

Visualization of Simulation Data

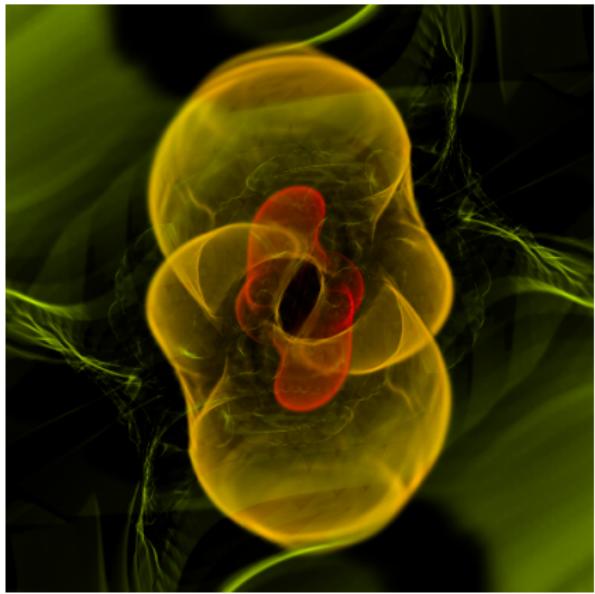
Jonah M. Miller

Los Alamos National Lab

European Einstein Toolkit Workshop
September, 2018

Why Visualize in 3D?

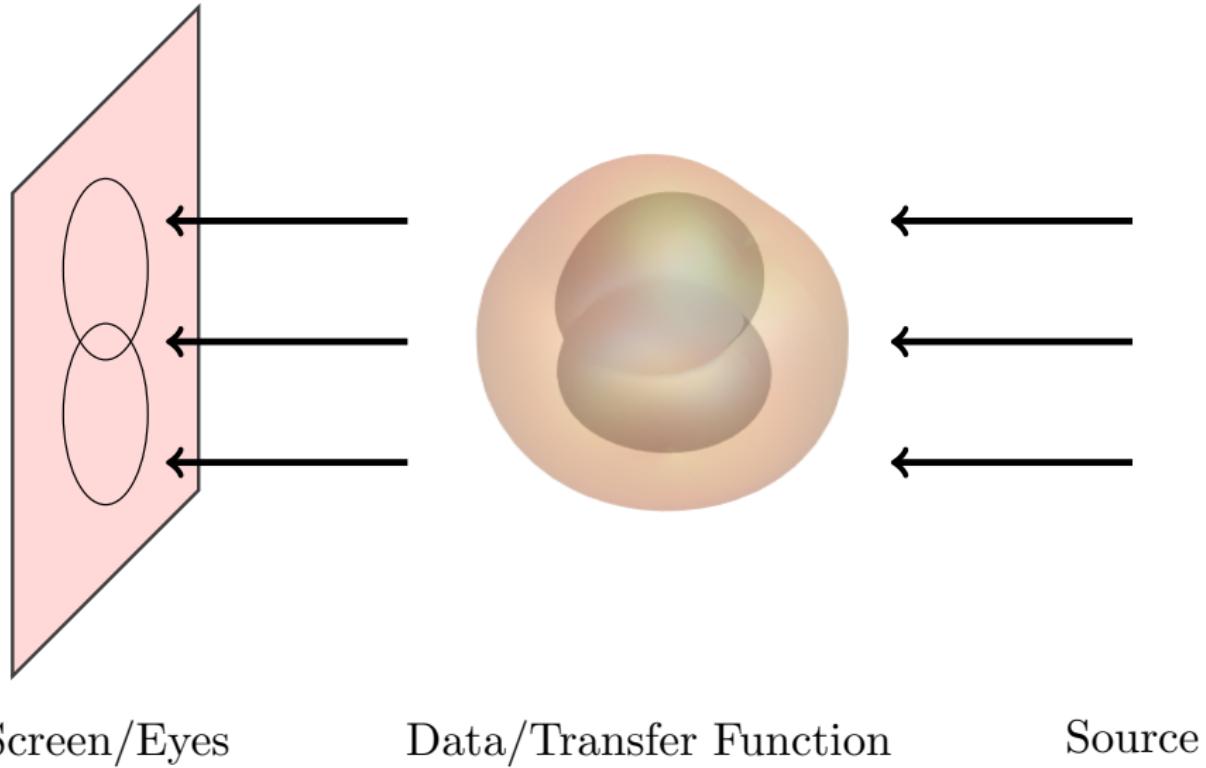
- Eye catching—great for journals, posters, talks, etc.
- 3D structures not visible in slice plots can be visible in 3D volume renders
- Useful for exploring and understanding your own data



What You Will Learn

- ① (Very roughly) how does volume rendering work?
- ② How to put your simulation data into these tools
 - Your code has one representation of the physics, the data file has another, the visualization tool yet another.
 - Mapping these representations into each other can be hard.
 - We'll do it for:
 - Einstein Toolkit data
 - Arbitrary simulation data
- ③ Some tips and tricks
 - Visualization best practices
 - Idiosyncracies of each tool

Volume Rendering and Ray Tracing



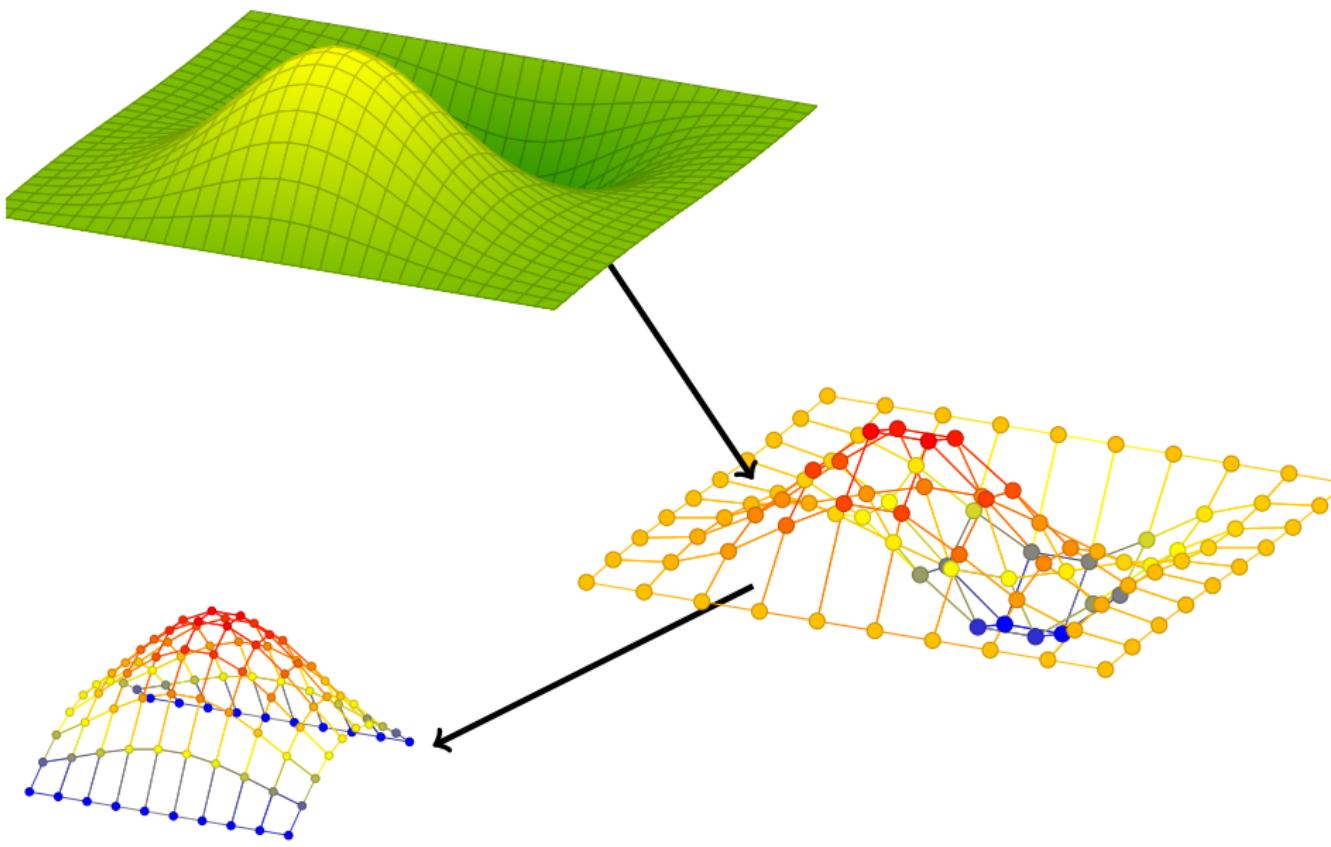
Screen/Eyes

Data/Transfer Function

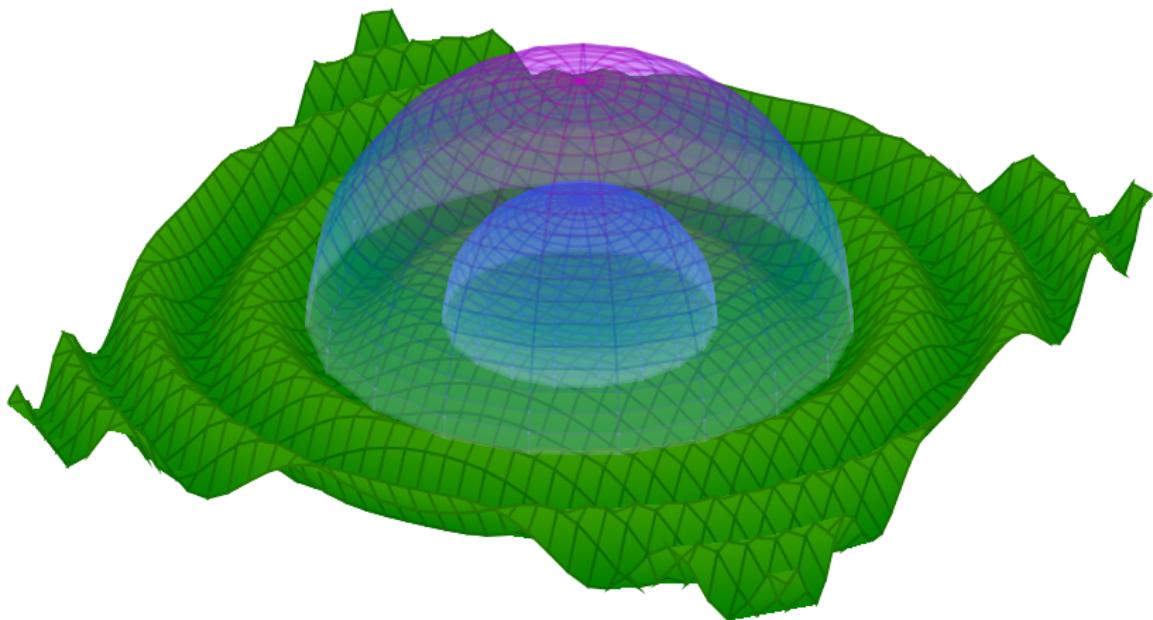
Source

Mapping Your Simulation Data Into a Visualization Tool

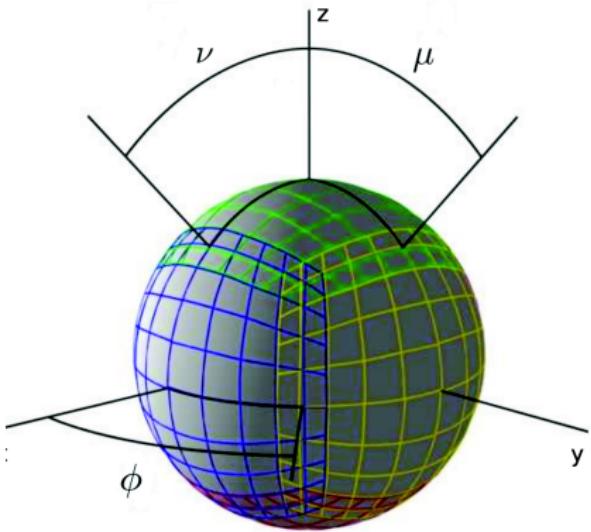
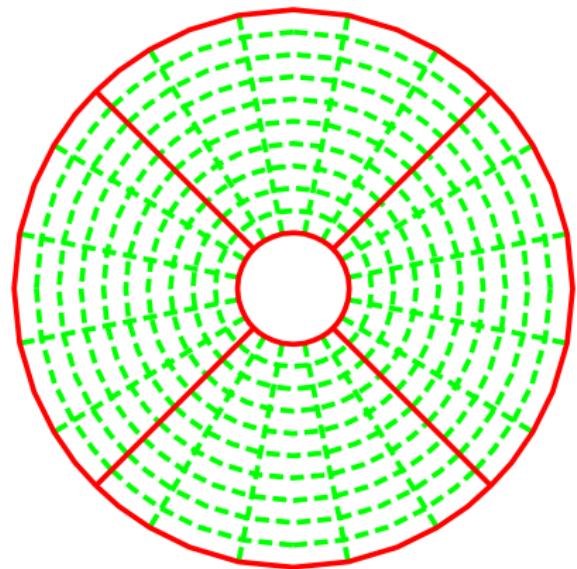
What do we care about in a simulation?



What do we care about in a simulation?



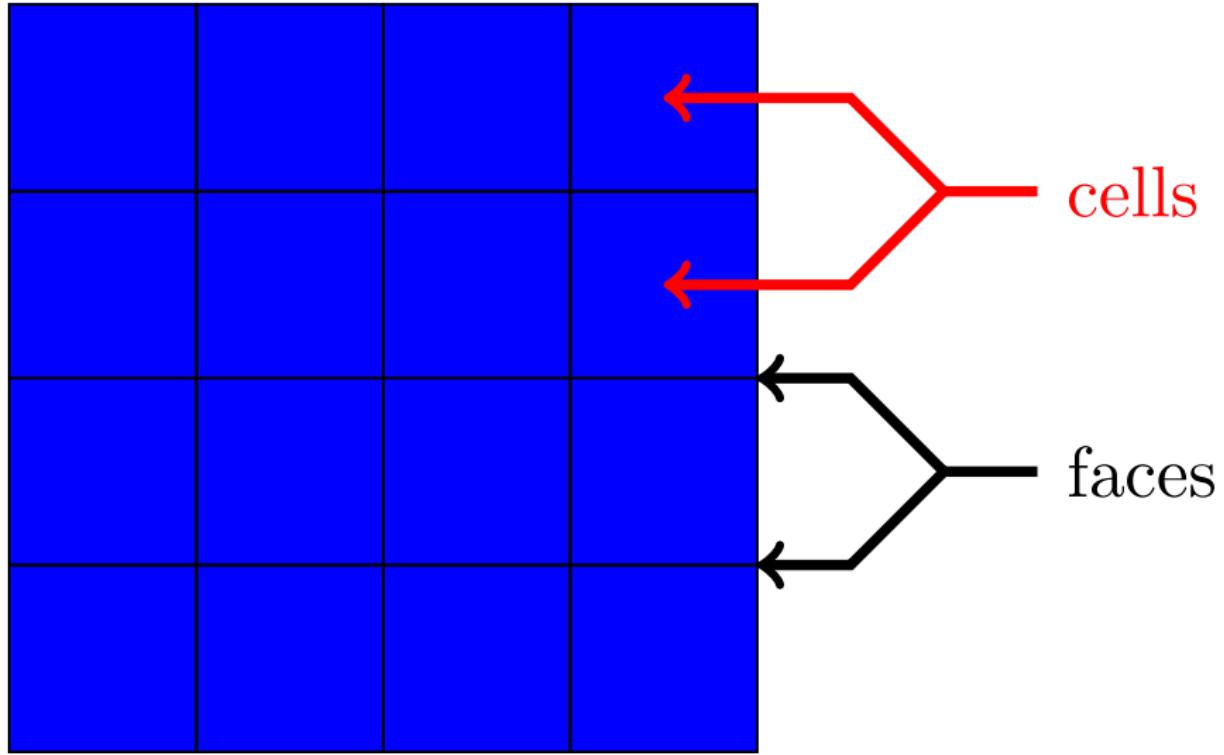
What do we care about in a simulation?



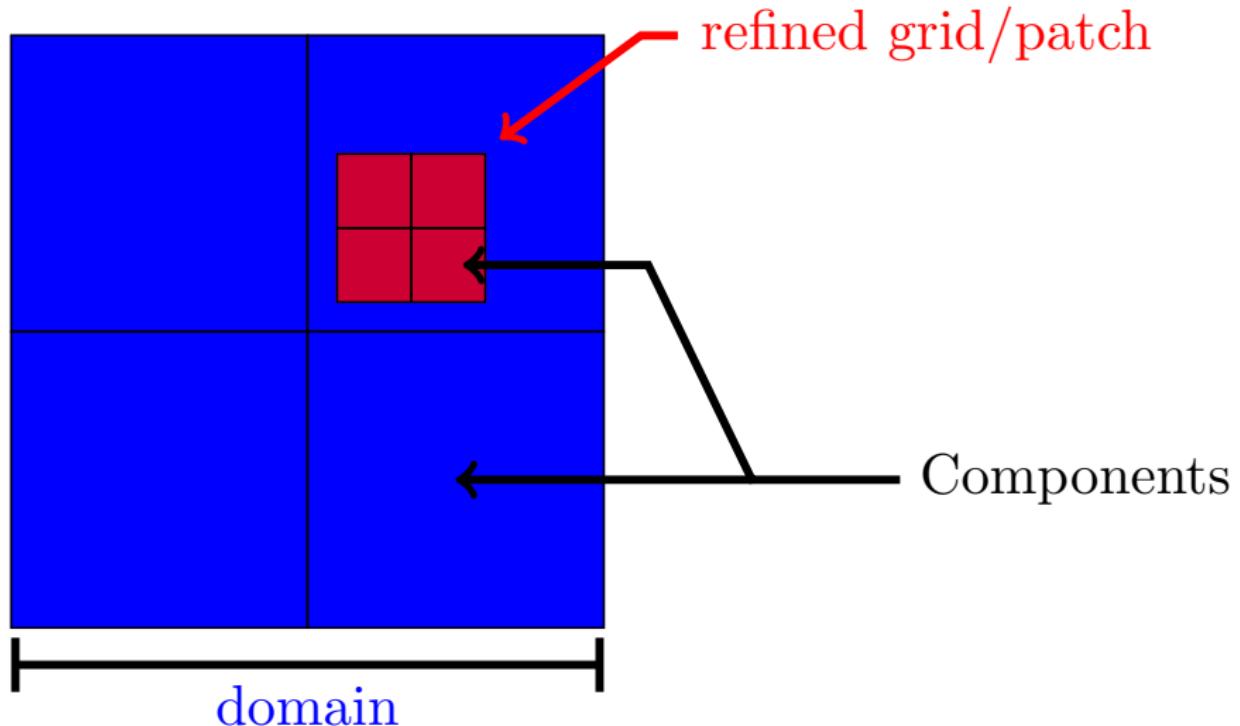
Schnetter *et al* 2006 *Class. Quantum Grav.* **23** S553

Pollney *et al* 2011 *Phys. Rev. D* **83** 044045

Important Finite Volume Concepts



Important Carpet AMR Concepts



Visualization Tools



- **Pros:** Simple. Easy to explore/inspect data.
 - **Cons:** Limited. 3D visualization hard.
-



- **Pros:** Incorporates meaningful physics and analysis. Hackable. Parallel.
 - **Cons:** Less flexible. No interactive UI.
-

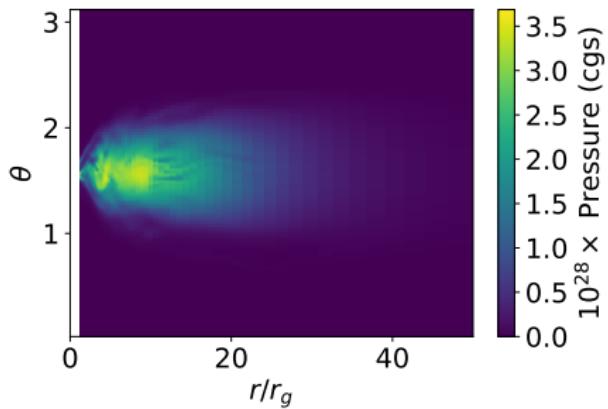


- **Pros:** Very flexible. Contains graphical UI.
- **Cons:** Opaque. Hard to use in parallel.

Pure Python

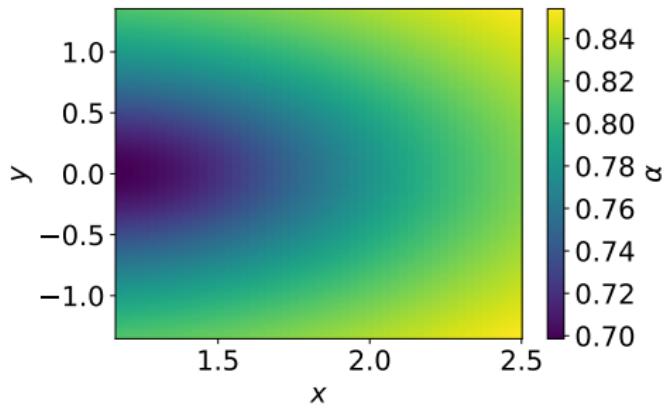
Exercise: Inspecting Arbitrary Data in Python

```
import h5py
import numpy as np
data = {}
with h5py.File('harmdisk2d/data.h5', 'r') as f:
    for k,v in f.items():
        data[k] = v.value
```



Exercise: Inspecting Carpet AMR Data in Python

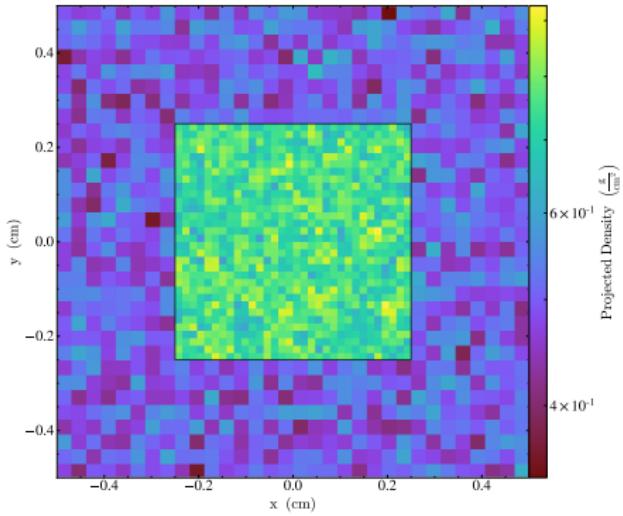
```
lapse_filenames = sorted(glob('admbase-lapse.*.h5'))
lapse = {}
for filename in lapse_filenames:
    with h5py.File(filename, 'r') as f:
        for k,v in f.items():
            try:
                lapse[k] = v.value
            except:
                pass
```



yt

(Optional) Exercise: Installing yt

- pip install yt -user



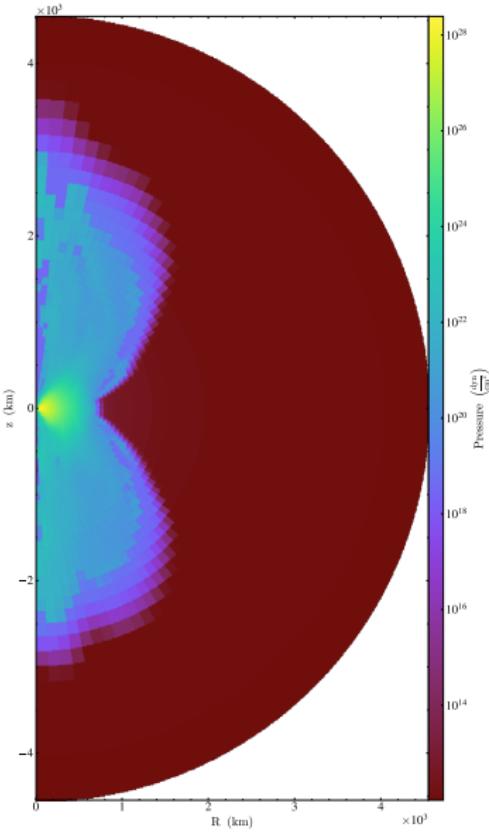
Exercise: Loading Curvilinear Data into yt

- **Advantages**

- Easy plotting and data analysis
- More advanced than pure python

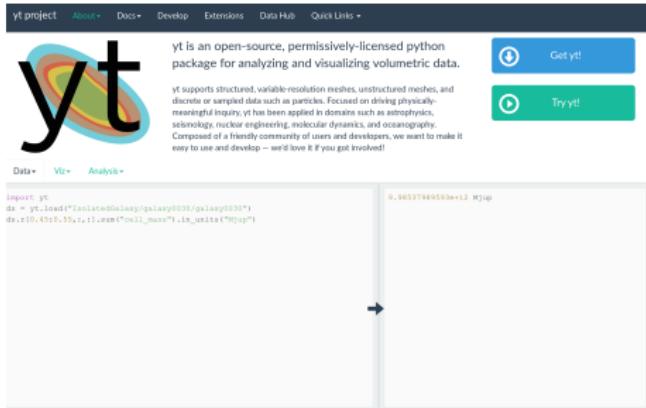
- **Limitations**

- No volume rendering or projection plots available
- Ad hoc—parallelism unlikely to work



yt's Best Feature: The Community

- Website: yt-project.org
- IRC:
yt-project.org/irc.html
- Slack
- Mailing list:
yt-users@python.org



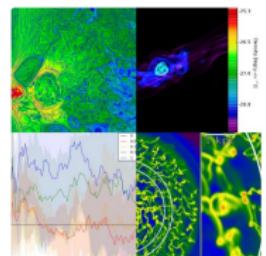
The screenshot shows the homepage of the yt project website. At the top, there is a navigation bar with links for 'yt project', 'About', 'Docs', 'Develop', 'Extensions', 'Data Hub', and 'Quick Links'. Below the navigation bar is a large logo consisting of a stylized 'yt' monogram with a multi-colored elliptical background. To the right of the logo, a text block states: 'yt is an open-source, permissively-licensed python package for analyzing and visualizing volumetric data. It supports structured, variable-resolution meshes, unstructured meshes, and discrete or sampled data such as particles. Focused on driving physically-meaningful inquiry; yt has been applied in domains such as astrophysics, seismology, nuclear engineering, molecular dynamics, and oceanography. Composed of a friendly community of users and developers, we want to make it easy to use and develop — we'd love it if you got involved!'. Below this text are three buttons: 'Get yt!' (blue), 'Try yt!' (green), and a 'Documentation' link. At the bottom of the page, there is a section titled 'Learn: getting started.' with a 'Get Started' button, followed by a paragraph of text and several small visual examples of data analysis results.

Learn: getting started.

To get started using yt to explore data, we provide resources including [documentation](#), [workshop material](#), and even a fully-executable [quick start guide](#) demonstrating many of yt's capabilities.

But if you just want to dive in and start using yt, we have a long list of [recipes](#) demonstrating how to do various tasks in yt. We even have [sample datasets](#) from all of our supported codes on which you can test these recipes. While yt should just [work](#) with your data, here are some [instructions](#) on loading in datasets from our supported codes and formats.

[Get Started](#)

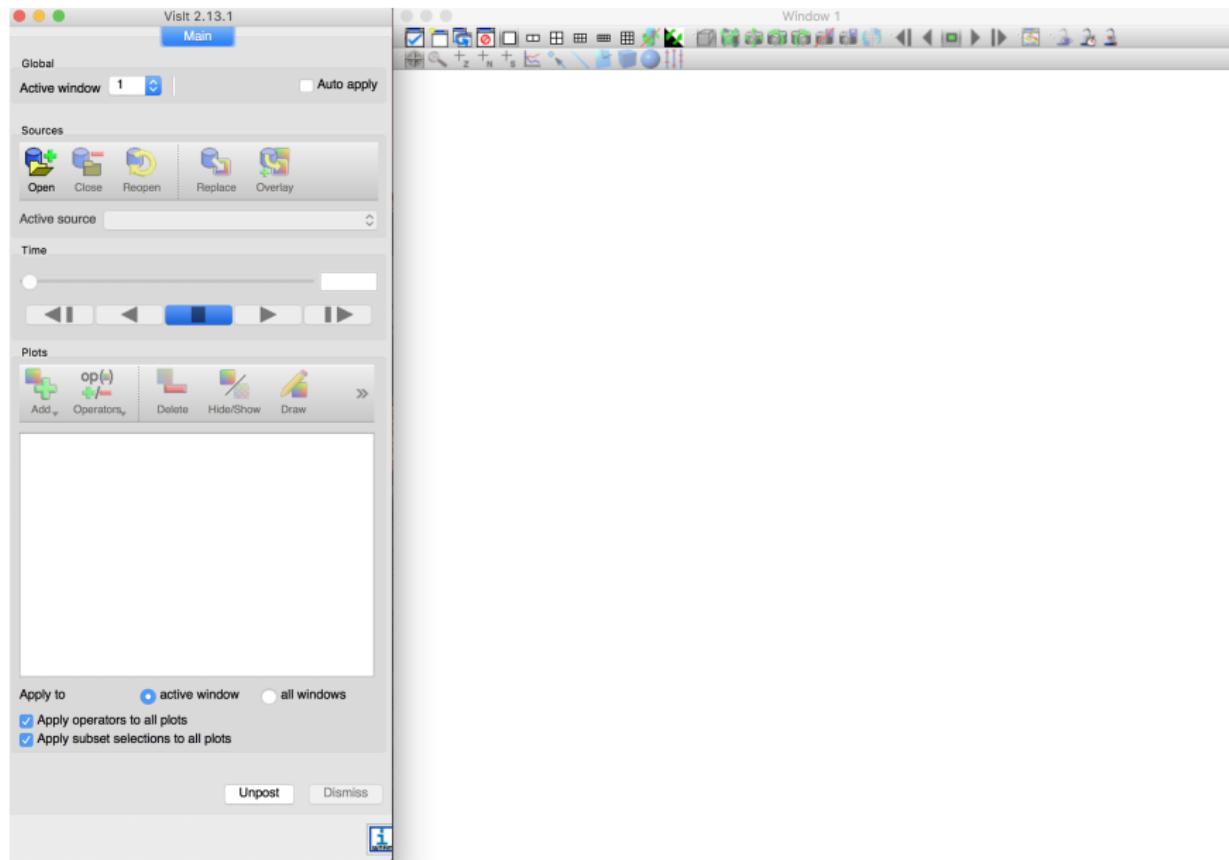


Help Wanted

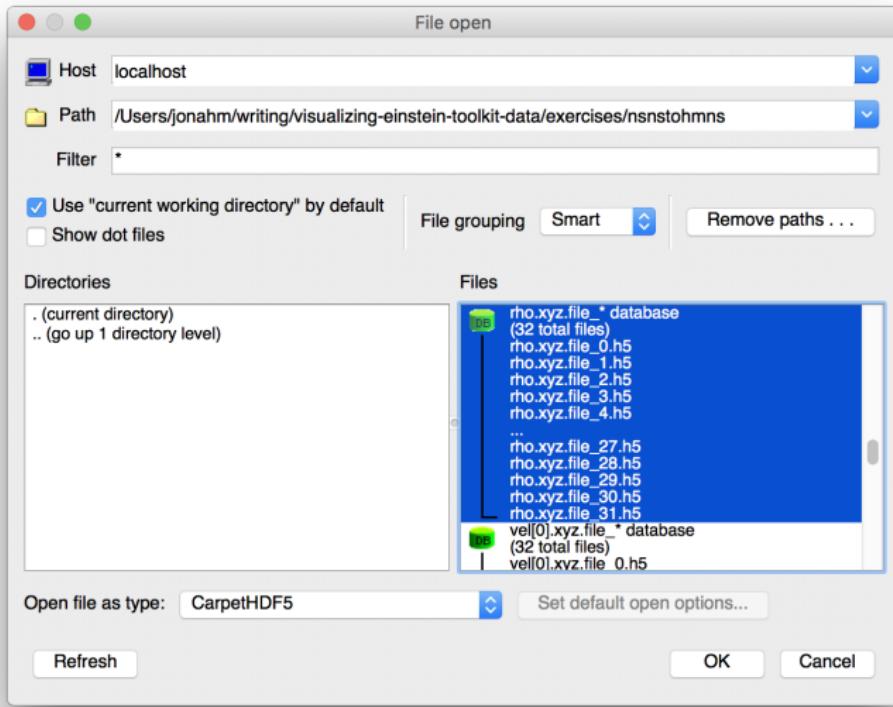
- Carpet AMR and yt are not easy to integrate
- Ideally a frontend for yt can be written to integrate it with Carpet

Visit

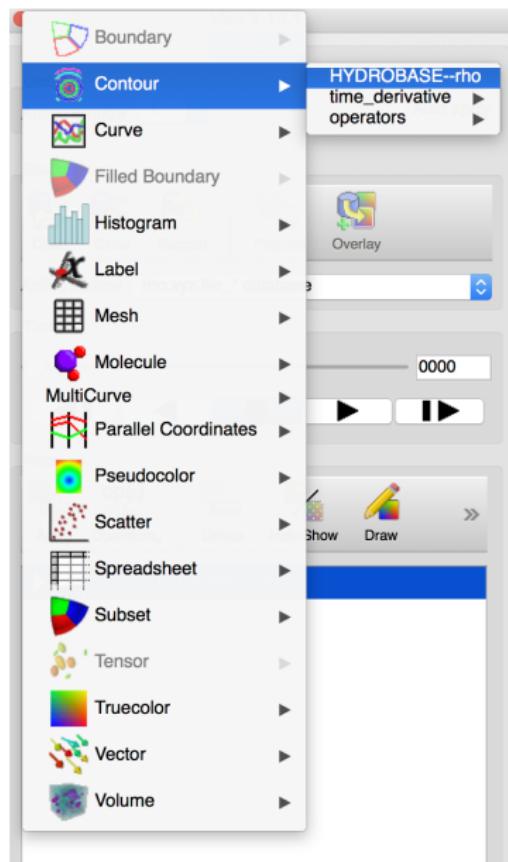
Visualizing ET Data in Visit: Opening Visit



Visualizing ET Data in Visit: Loading Data



Visualizing ET Data in Visit: The Contour Plot

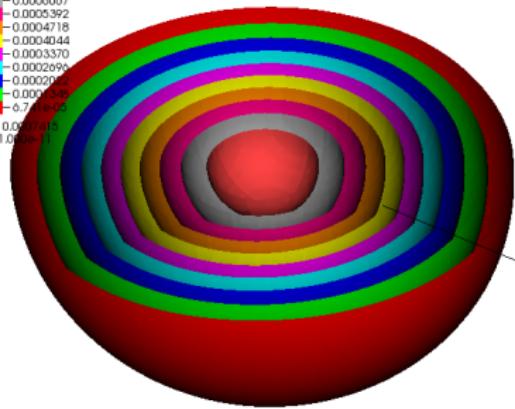


DB: rho.xyz.file_0.h5
Cycle: 0 Time:0

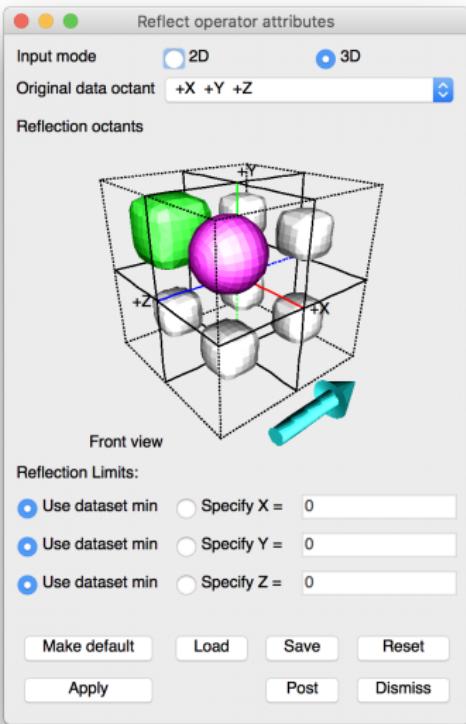
Contour Var: HYDROBASE--rho

-0.0006741
-0.0006067
-0.0005392
-0.0004718
-0.0004044
-0.0003370
-0.0002696
-0.0002022
-0.0001348
-6.74e-05

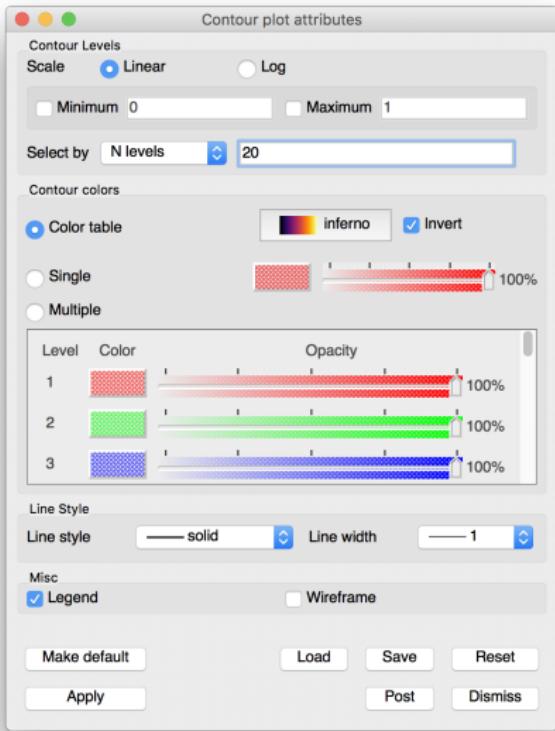
Max: 0.0007413
Min: 1.000e-01



Visualizing ET in Visit: Accounting for Symmetry

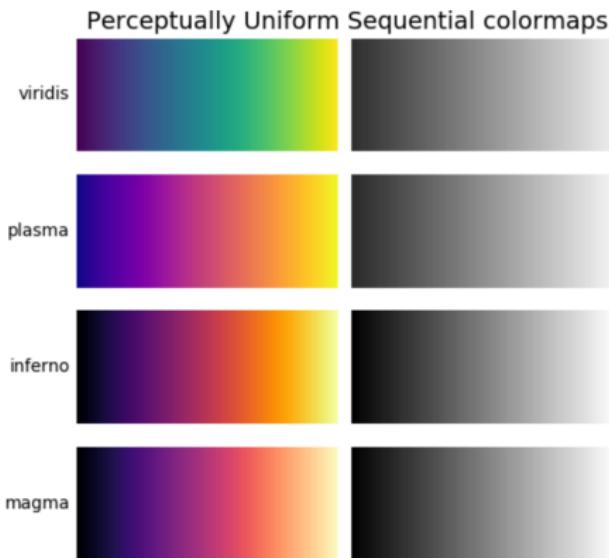


Visualizing ET in Visit: Changing the Colormap

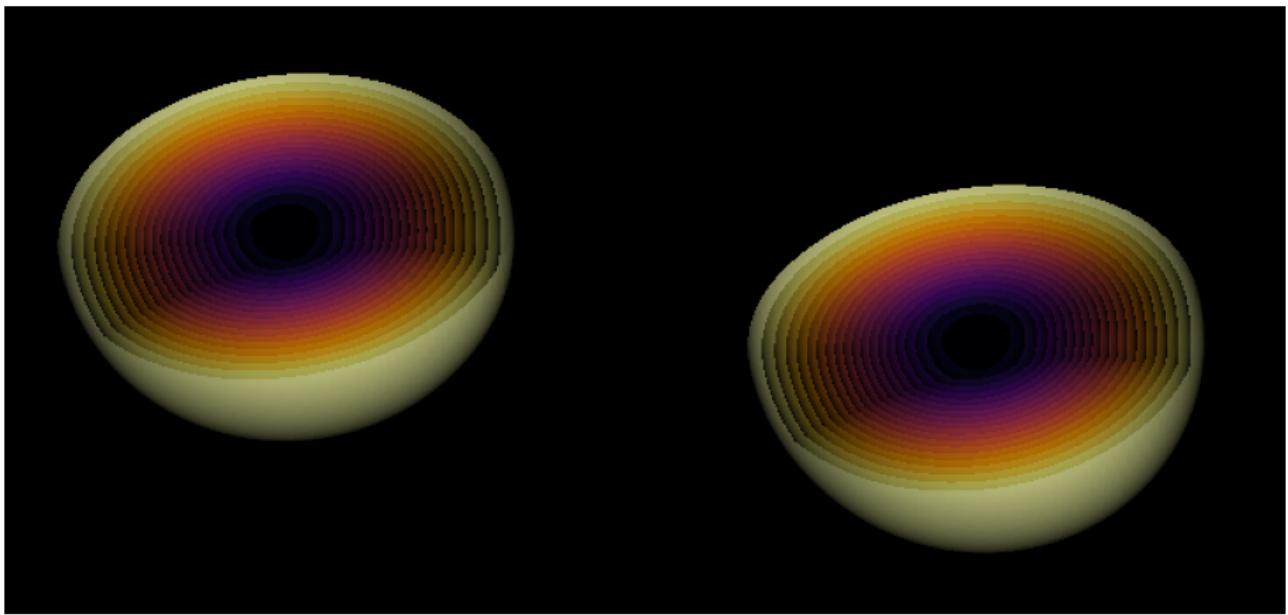


Aside: The Rainbow Colormap is Bad!

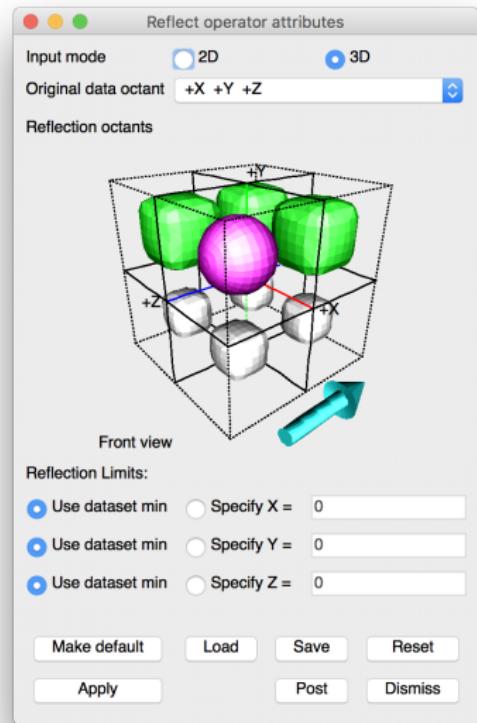
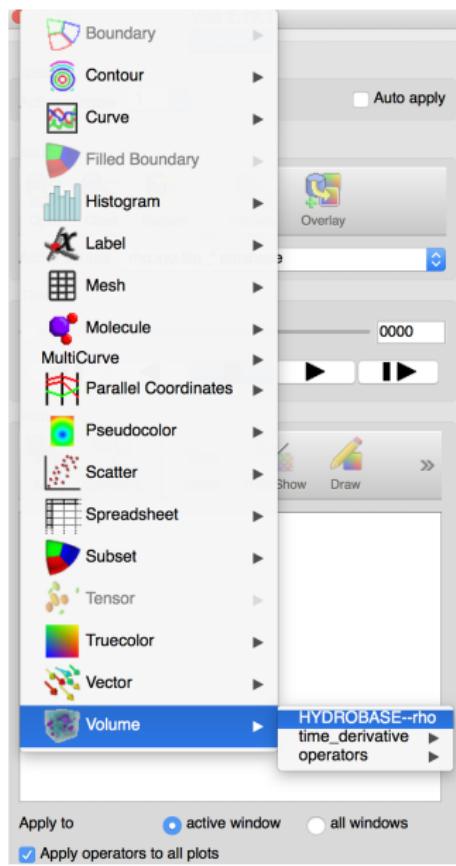
- Plots should convert to grayscale easily
- Plots should be colorblind friendly
- The human mind/eye doesn't perceive differences in color uniformly
- The canonical “rainbow” colormap is bad about all of this
- A good choice are “perceptually uniform” colormaps, which solve many of these problems
- For more info, see <https://matplotlib.org/>



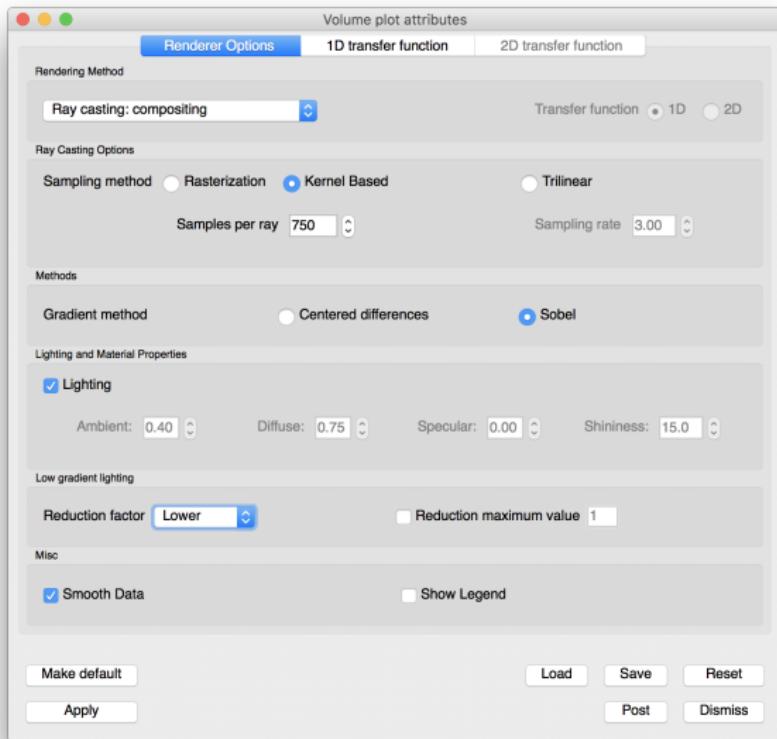
Visualizing ET in Visit: The Two Stars



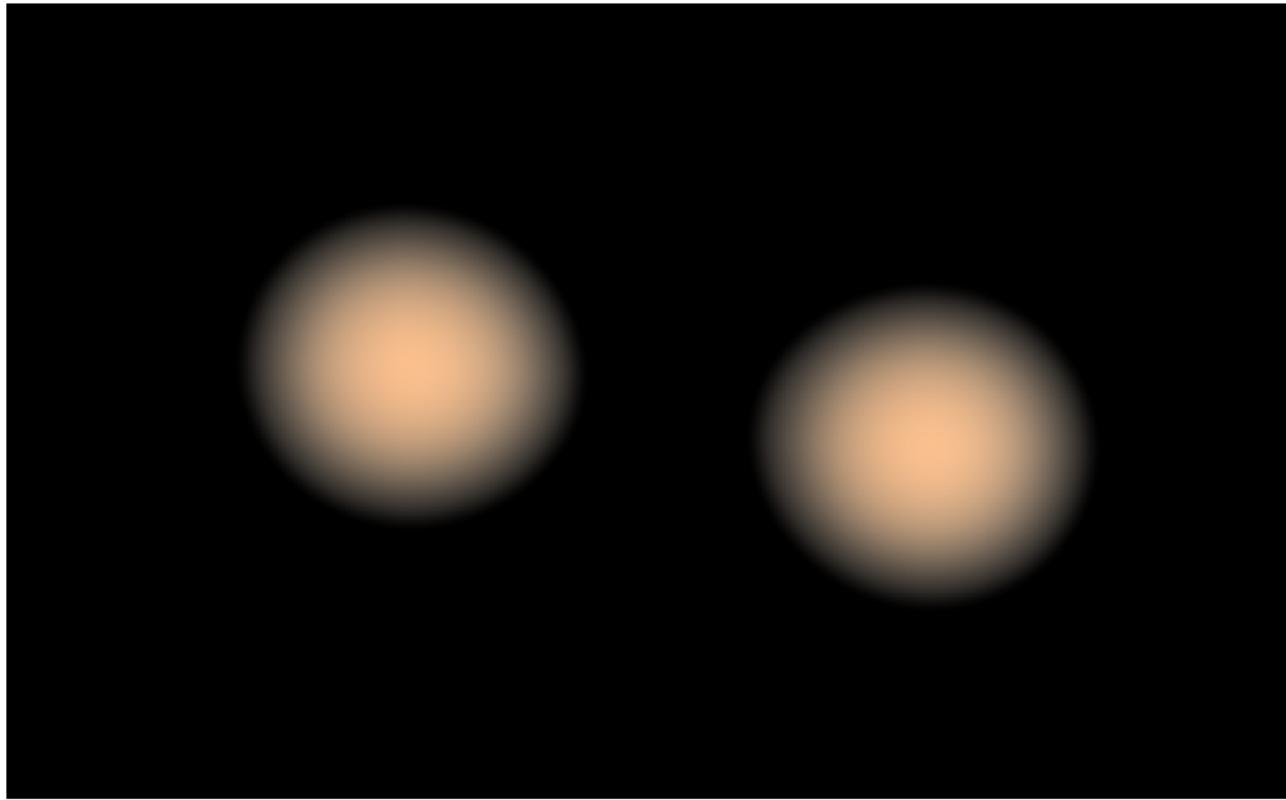
Visualizing ET in Visit: The Volume Render



Visualizing ET in Visit: Ray Tracing



Visualizing ET in Visit: Ray Tracing



Using Arbitrary Data With Visit: XDMF

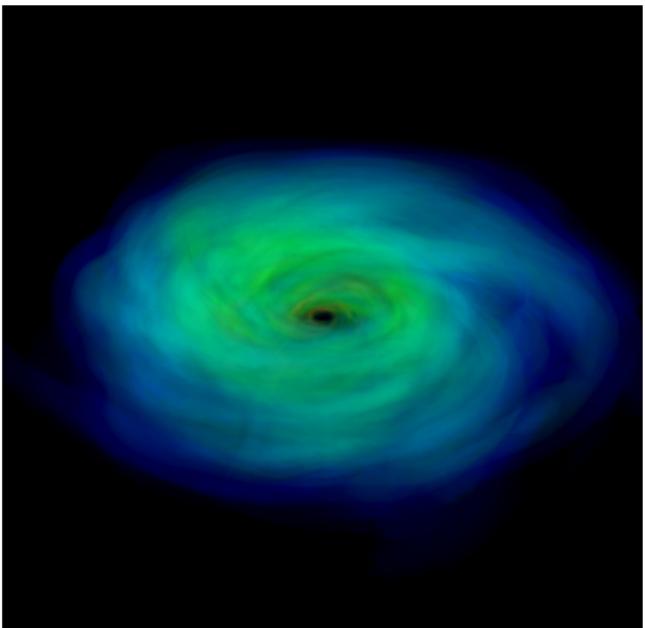
- Suppose you generated data with a code and you want to read it in to visit
- You can create a *XDMF* file to tell Visit how to read your data
- *XDMF* is XML, which is human readable/writable.
- Resources:
 - http://www.xdmf.org/index.php/XDMF_Model_and_Format
 - https://www.visitusers.org/index.php?title=Using_XDMF_to_read_HDF5

Using Arbitrary Data With Visit: XDMF

```
<Domain>
  <Grid Name="mesh1" GridType="Uniform">
    <Topology TopologyType="2DSMesh"
      NumberOfElements="21 31"/>
    <Geometry GeometryType="X_Y">
      <DataItem Dimensions="21 31" NumberType="Float"
        Format="HDF">
        xdmf2d.h5:/X
      </DataItem>
      <DataItem Dimensions="21 31" NumberType="Float"
        Format="HDF">
        xdmf2d.h5:/Y
      </DataItem>
    </Geometry>
    <Attribute Name="Pressure"
      AttributeType="Scalar" Center="Cell">
      <DataItem Dimensions="20 30" Format="HDF">
        xdmf2d.h5:/Pressure
      </DataItem>
    </Attribute>
  </Grid>
</Domain>
```

Exercise: Read in Simulation Data With XDMF

- Write an XDMF file to read the data in `exercises/harmdisk3d` and visualize it in Visit.
- Resources:
 - http://www.xdmf.org/index.php/XDMF_Model_and_Format
 - https://www.visitusers.org/index.php?title=Using_XDMF_to_read_HDF5



You Have (Hopefully) Learned

- ① (Very roughly) how does volume rendering work?
- ② How to put your simulation data into these tools
 - Your code has one representation of the physics, the data file has another, the visualization tool yet another.
 - Mapping these representations into each other can be hard.
 - We'll do it for:
 - Einstein Toolkit data
 - Arbitrary simulation data
- ③ Some tips and tricks
 - Visualization best practices
 - Idiosyncracies of each tool