

The Newtonian Limit in CDT

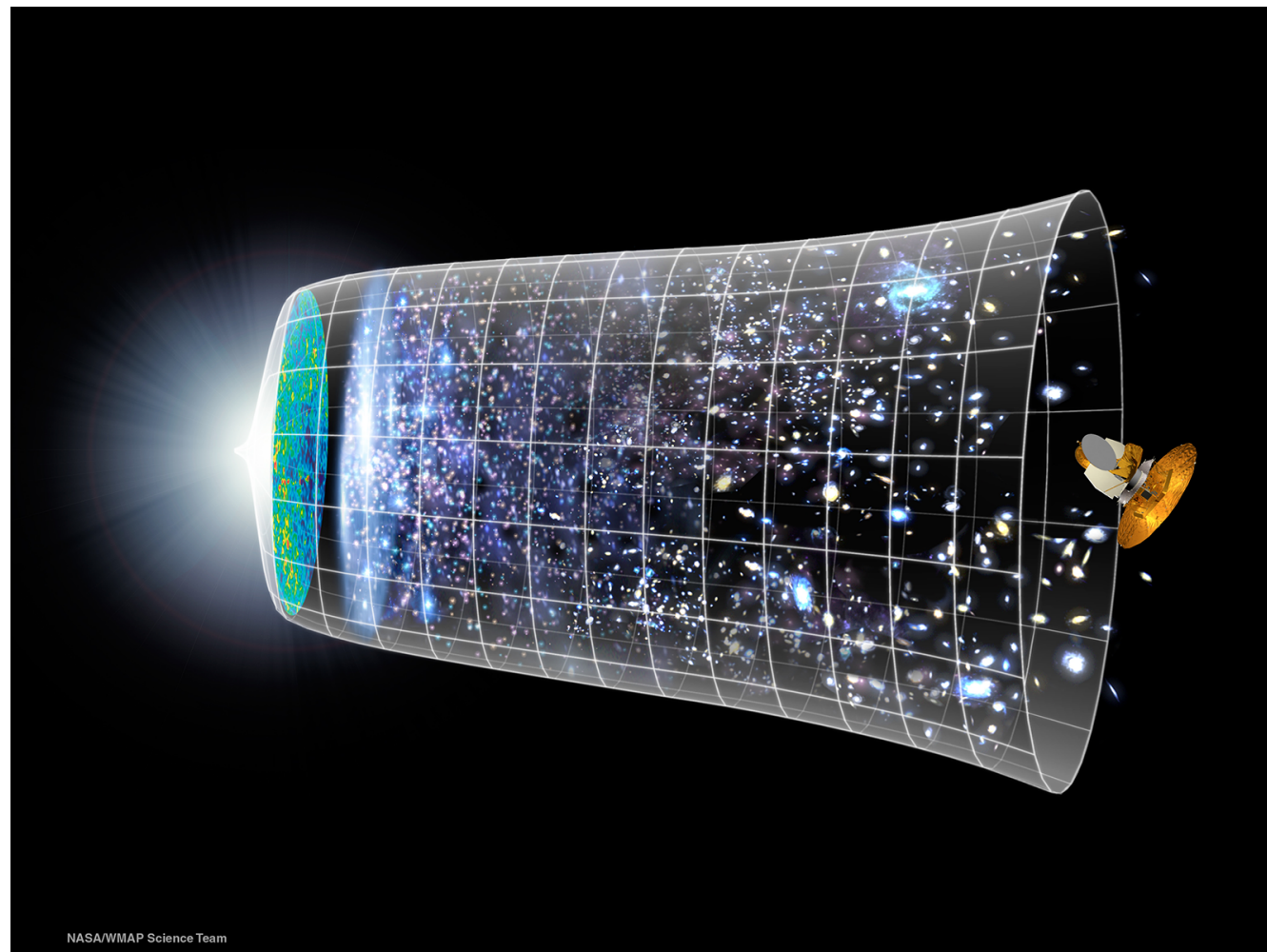
Adam Getchell (UC Davis) acgetchell@ucdavis.edu



Path Integral

$$I = \int \mathcal{D}[g] e^{iS_{EH}}$$

$$S_{EH} = \frac{1}{16\pi G_N} \int d^4x \sqrt{-g} (R - 2\Lambda)$$

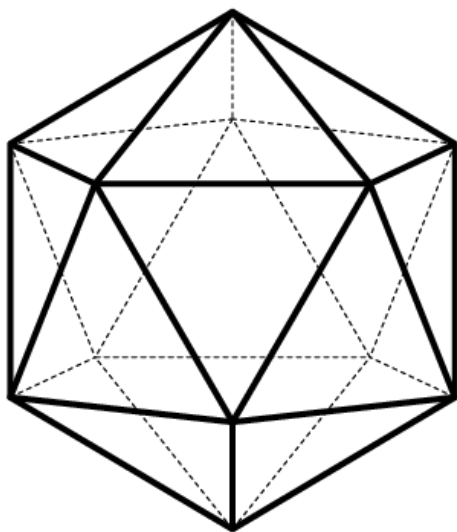


Credit: NASA/WMAP Science Team

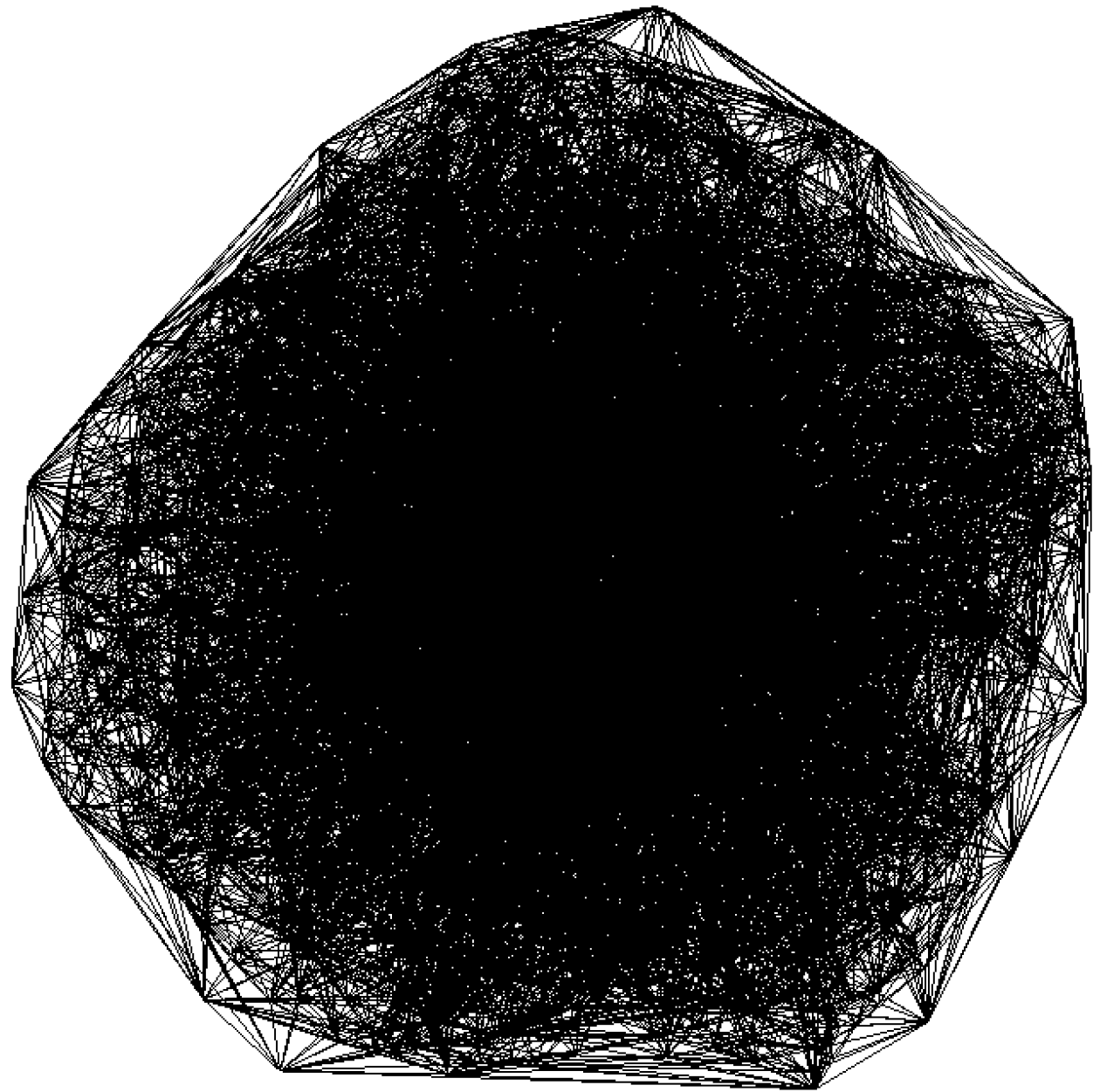
CDT Path Integral

$$I = \sum_{\text{triangulations}} \frac{1}{C(T)} e^{iS_R(T)}$$

$$S_R = \frac{1}{8\pi G_N} \left(\sum_{\text{hinges}} A_h \delta_h - \Lambda \sum_{\text{simplices}} V_s \right)$$



2D Icosahedron: 1 timeslice, 30 Vertices, 20 Simplices



3D Delaunay Triangulation: 256 Timeslices, 7473 Vertices, 47021 Simplices

Does CDT have a Newtonian Limit?

- CDT looks like GR at cosmological scales, does it have a Newtonian limit?

$$F = \frac{Gm_1m_2}{r^2}$$

- At first glance, this is hard:
 - CDT is not well suited for approximating smooth classical space-times
 - We don't have the time or resolution to watch objects fall

A trick from GR

- The static axisymmetric Weyl metric

$$ds^2 = e^{2\lambda} dt^2 - e^{2(\nu-\lambda)} (dr^2 + dz^2) - r^2 e^{-2\lambda} d\phi^2$$

- With two-body solutions

$$\lambda(r, z) = -\frac{\mu_1}{R_1} - \frac{\mu_2}{R_2}, \quad R_i = \sqrt{r^2 + (z - z_i)^2}$$
$$\nu(r, z) = -\frac{\mu_1^2 r^2}{R_1^4} - \frac{\mu_2^2 r^2}{R_2^4} + \frac{4\mu_1 \mu_2}{(z_1 - z_2)^2} \left[\frac{r^2 + (z - z_1)(z - z_2)}{R_1 R_2} - 1 \right]$$

- Leads to a “strut”

$$\nu(0, z) = \frac{4\mu_1 \mu_2}{(z_1 - z_2)^2}$$

- With a stress

$$T_{zz} = \frac{1}{8\pi G} \left(1 - e^{-\nu(r, z)} \right) 2\pi \delta(r)$$

- That can be integrated to get the Newtonian force

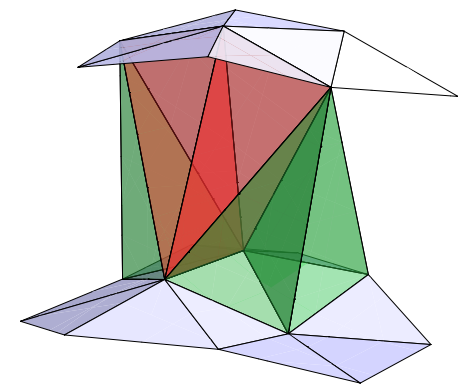
$$F = \int T_{zz} dA = \frac{1}{4G} \left(1 - e^{-\nu(r, z)} \right) = \frac{Gm_1 m_2}{(z_1 - z_2)^2} \quad \text{for } \mu_i = Gm_i$$

Measurements in CDT

Mass \rightarrow Epp quasilocal energy

$$E_E \equiv -\frac{1}{8\pi G} \int_{\Omega} d^2x \sqrt{|\sigma|} \left(\sqrt{k^2 - l^2} - \sqrt{\bar{k}^2 - \bar{l}^2} \right)$$

$$l \equiv \sigma^{\mu\nu} l_{\mu\nu} \quad k \equiv \sigma^{\mu\nu} k_{\mu\nu}$$



- In 2+1 CDT, extrinsic curvature at an edge is proportional to the number of connected tetrahedra
- In 3+1 CDT, extrinsic curvature at a face is proportional to the number of connected pentachorons (4-simplices)

Einstein Tensor in Regge Calculus (Barrett, 1986)

Some Computational Methods

Distance

- Calculate single-source shortest path between the two masses using Bellman-Ford algorithm in $O(VE)$
- Modify allowed moves in a sweep to not permit successive moves which increase or decrease distance

Hausdorff Distance

- Calculate Voronoi diagram of Delaunay triangulation
- Use Voronoi diagram to find minimum Hausdorff distance for sets (Huttenlocher, Kedem and Kleinberg) in $O((m+n)^6 \log(mn))$

My Work

Re-implement CDT

- Rewrite in modern C++
 - C++14 standard
- Use well-known libraries
 - CGAL
 - GeomView
 - Eigen
 - MPFR, GMP
 - Intel TBB

My Work

Use current toolchains

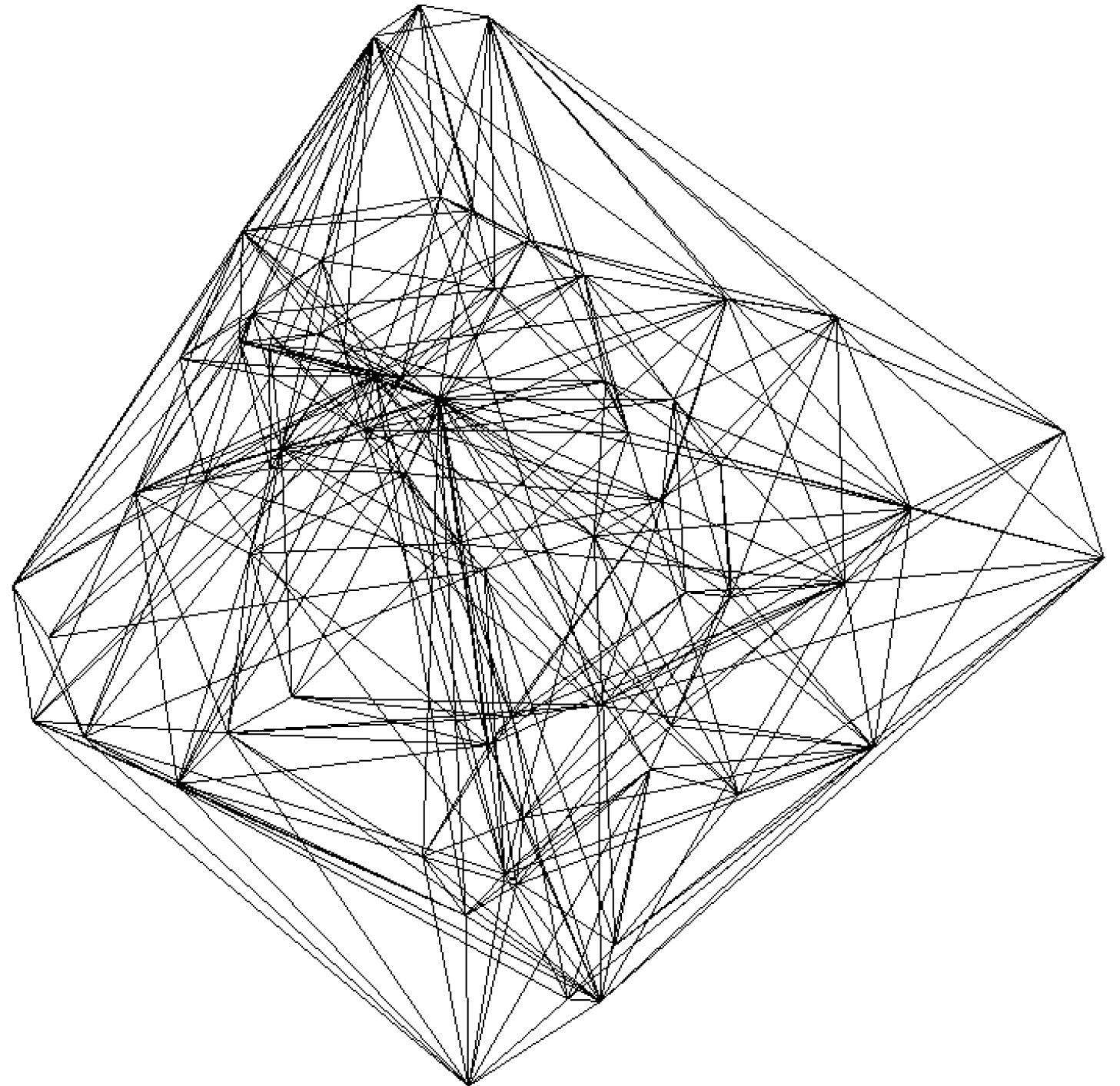
- LLVM/Clang
- Hosted on GitHub
 - Travis-CI for continuous testing
 - GoogleMock
 - CMake for cross-platform building
 - Doxygen for document generation
 - Others

Easy to evaluate, use, and contribute

Fast Foliated Delaunay Triangulations

8 timeslices, 68 vertices,
619 faces, 298 simplices

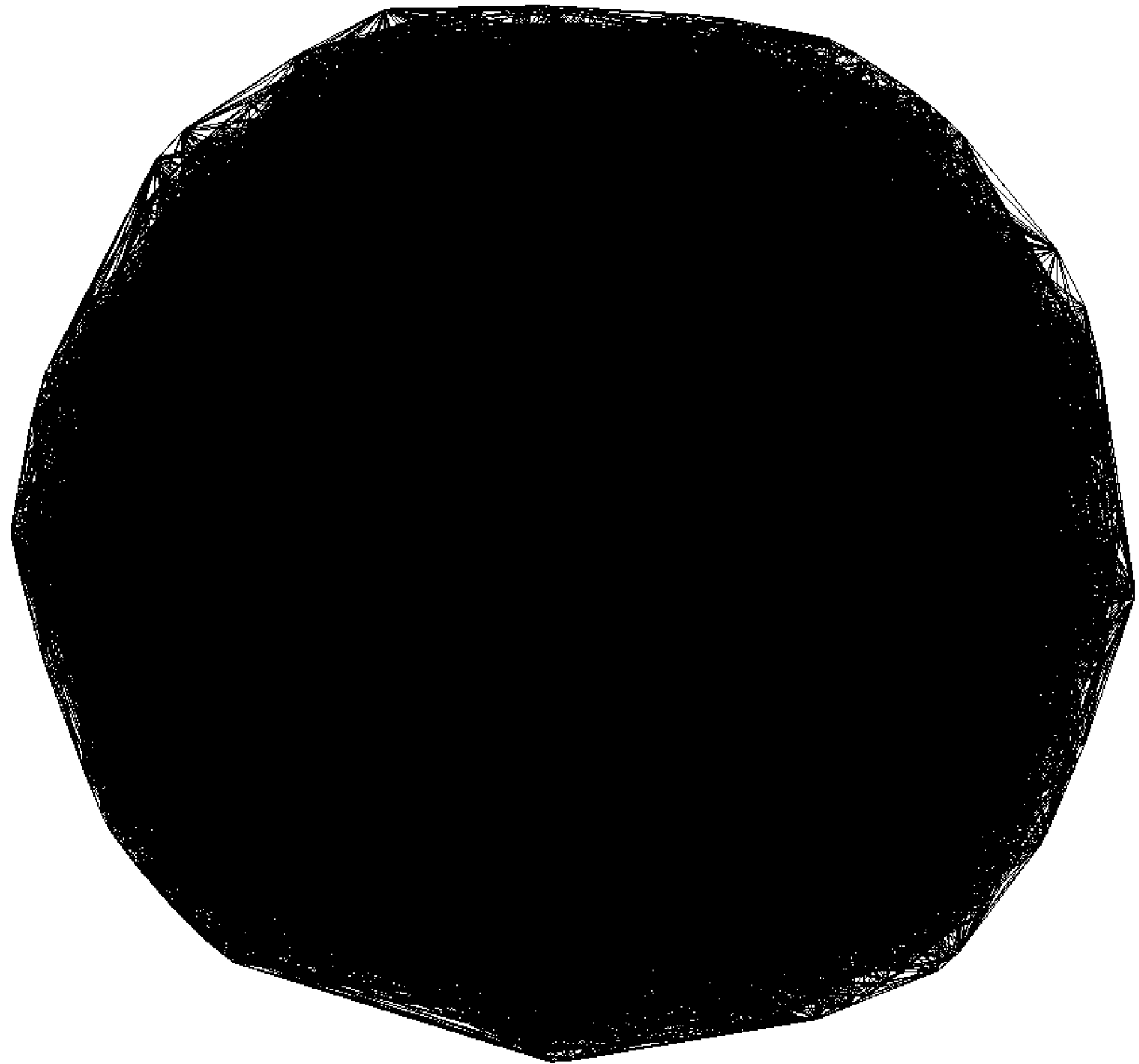
Creation Time: 0.043336s
(MacBook Pro Retina, Mid 2012)



Fast Foliated Delaunay Triangulations

256 timeslices, 222,136 vertices,
2,873,253 faces, 1,436,257 simplices

Creation Time: 284.596s
(MacBook Pro Retina, Mid 2012)



Interested? Please join!

📖 README.md

CDT-plusplus build passing

Quantize spacetime on your laptop.

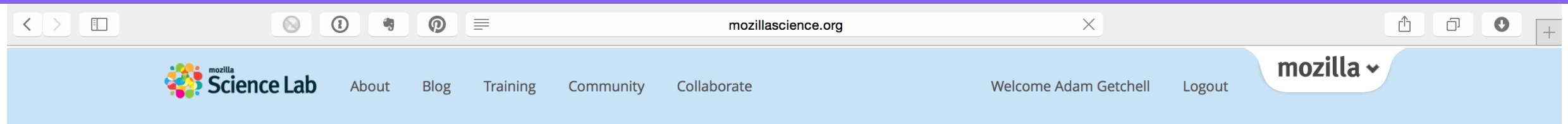
[Causal Dynamical Triangulations](#) in C++ using the [Computational Geometry Algorithms Library](#) and [Eigen](#)>3.1.0, compiled with [CMake](#) using [Clang/LLVM](#). Arbitrary-precision numbers and functions by [MPFR](#) and [GMP](#). [Docopt](#) provides a beautiful command-line interface. [Gmock 1.7](#) may be optionally installed in order to build/run unit tests. [Ninja](#) is a nice (but optional) replacement for `make`. Follows (mostly) the [Google C++ Style Guide](#), which you can check by downloading and running the [cpplint.py](#) script:

```
# python cpplint.py <filename>
```

The goals and targets of this project are:

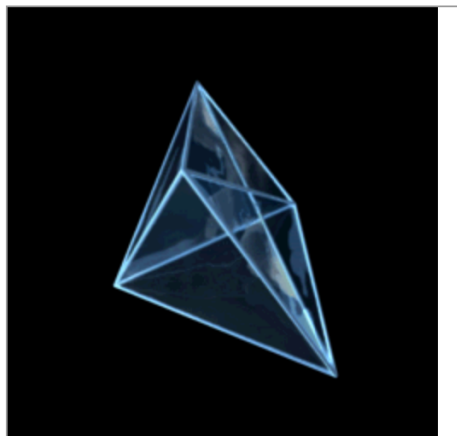
- ☒ Developed with [literate programming](#) generated using [Doxygen](#)
- ☒ Validation tests using [CTest](#)
- ☒ Unit tests with [Gmock](#)
- ☒ Test builds with [Travis CI](#)
- ☒ 3D Simplex
- ☒ 3D Spherical triangulation
- ☒ 2+1 foliation
- ☒ Integrate [docopt](#) CLI
- ☒ S3 Bulk action
- ☐ S3 Boundary action
- ☐ 3D Ergodic moves
- ☐ Metropolis algorithm

Interested? Please join!





Causal Dynamical Triangulations in C++ using CGAL

Physics, General Relativity, Quantum Cosmology



 [Adam Getchell](#)

 University of California Davis

 Active Project

Tags: C++

Looking for: Developers, C++

An attempt to replicate Newtonian gravity in a model of quantum gravity.

WHAT WE'RE DOING

[Causal Dynamical Triangulations](#) is a candidate theory of [quantum gravity](#) in which the smooth geometry of spacetime is replaced by piece-wise flat simplicial geometries using tetrahedrons and their 4D analogues. This allows the gravitational path integral to be computed using numerical methods. We start by developing an efficient way to compute 2+1 and 3+1 universes with large numbers of simplices. We then put in the equivalent of matter into the theory to see if it replicates Newtonian gravity. If it does, CDT becomes a springboard which can be used to examine other questions for which we don't yet have a good means to do so.

PARTICIPANTS

[Website](#)[Go to the Repo](#)[Edit Project](#)

MORE INFORMATION

[Mozilla Science Lab Forum](#)

LICENSE

MAIN DELIVERABLES

- **Implement `Triangulation_vertex_base_with_info.h`** that extends the [TriangulationDSVertex Concept](#) and allows information to be stored in a Vertex.
- **Implement `Triangulation_full_cell_base_with_info.h`** that extends the [TriangulationFullCell Concept](#) and allows information to be stored in a Full Cell.

• Implement the 4D Background geometry (2,3), (4,3), (3,4), (3,3) and their associated [described here](#).