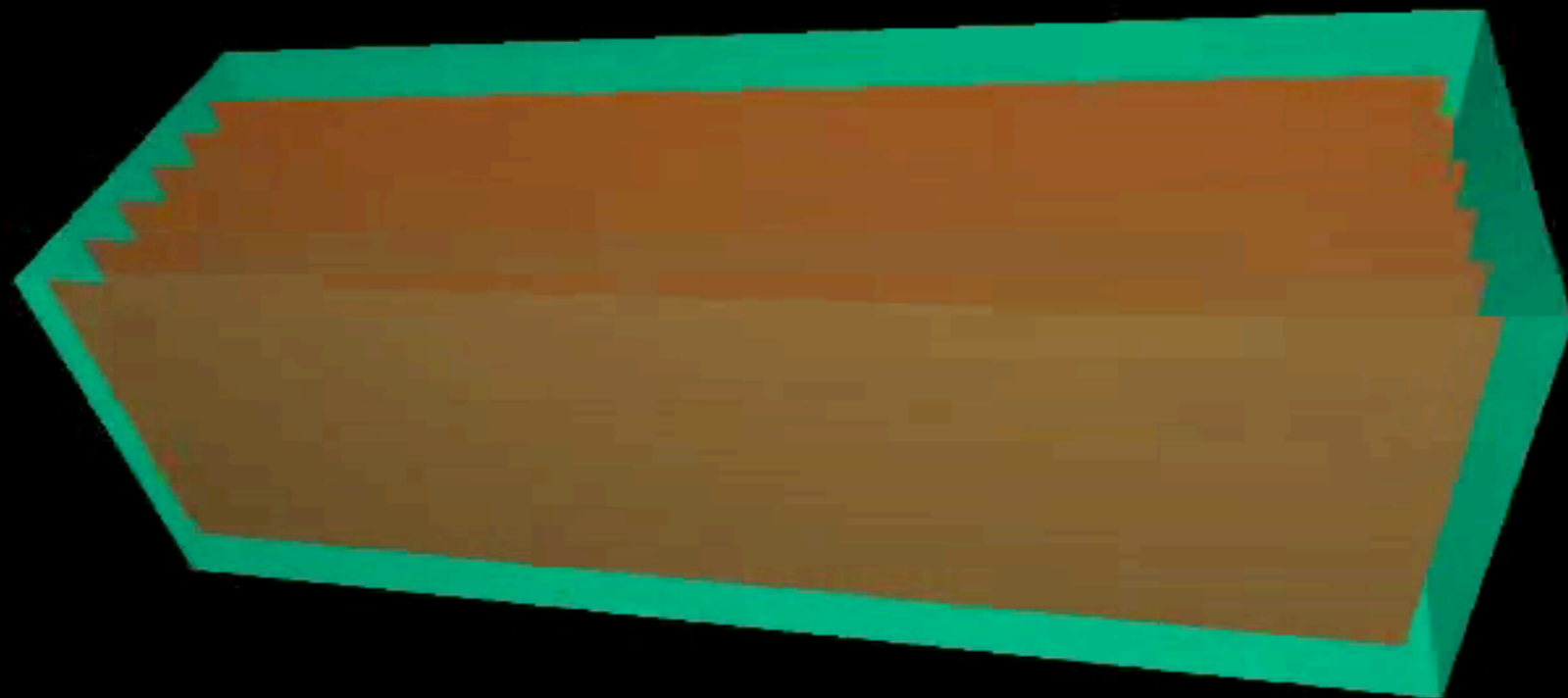




# LBNE Far Detector

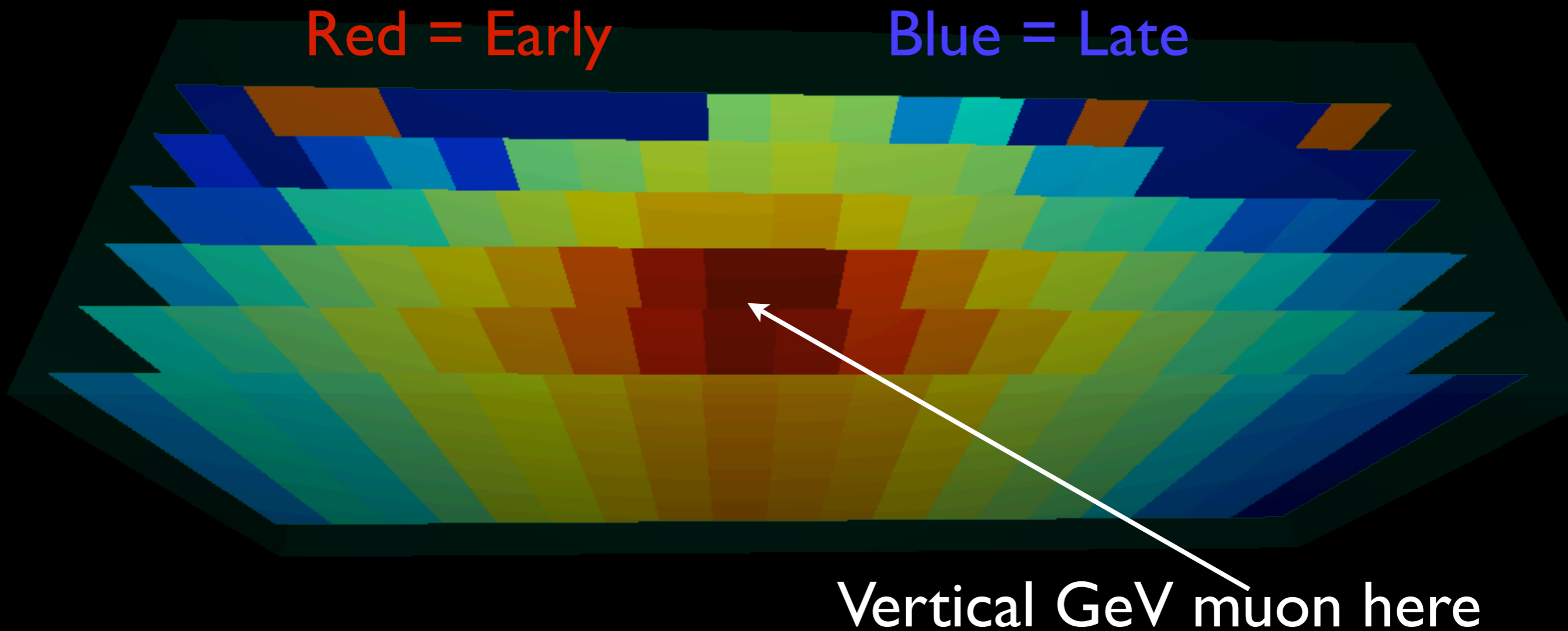
16m wide  
48.6m long  
16m tall

*(Not 10 kton dimensions!)*



Anode plane spacing: 2.5m  
= maximum drift distance of 1.25m

# Photon Detection Time



24 million photons in 6.4 seconds in this very simple geometry.

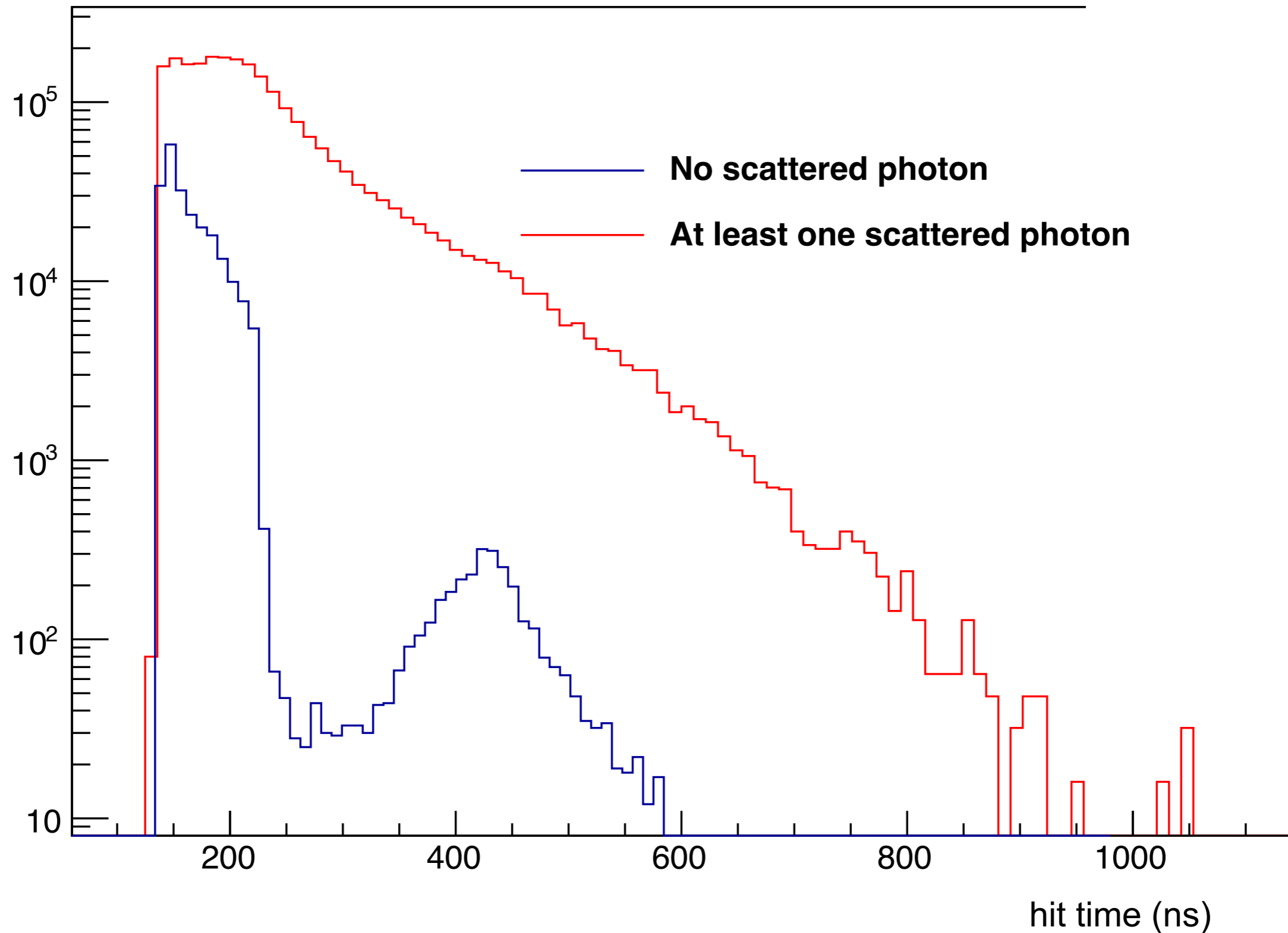
# Chroma can do real photon physics!

In full simulation mode, Chroma does realistic propagation of individual photons, just like GEANT4:

- Wavelength-dependent index of refraction
- Fresnel refraction and reflection coefficients at the boundary between media
- Rayleigh scattering
- Specular reflections off surfaces
- Diffuse reflections off surfaces
- “Detection” of the photon at a photosensitive surface (incl. complex index of refraction if desired)
- Re-emission in the bulk (For scintillation detectors)
- Re-emission from surfaces (For liquid Ar detectors)

(Added by A. Mastbaum at UPenn)

# Simulated Photons in 200 kton WC



*Simulation of 1 GeV muons w/ one GPU: 4 per second!*



# Why is Chroma so fast?

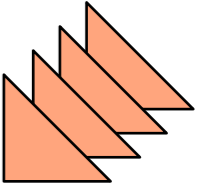
To win 200x in photon propagation, we had to combine two different lines of technological development:

- **Better software:** Computer graphics researchers have developed many techniques for efficiently tracing the paths of photon rays. These techniques focus on *surfaces*, rather than *solids*.

Existing rendering libraries try to avoid physics whenever possible!  
Can't just download Blender/Maya/POV-Ray...

- **Better hardware:** Standard graphics processing units (GPUs) are now fully programmable, massively parallel vector machines. Moreover, the exponential growth of computing power in GPUs is currently faster than CPUs.

# Technology #1: Faster Geometries with Triangles



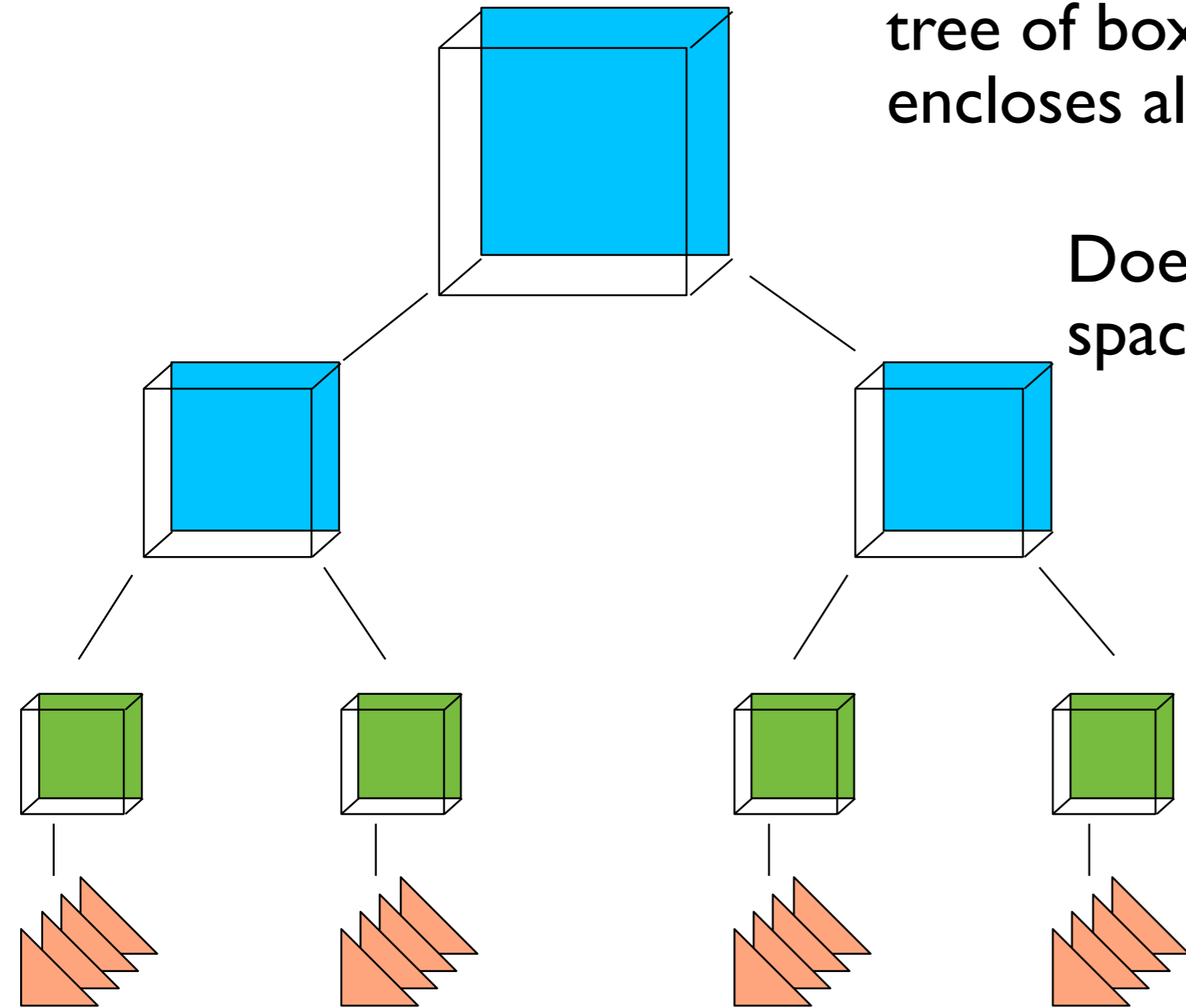
- GEANT4 is slow because of the overhead of delegating geometric operations to polymorphic methods. *Abstraction prevents simplification and optimization, in this case.*
- A triangle mesh can reasonably approximate most surfaces, and is pure data. No geometry-specific code → one code path to optimize.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

# 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



Testing a ray with a mesh of 50 million triangles only requires 130 box intersections and 2 triangle intersection tests.

# Technology #2: The GPU

Vector computing was killed in the 1990s by cheap, powerful PCs, only to be resurrected 15 years later by the 3D graphics industry to fuel demand for video games.



*GeForce GTX 580 3GB*  
**\$500**  
*1.581 TFLOPS*  
*3GB device memory*  
*192 GB/sec memory bandwidth*



*Tesla C2070*  
**\$2200**  
*1.288 TFLOPS*  
*6GB device memory*  
*144 GB/sec memory bandwidth*

# What else can Chroma be used for?

*A sufficiently fast Monte Carlo is indistinguishable from a maximum likelihood reconstruction algorithm.*

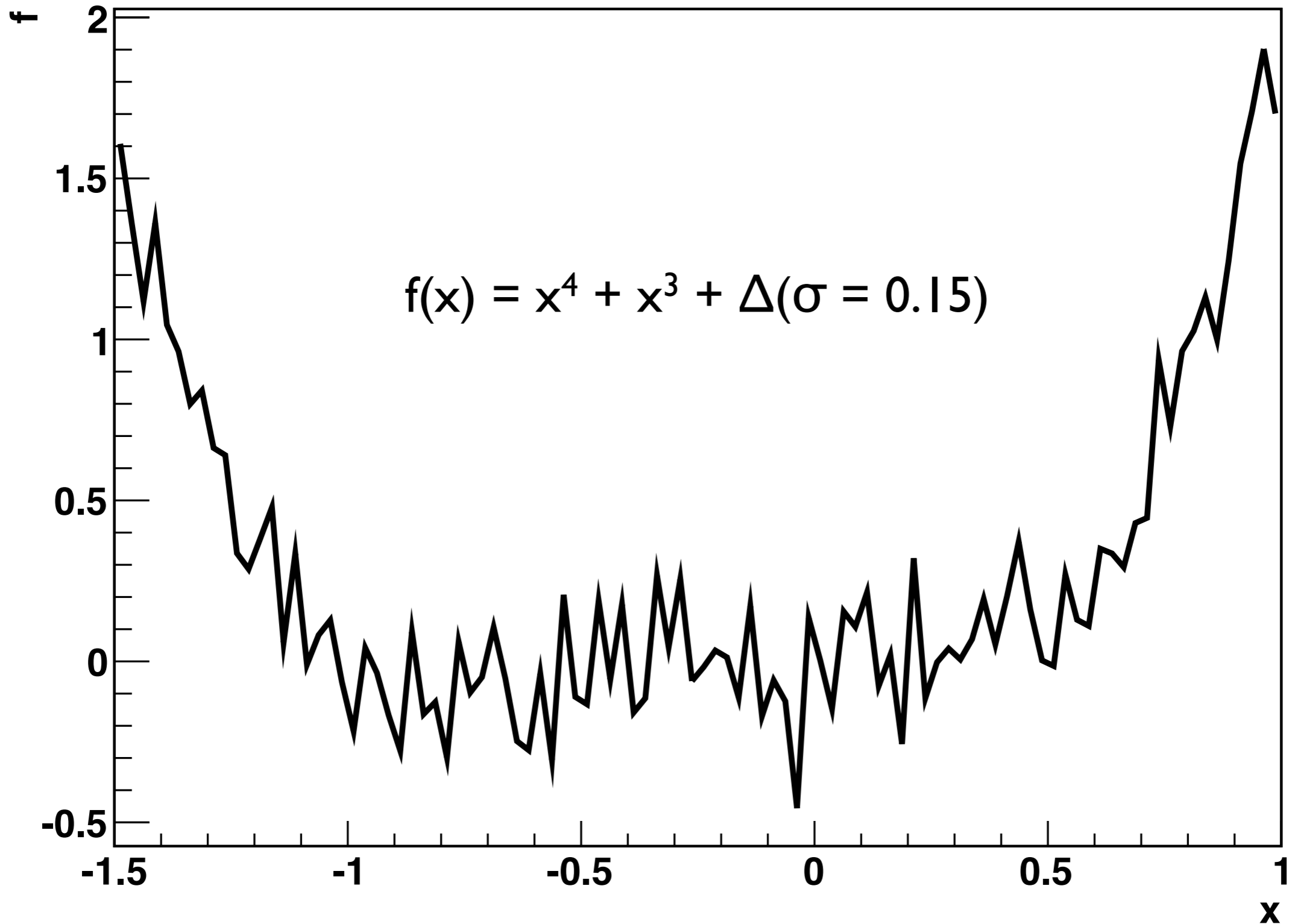
*(With apologies to Arthur C. Clarke)*

# Maximum Likelihood Reconstruction

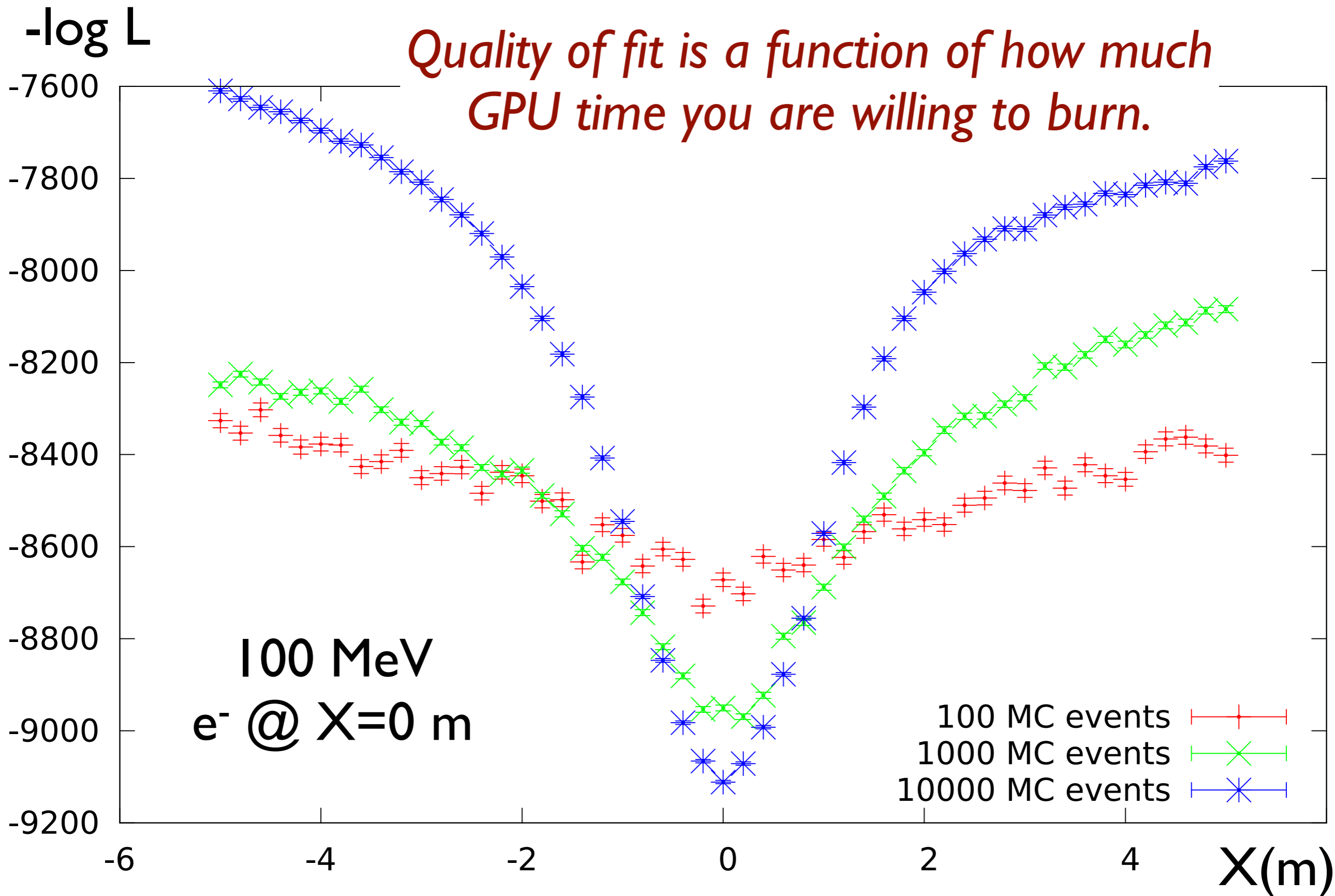
- In the limit of infinite computing power, for each point in parameter space you could produce a PDF for every observable of interest.
- All you need is a *fast* Monte Carlo (like Chroma) that reproduces the real detector geometry and all the physics.
- Very natural to include scattered light, realistic geometry effects, DAQ, etc.
- Easy to track changing detector conditions. Just change the detector description!
- If your MC is *accurate*, then *this is the best you could possibly do*.



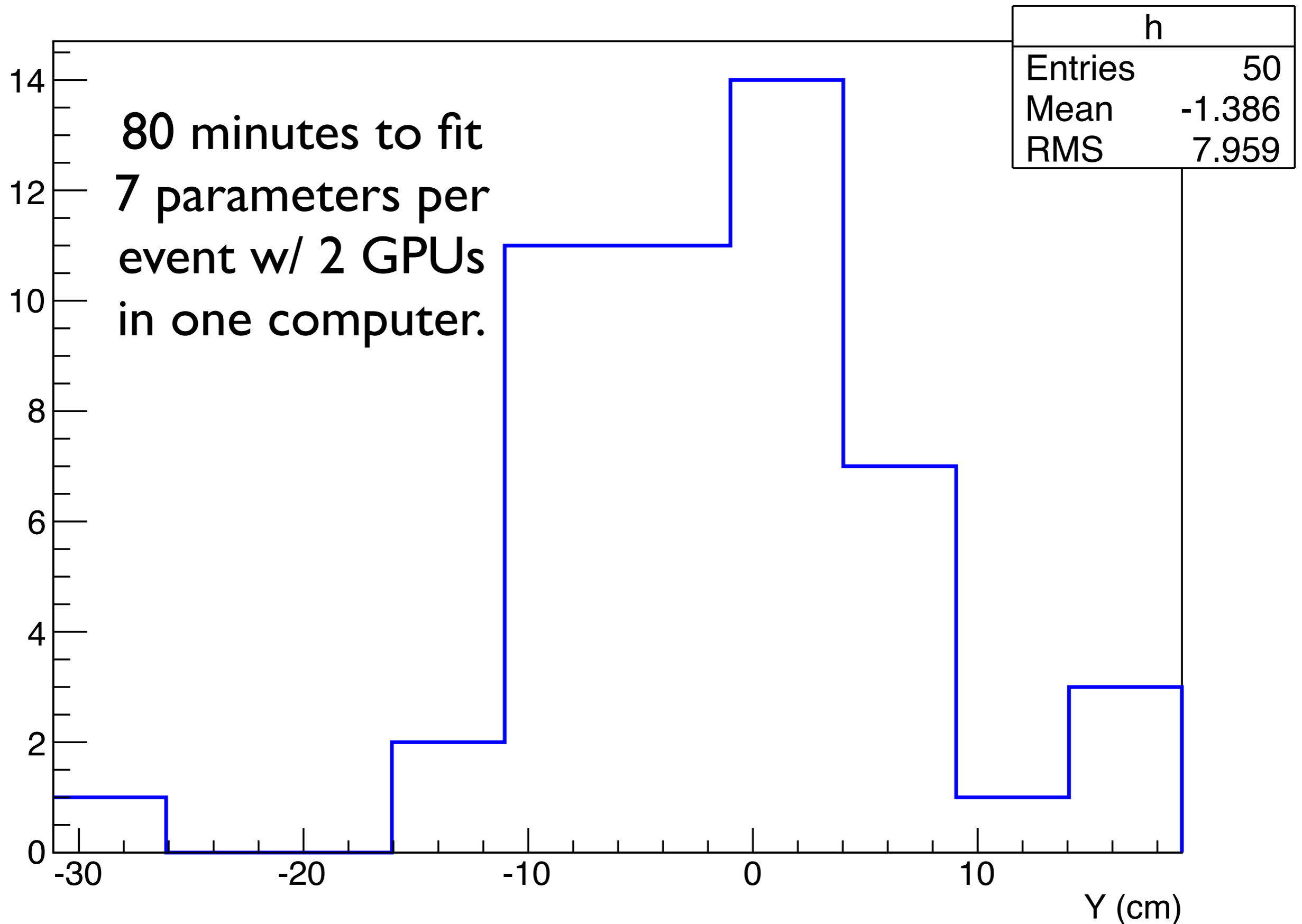
# Minor Challenge: Minimization in a Stochastic Space



# Example Likelihood Scan

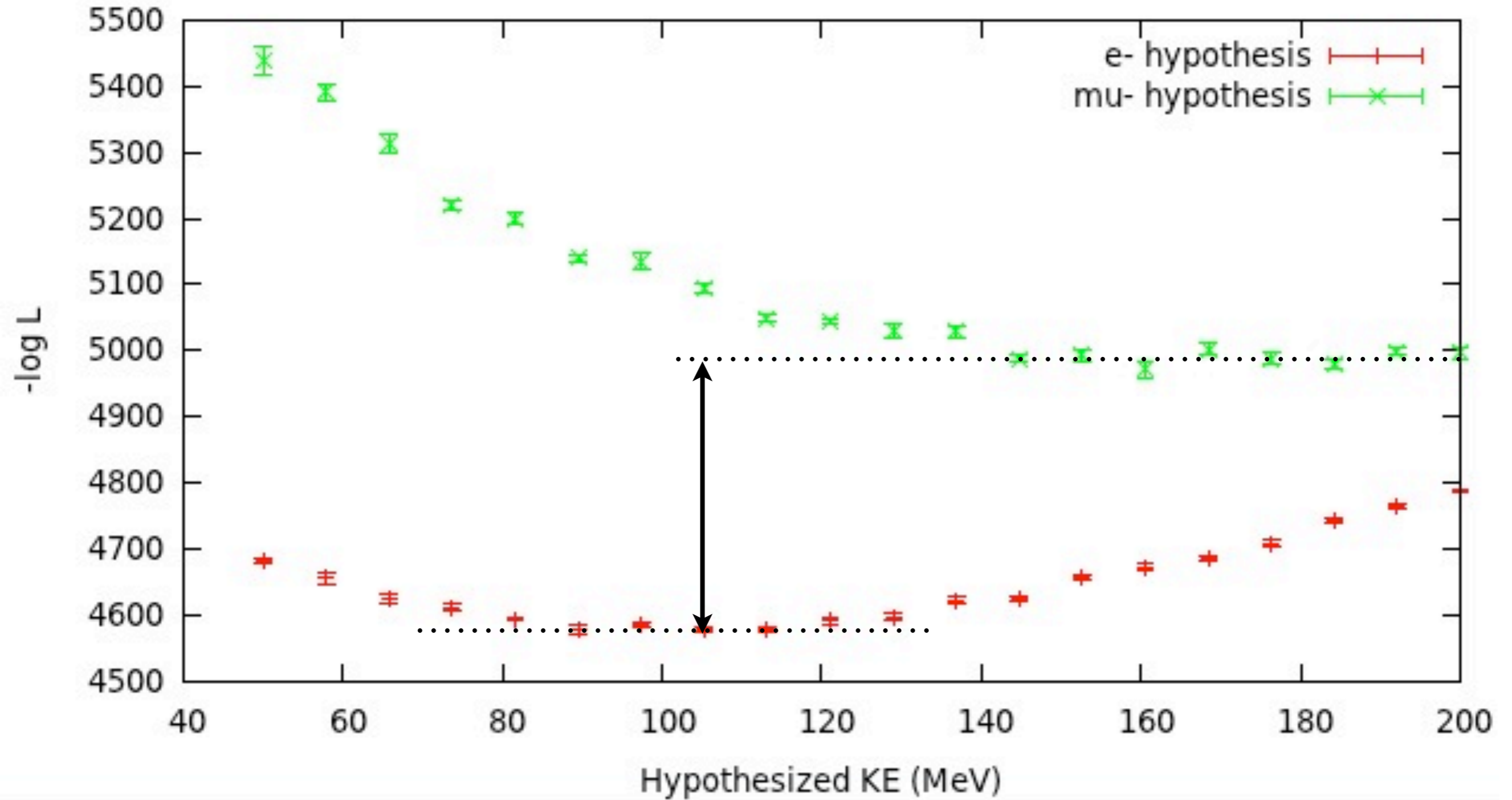


# Ex #1: Electron hypothesis fit to 100 MeV e-

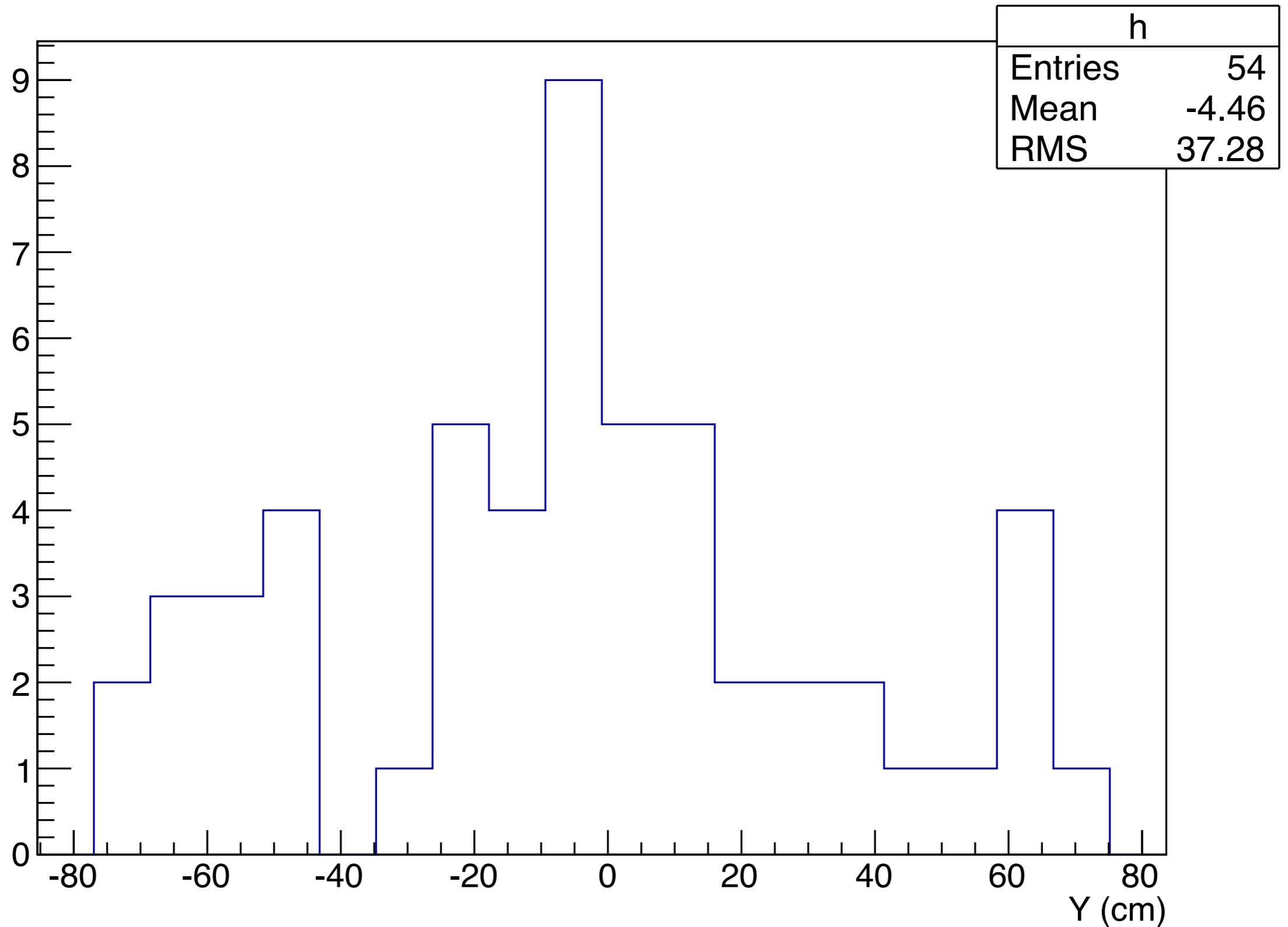


# Ex #2: Particle ID in Water Cherenkov Detector

## 100 MeV e- event



# Ex #3: MICA fit: 20 MeV e<sup>-</sup>



# Recent Developments

- Addition of photon re-emission in bulk and surface materials to support studies with SNO+ and liquid argon detectors.
- Automatic memory “spilling” of detector model from GPU to CPU memory. Good for detectors that don’t fit in the GPU memory. (Ex: 30,000 PMT detector takes ~2.5 GB of memory. Few cards have that much memory.)
- (Almost done) Support for replacing/supplementing photon physics processes with your own processes. Will allow for more complex detector-specific re-emission models or specific scattering models without cluttering up the common case.

# Next Steps

- Make Chroma more end-user focused with better examples, tutorials and documentation.
- Chroma right now is a great photon Monte Carlo and a really fiddly reconstruction tool.
  - Stochastic minimizer is incredibly simple and fragile. Need more function evaluations to do smarter things, but likelihood function takes too long.
  - Need 10-100x speed improvement before Chroma can be easily used as an event fitter. Have ideas for where to get some of this performance.
  - Should explore other PDF estimation techniques beyond the N'th nearest neighbor method used now.
- *Write a paper!*



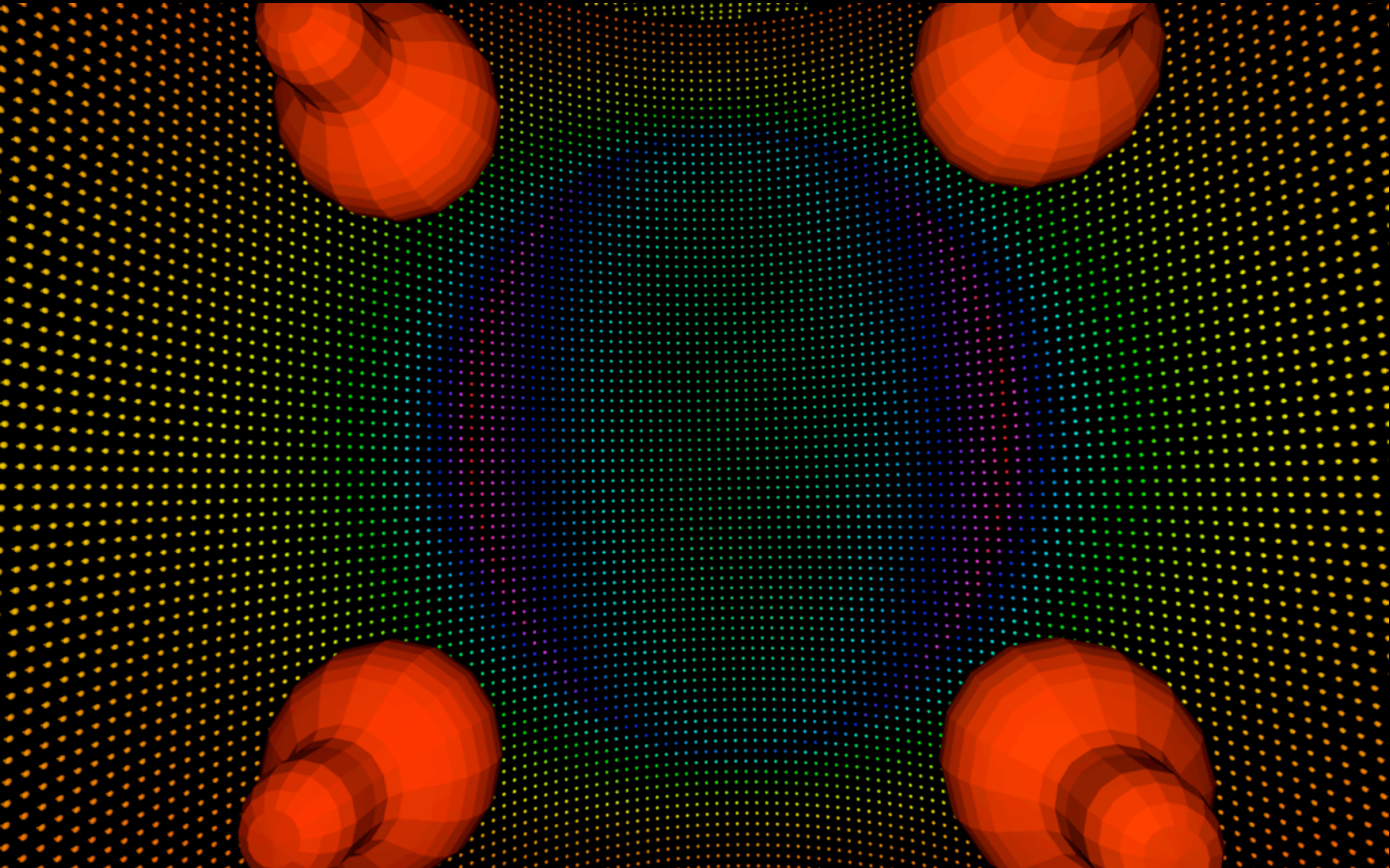
# Summary

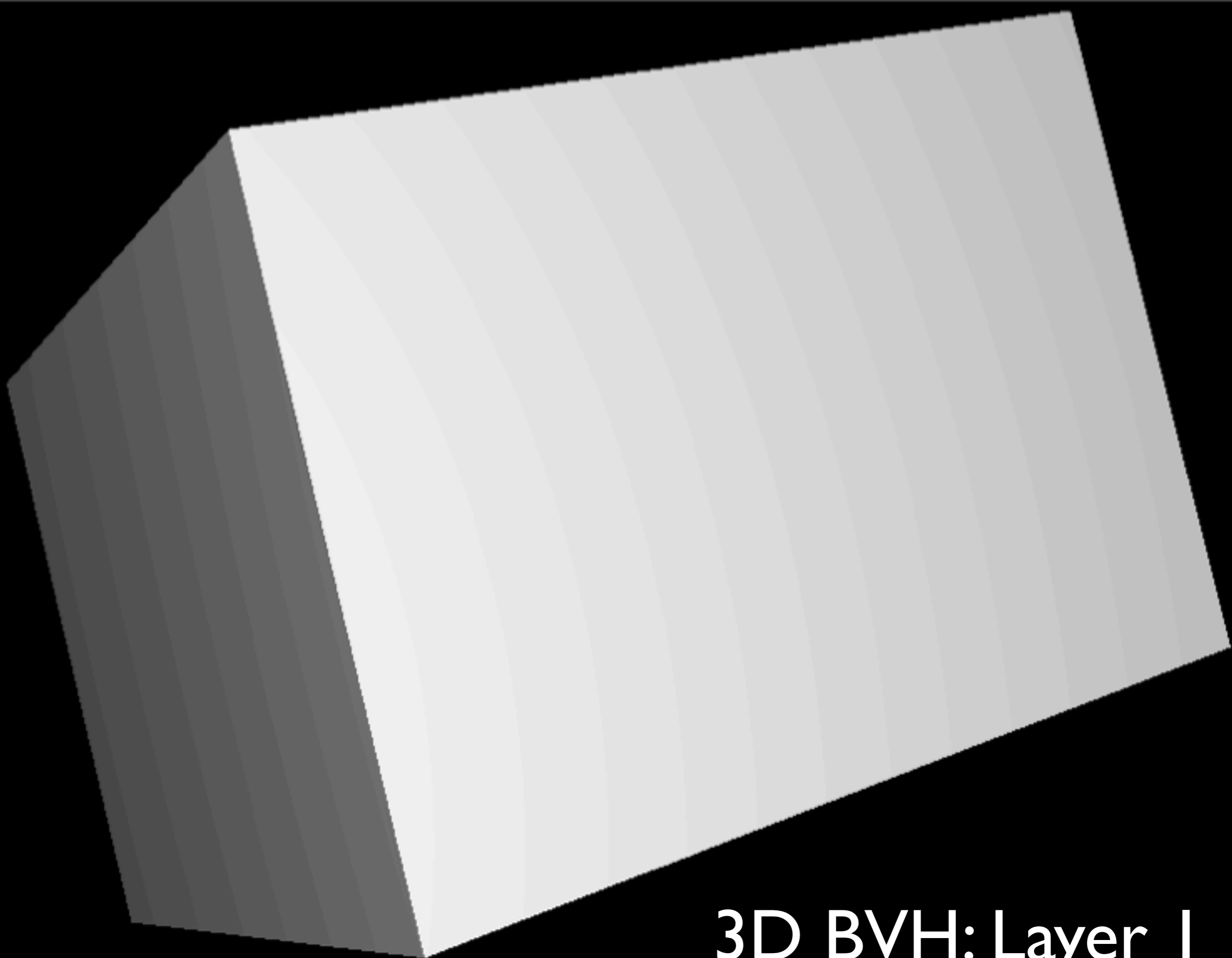
- The next generation of detectors will demand more sophisticated analyses to match their improved capabilities (and reduced funding).
- *A fast Monte Carlo is a very flexible reconstruction algorithm!*
- Chroma was created to explore this concept for photon-based detectors.
- It works, and in ways we didn't initially expect. Chroma can be used for simulation, visualization and reconstruction.
- Pushing hard on the practicality envelope at many hours per event, but in 5 years, this will be a completely standard way to do reconstruction.

<http://chroma.bitbucket.org>

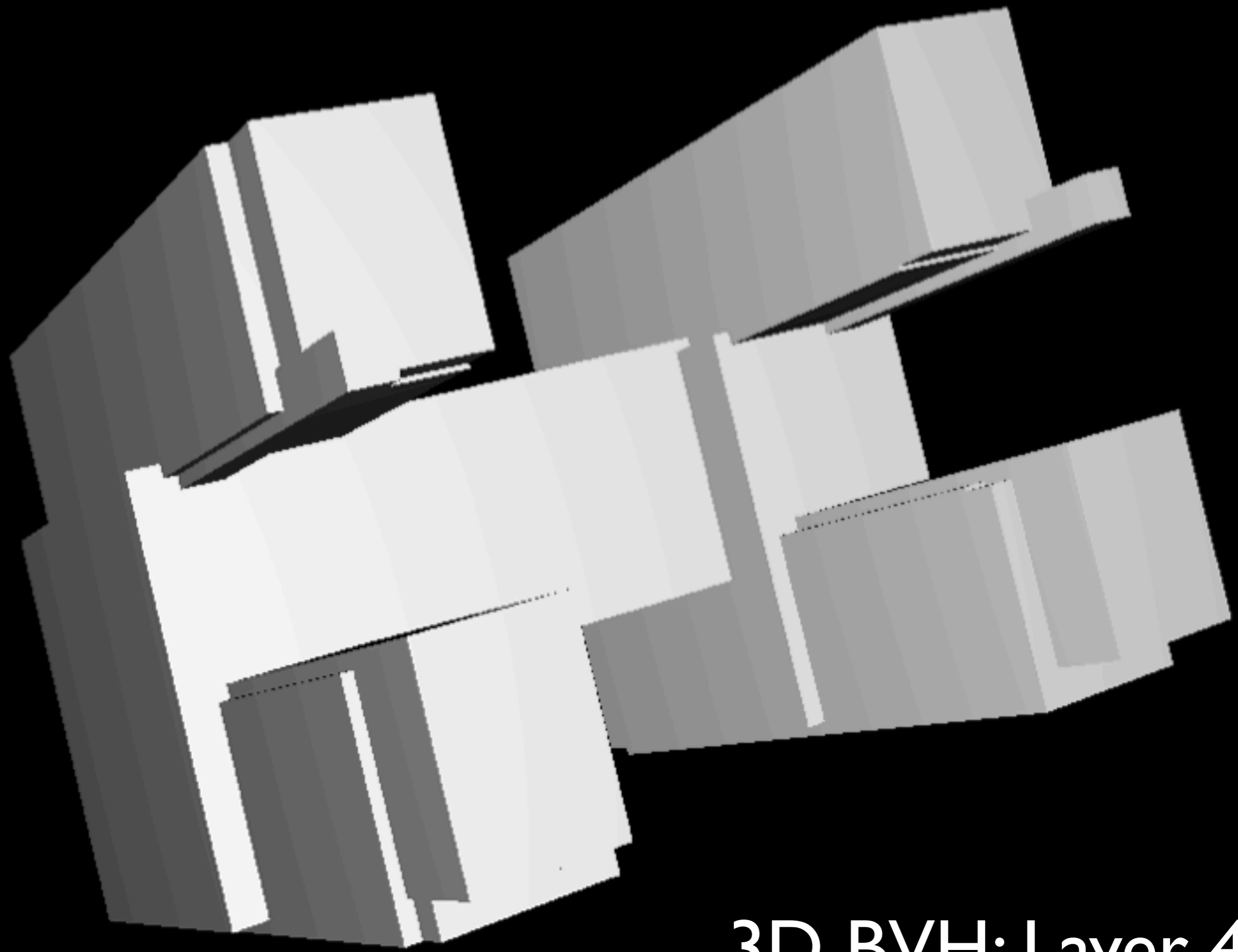
# Backup Slides

# 1 GeV electron hit probability



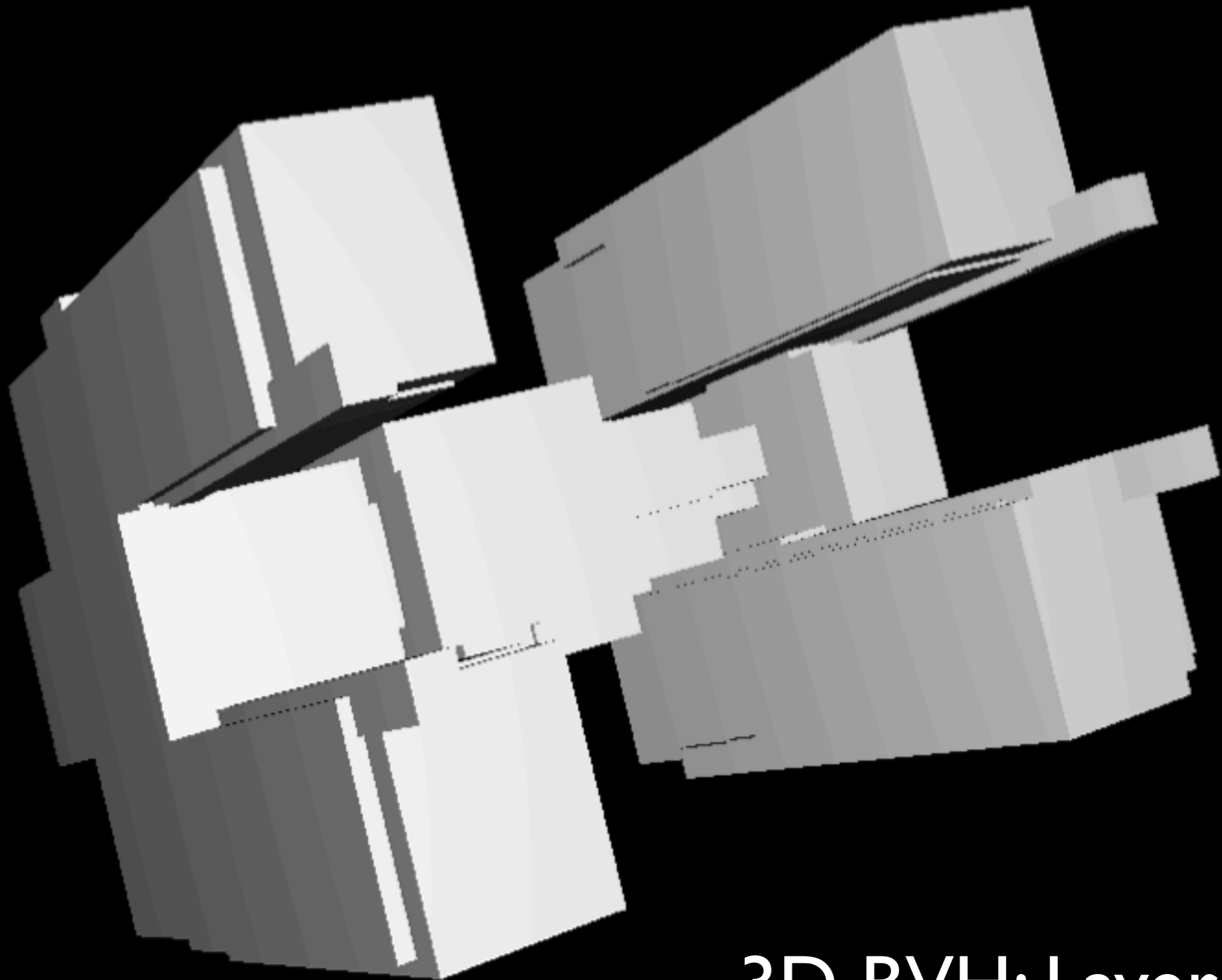


# 3D BVH: Layer I

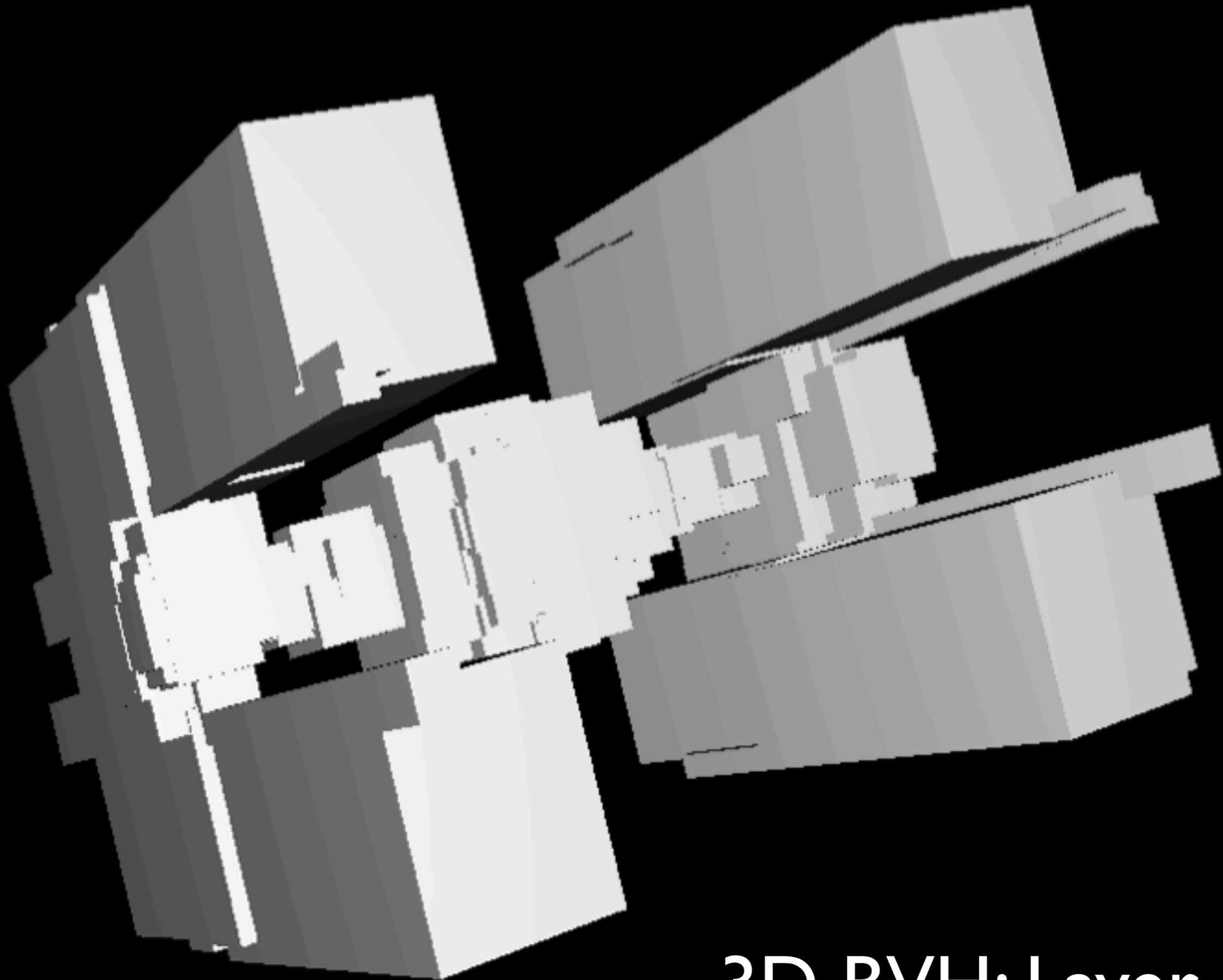


3D BVH: Layer 4

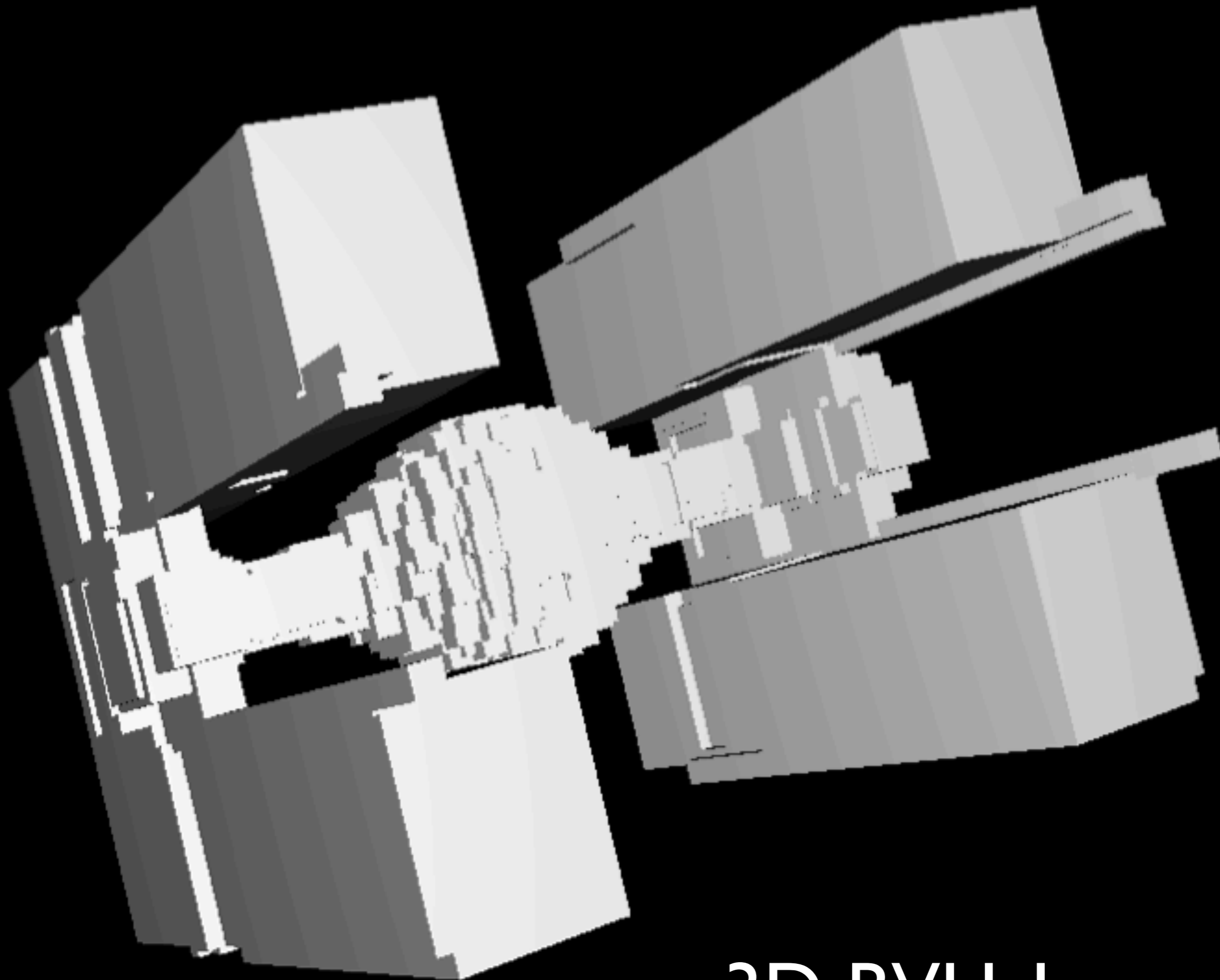




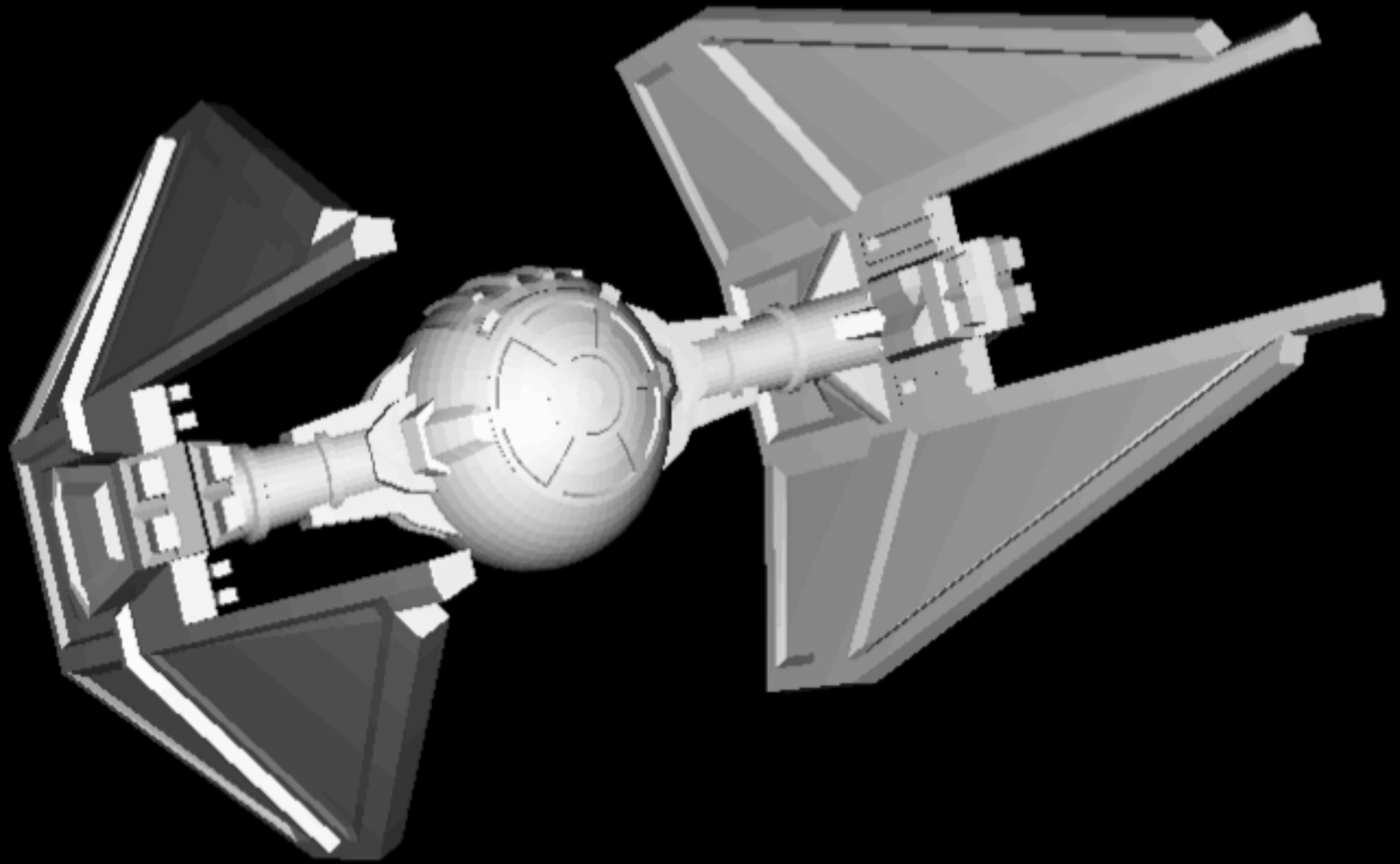
3D BVH: Layer 7



3D BVH: Layer 10



3D BVH: Layer 13



Actual Model

# How is this possible?

Specialization creates efficiency!

All computing architectures now are fundamentally power-limited.

GPUs are *not* general purpose, and therefore can spend their power budget on raw floating point throughput, and wider memory buses.

The optimal usage of a GPU is achieved by a program that:

- Has a high ratio of mathematical operations to memory operations.
- Needs to apply the same instruction to many elements of data in parallel.
- Reads and writes data in a predictably uniform pattern.



# End of Easy Scaling in Silicon

	Pre-2003	Post-2003
Feature Size	1/2x	1/2x
Voltage	1/2x	1x
Switching Energy ( $CV^2$ )	1/8x	1/2x
Clock Frequency	2x	2x
Density	4x	4x
Performance	8x	8x
Power	1x	4x!

Now we have to hold or cut clock rate to control power usage.

For a long time, performance improvements came at no power cost.

Source: William Daly, Stanford  
CSI 93G Lecture,  
May 10, 2010

# Power Usage on a GPU

Task	Power
Floating point multiply-add (0.1 mm <sup>2</sup> , 1.5 GHz)	100 pJ
Move 64 bits on chip a distance of 1 mm	25 pJ
Move 64 bits off chip	1 nJ
Read word from DRAM	20 nJ

**Most power is spent on moving data!**

*Source: William Daly, Stanford  
CSI 93G Lecture,  
May 10, 2010*