Chroma Introduction

Ben Land
Theia Meeting
November 2019

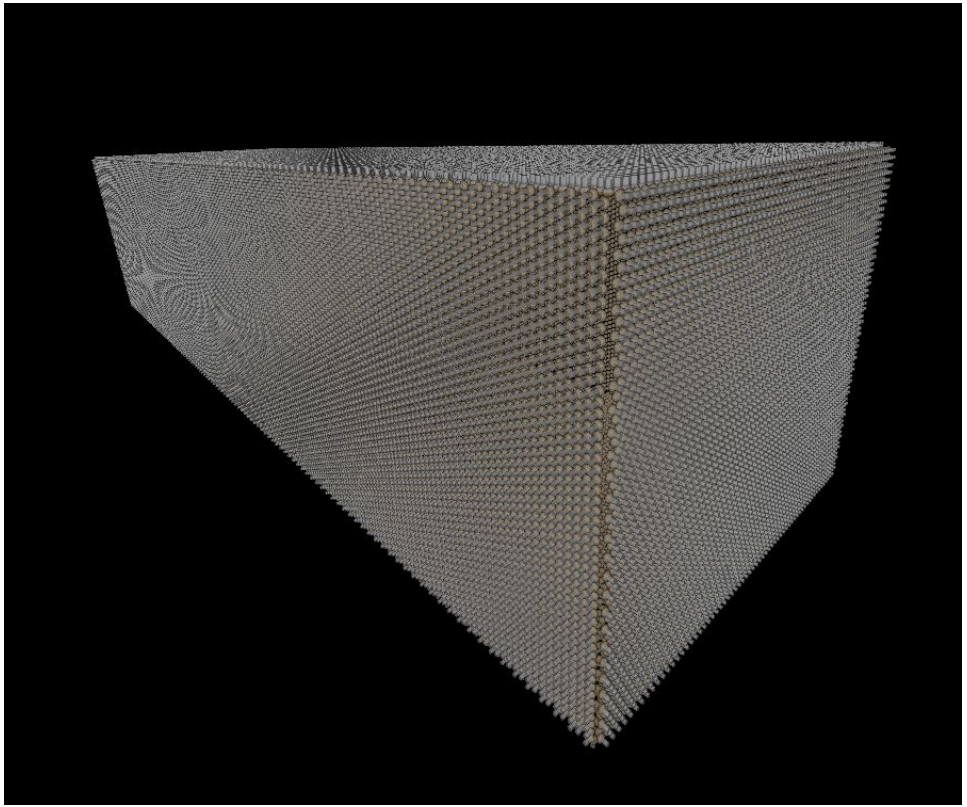
What is Chroma?

- Python package with CUDA code to propagate optical photons on a GPU
  - Written by S. Seibert and A. LaTorre in 2011
- Represents the geometry as a triangular mesh
  - Leverages well developed optical ray tracing algorithms
  - Compare to Geant4 volume-based approach
- Interfaces with Geant4 to generate photons from physics
  - Geant4 geometry can be very simple - just a single volume of target
  - Photons are killed immediately in Geant4, propagated with Chroma
- At least 200x faster than the equivalent Geant4 simulation
  - Does require a reasonable GPU
  - Can be tricky to simulate the physics fast enough for the optics...

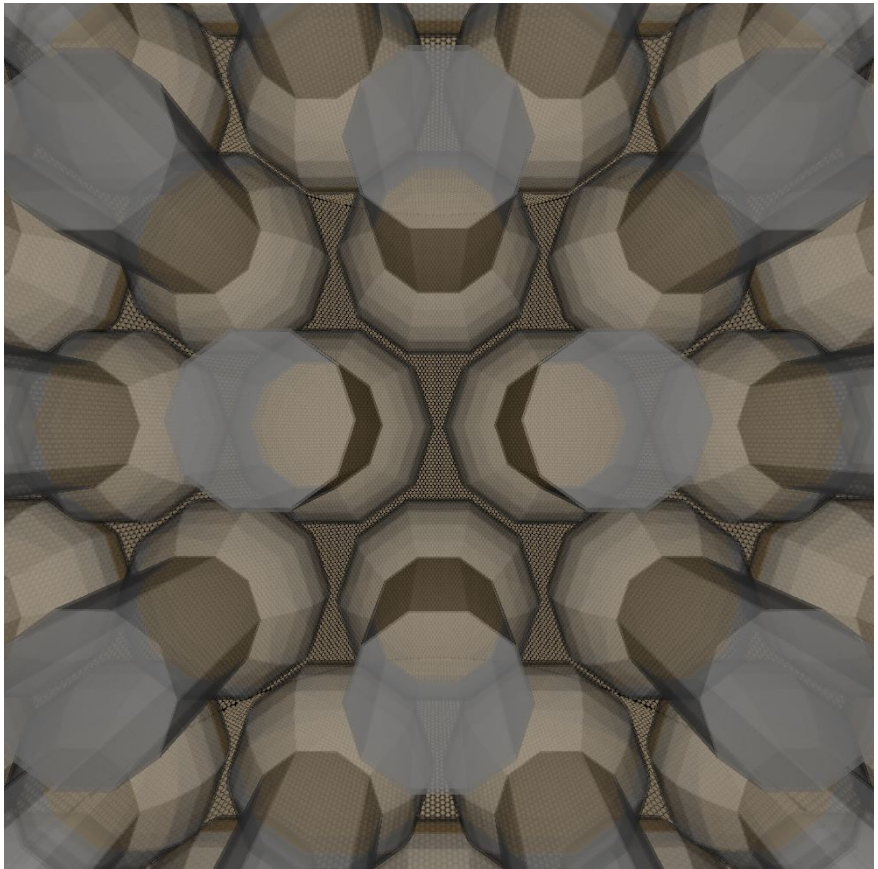
Why use Chroma?

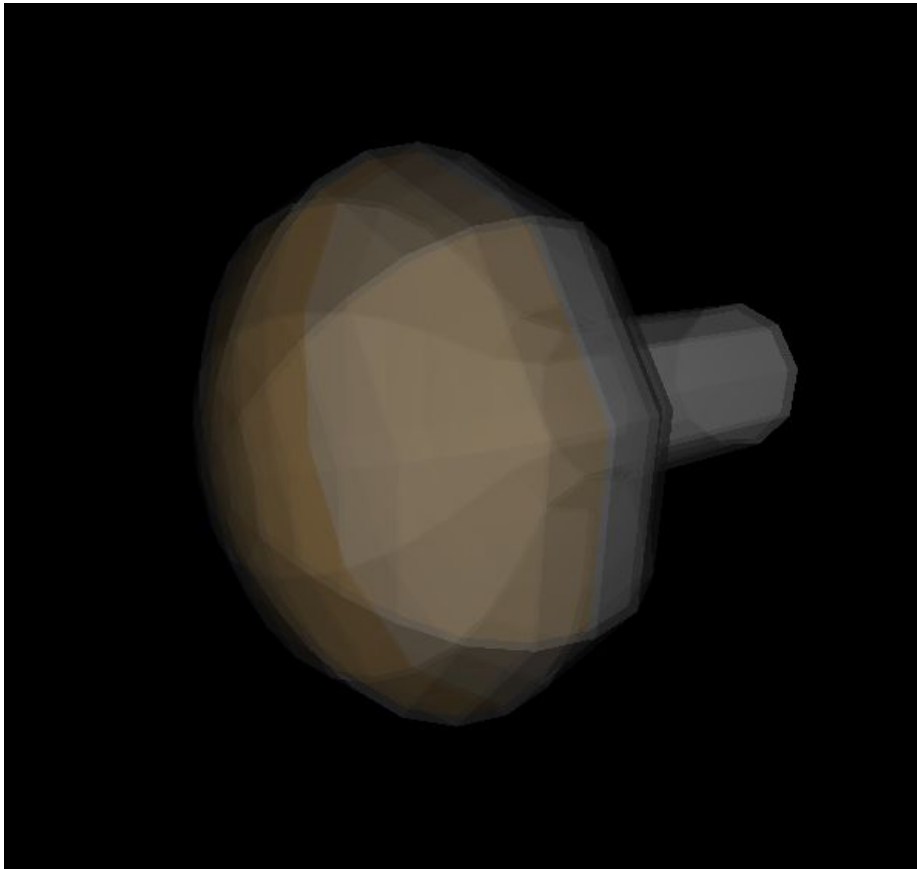
- Specifically for Theia
  - Building a full size Theia100 in Geant4 with PMTs has issues
    - Takes a very long time to initialize O(hour)
    - Uses a lot of system RAM O(10GB)
    - Tracks particles very, very, slowly
      O(all of Gabriel's Savio compute allocation for one solar study)
  - Building a large target volume in Geant4 is fast, tracks fast, minimal memory overhead
  - Chroma can handle the optical part of the simulation (fine detail of PMTs)
    - Still some memory constraints, but appropriate hardware exists
- In general
  - CAD drawings can be directly imported into simulation as STL meshes
  - Fine control over the surface properties of each triangle
  - Relatively small, maintainable codebase
  - Now updated to use most recent Geant4*, Root, Boost, and Python3

Theia25 & Theia100

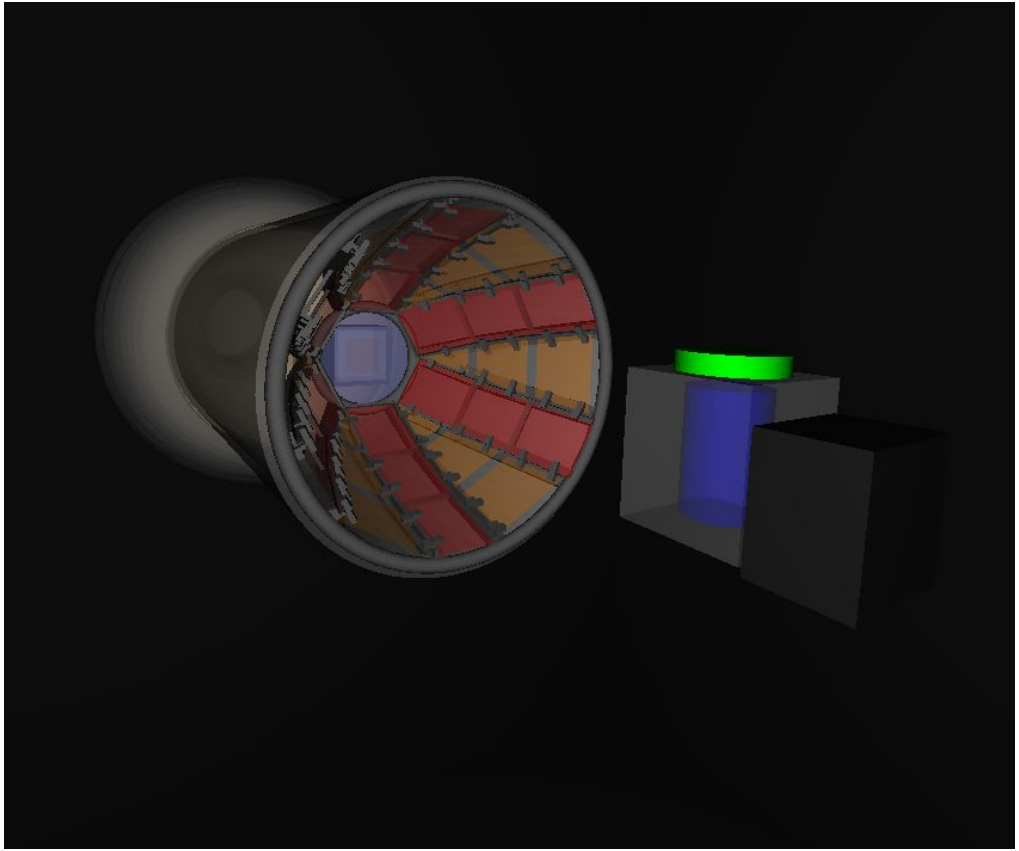


Memory constraints -> low triangle count





Potential for *very* detailed models



Current status

- Developing a dichroicon model in Chroma to reproduce benchtop measurements
- Geant4 needs some upstream changes to work out of the box
- Chroma needs a very new Boost library (shouldn't be an issue for long)
  - Building everything is a bit tricky - instructions pending
- Framework exists to port rat-centric code to a Chroma framework
  - Decay0 already 'ported' (wrapped with boost::python)
  - Lacking a root datastructure for simulations, but python has lots of options

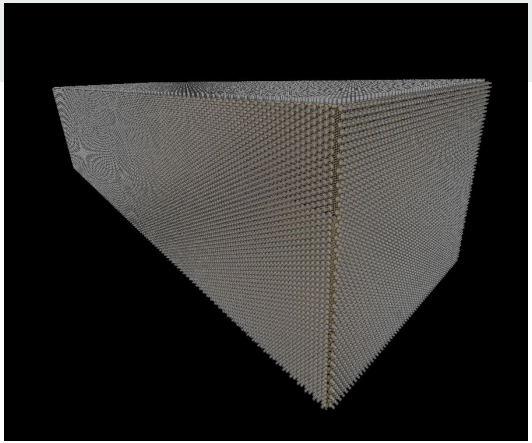
```
import decay0

d0_sr90 = decay0.beta_decay('Sr90')
d0_y90 = decay0.beta_decay('Y90')

def source_event(**kwargs):
    if np.random.random() < 0.5:
        return d0_sr90.gen_event(**kwargs)
    else:
        return d0_y90.gen_event(**kwargs)
```

- Can run physics simulations *now* in full sized Theia geometries with PMTs...

Theia25 simulation with LAB+PPO



- Ran 10 central generated electron events at 10MeV *today*
- 50k total PMTs
- 20k hits / event
- About 4s per event