Visualization of Simulation Data

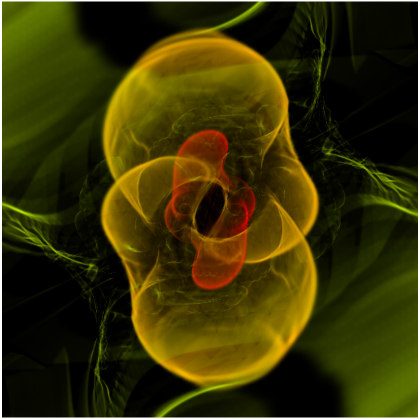
Jonah M. Miller

Los Alamos National Lab

European Einstein Toolkit Workshop
September, 2018

Why Visualize in 3D?

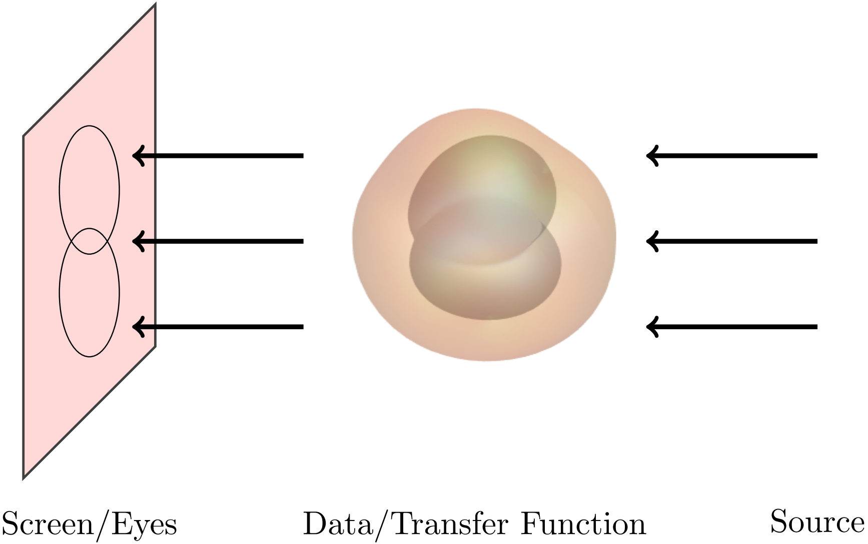
- Eye catching—great for journals, posters, talks, etc.
- 3D structures not visible in slice plots can be visible in 3D volume renders
- Useful for exploring and understanding your own data



What You Will Learn

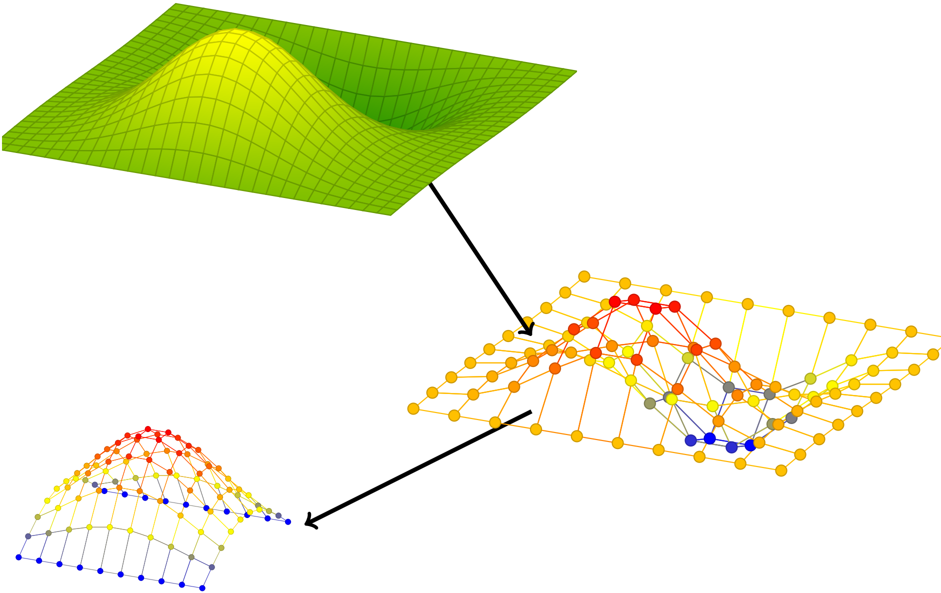
1. (Very roughly) how does volume rendering work?
2. How to put your simulation data into these tools
    - Your code has one representation of the physics, the data file has another, the visualization tool yet another.
    - Mapping these representations into each other can be hard.
    - We'll do it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

Volume Rendering and Ray Tracing

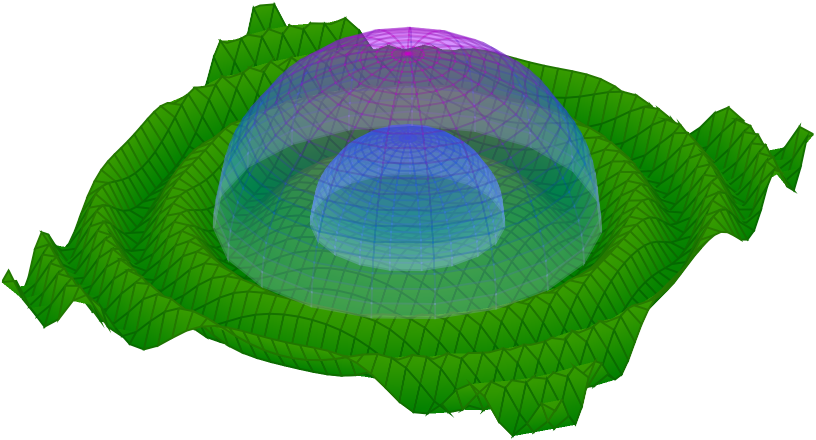


Mapping Your Simulation Data Into a Visualization Tool

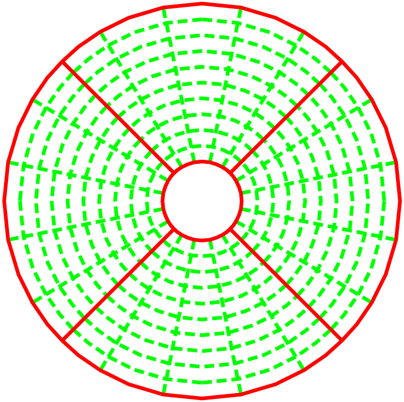
What do we care about in a simulation?

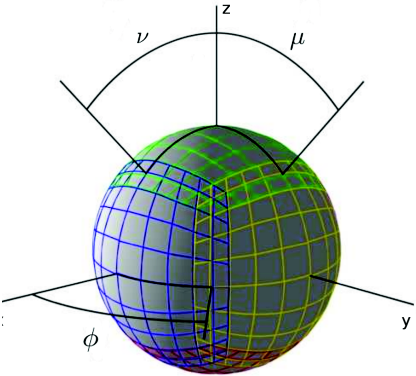


What do we care about in a simulation?



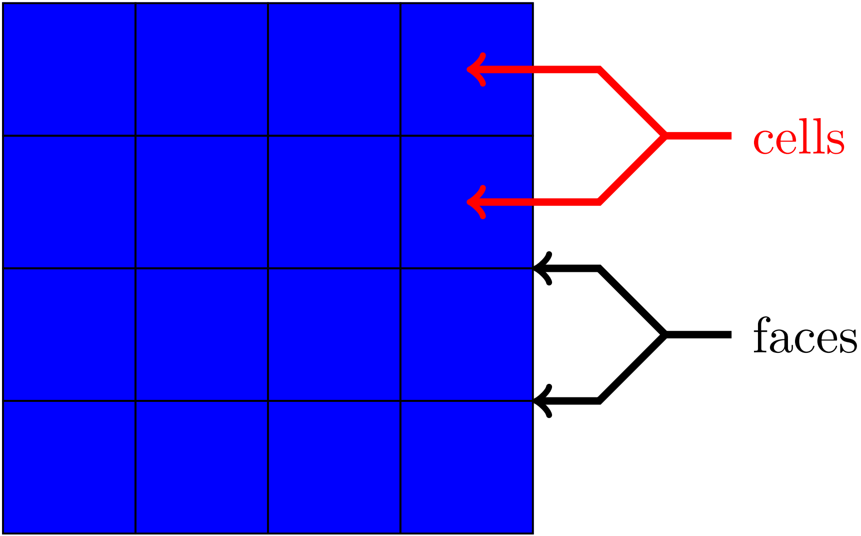
What do we care about in a simulation?



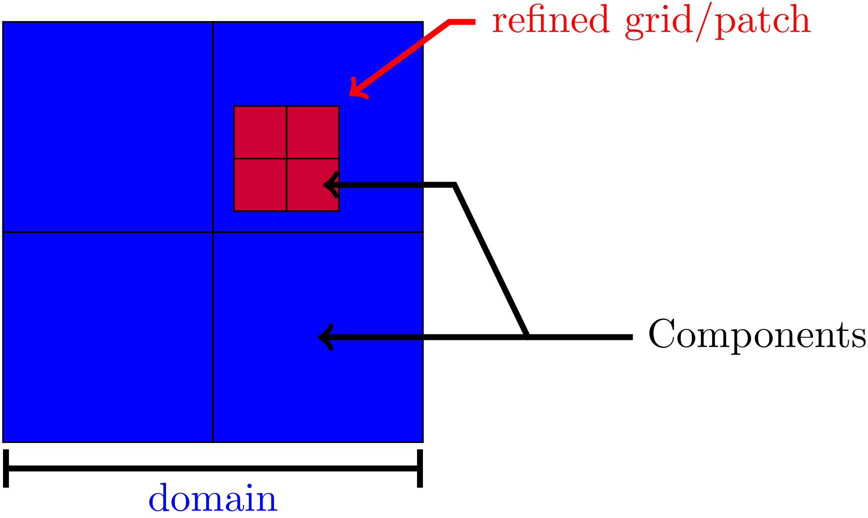


Schnetter *et al* 2006 Class. *Quantum Grav.* **23** S553
Pollney *et al* 2011 *Phys. Rev. D* **83** 044045

Important Finite Volume Concepts



Important Carpet AMR Concepts



Visualization Tools

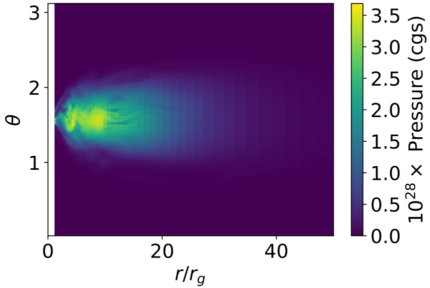
- **Pros:** Simple. Easy to explore/inspect data.
- **Cons:** Limited. 3D visualization hard.

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

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

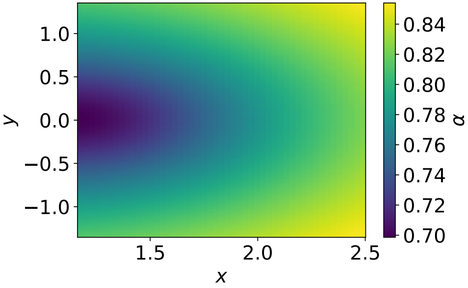
Pure Python

```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

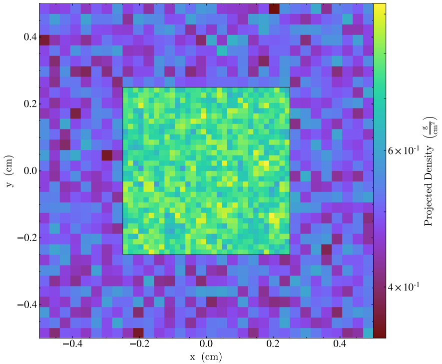
```
lapse_filenames = sorted(glob('admbase-lapse.*.h5'))
lapse = {}
for filename in lapse_filenames:
    with h5py.File(filename,'r') as f:
        for k,v in f.items():
            try:
                lapse[k] = v.value
            except:
                pass
```



yt

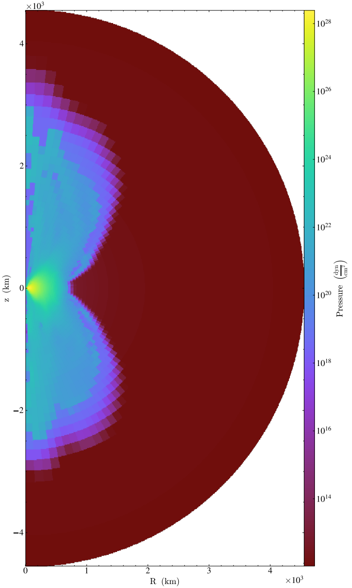
(Optional) Exercise: Installing yt

- pip install yt –user

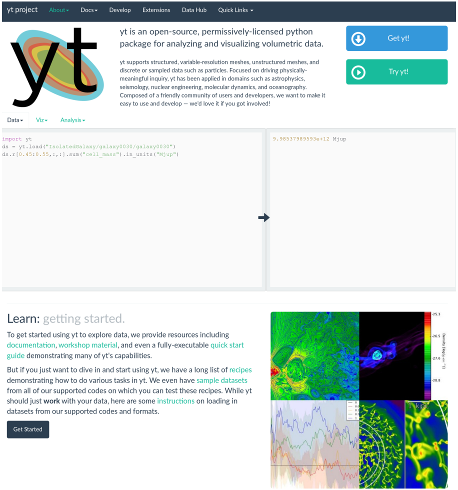


Exercise: Loading Curvilinear Data into yt

- **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



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

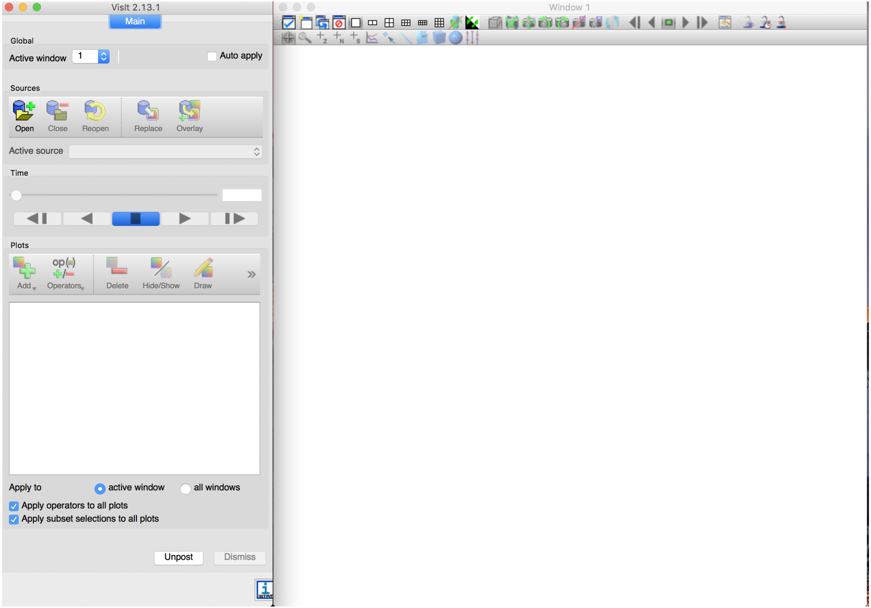


Help Wanted

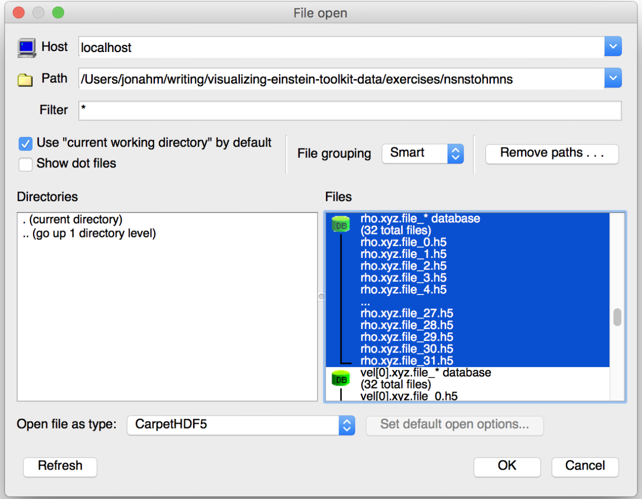
- Carpet AMR and yt are not easy to integrate
- Ideally a frontend for yt can be written to integrate it with Carpet

Visit

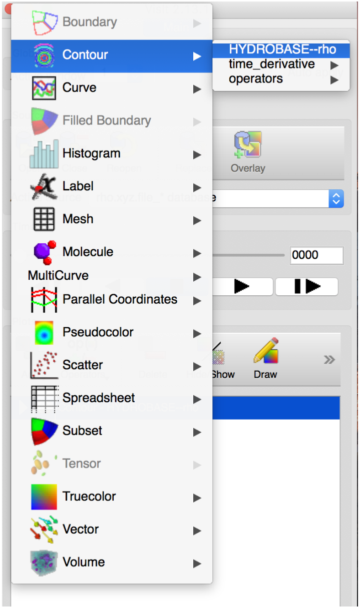
Visualizing ET Data in Visit: Opening Visit

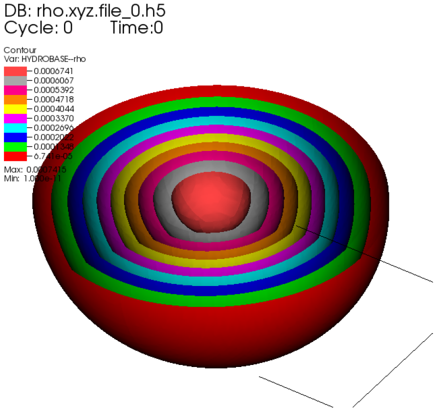


Visualizing ET Data in Visit: Loading Data

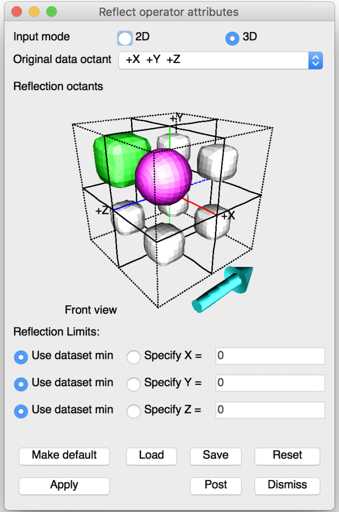


Visualizing ET Data in Visit: The Contour Plot





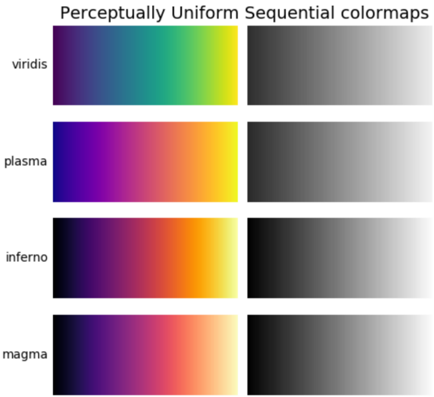
Visualizing ET in Visit: Accounting for Symmetry



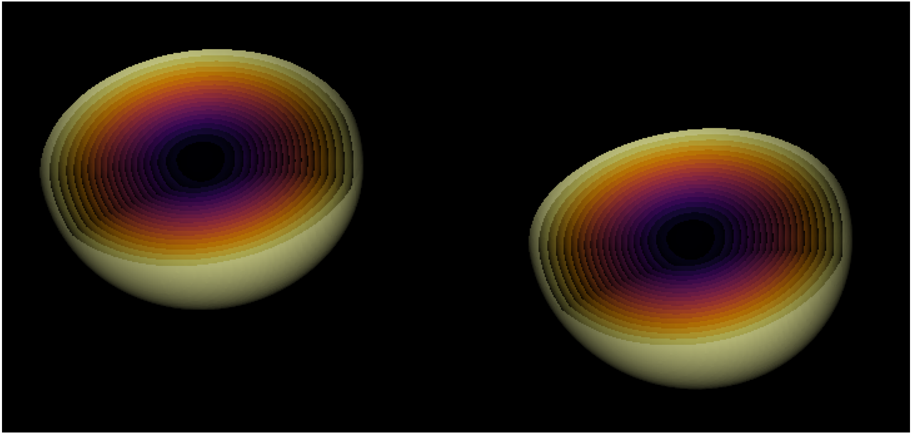
Visualizing ET in Visit: Changing the Colormap

Aside: The Rainbow Colormap is Bad!

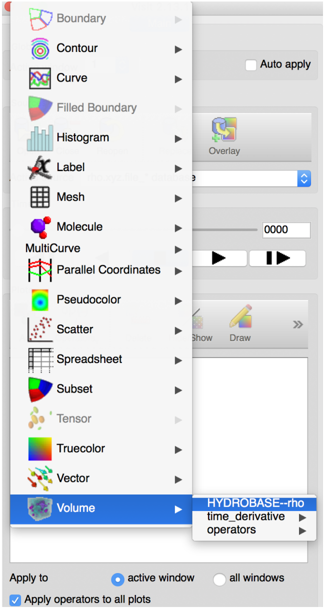
- Plots should convert to grayscale easily
- Plots should be colorblind friendly
- The human mind/eye doesn't perceive differences in color uniformly
- The canonical "rainbow" colormap is bad about all of this
- A good choice are "perceptually uniform" colormaps, which solve many of these problems
-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

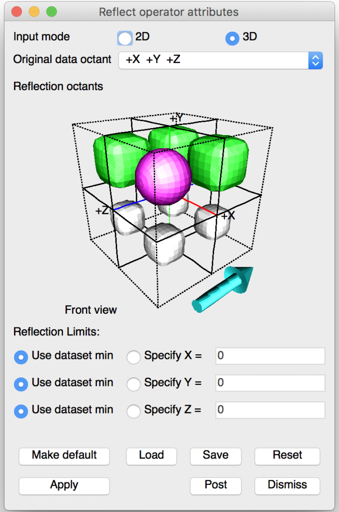


Visualizing ET in Visit: The Two Stars

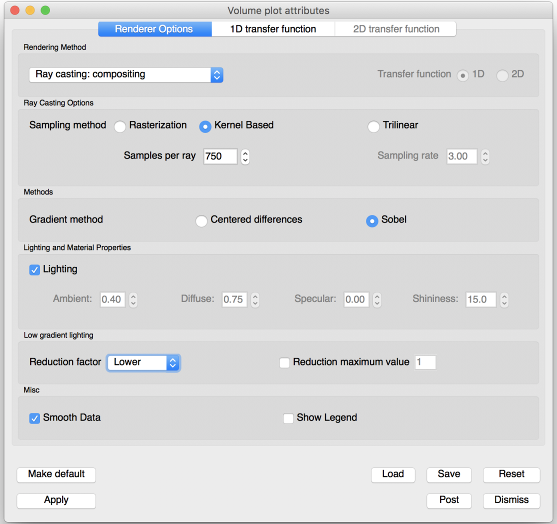


Visualizing ET in Visit: The Volume Render

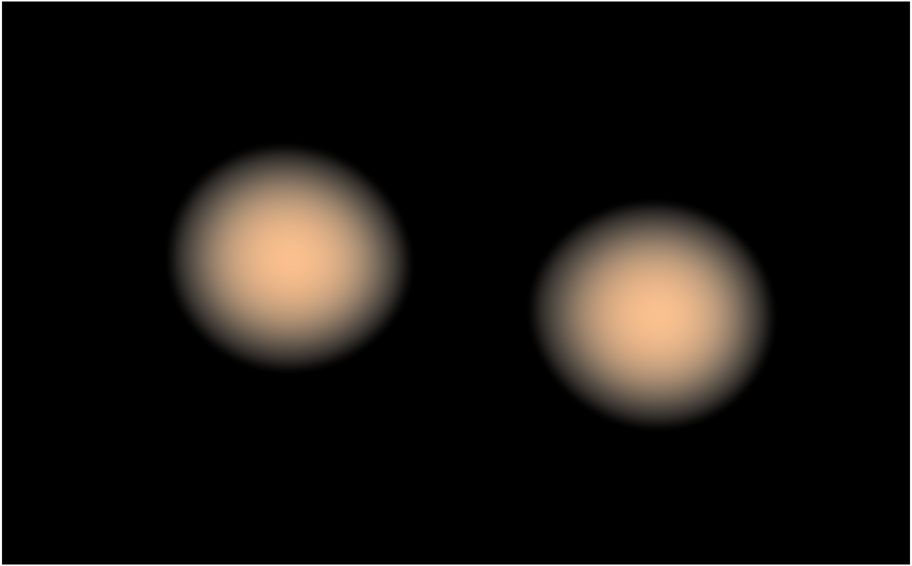




Visualizing ET in Visit: Ray Tracing



Visualizing ET in Visit: Ray Tracing



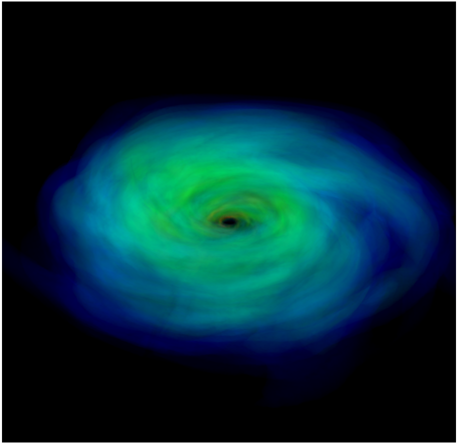
Using Arbitrary Data With Visit: XDMF

- 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`

```
<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