

SimulationIO and yt

Jonah M. Miller

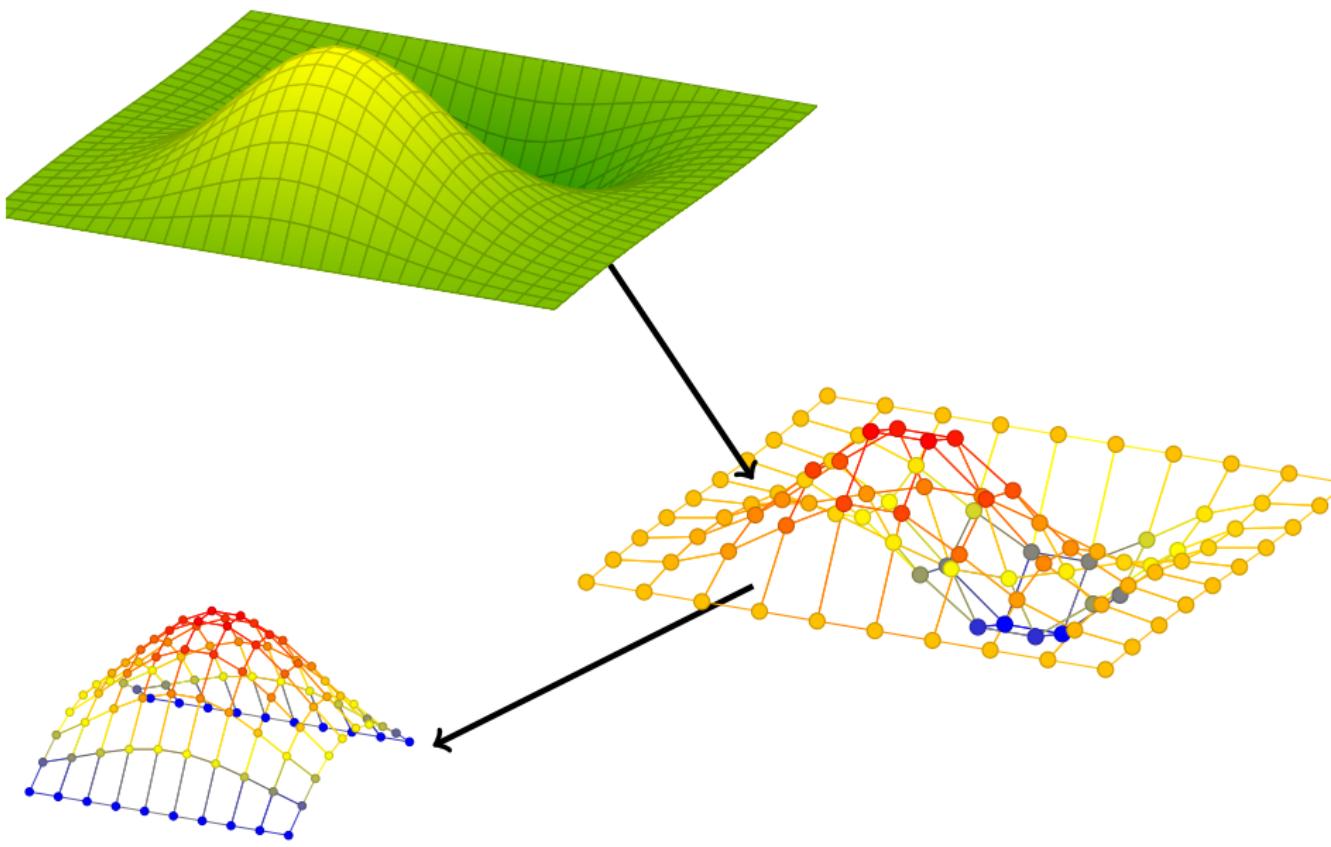
Perimeter Institute for Theoretical Physics

Einstein Toolkit Workshop
Wednesday, June 15

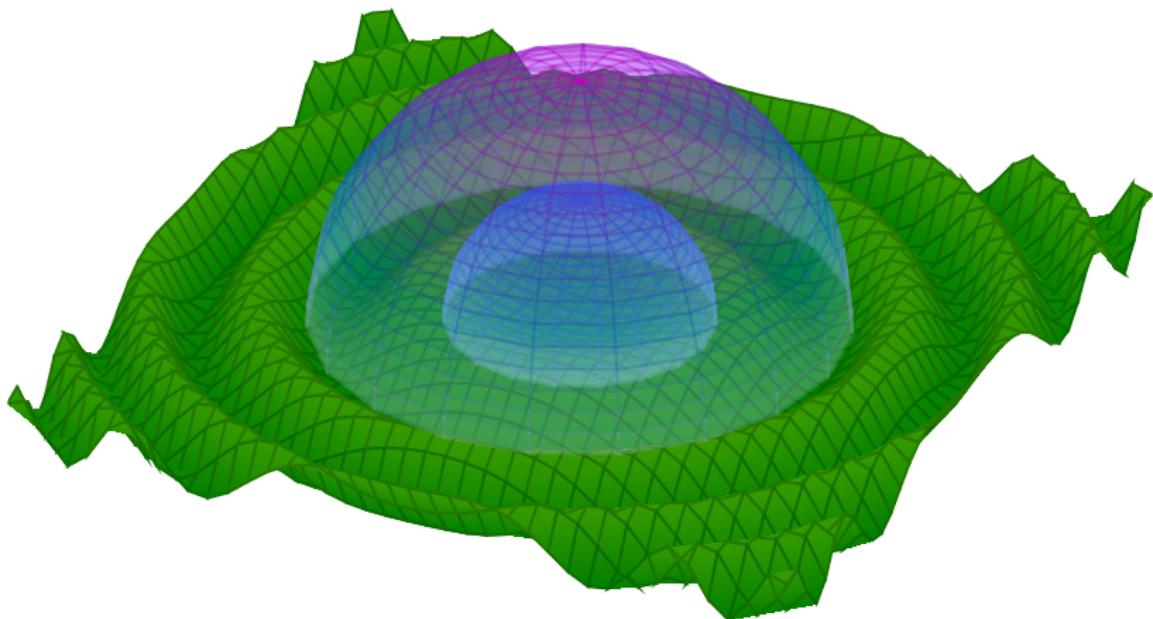
What are they?

- **SimulationIO** is a library
 - Written by Erik Schnetter
 - Open source
 - Meant to be a *high-level* description of the output of potentially many simulations
 - Abstracts away simulation details
 - Describes an API API
 - Uses category theory to cleanly remove ambiguities
- **yt** is a visualization (and data analysis) toolkit
 - Open source
 - Community driven
 - Written in Python
 - Very hackable

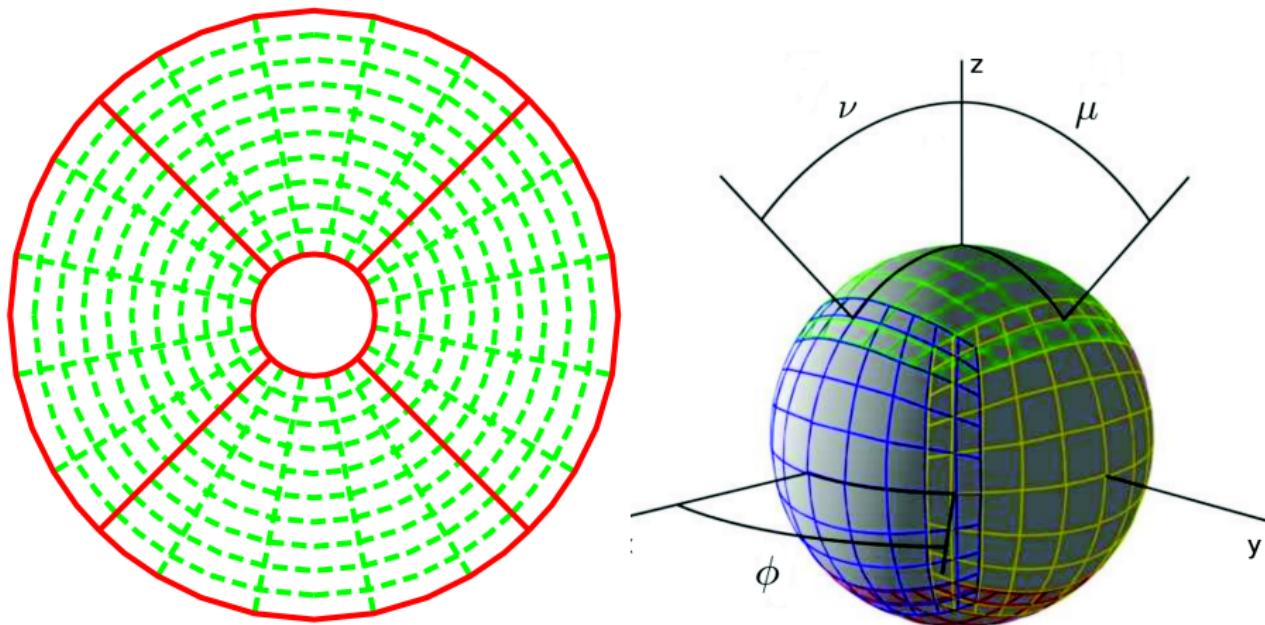
What do we care about in a simulation?



What do we care about in a simulation?



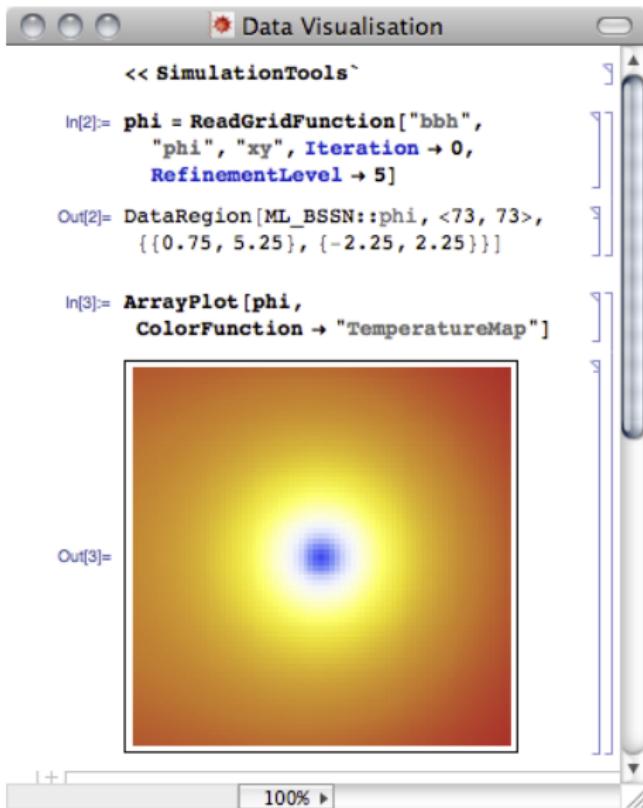
What do we care about in a simulation?



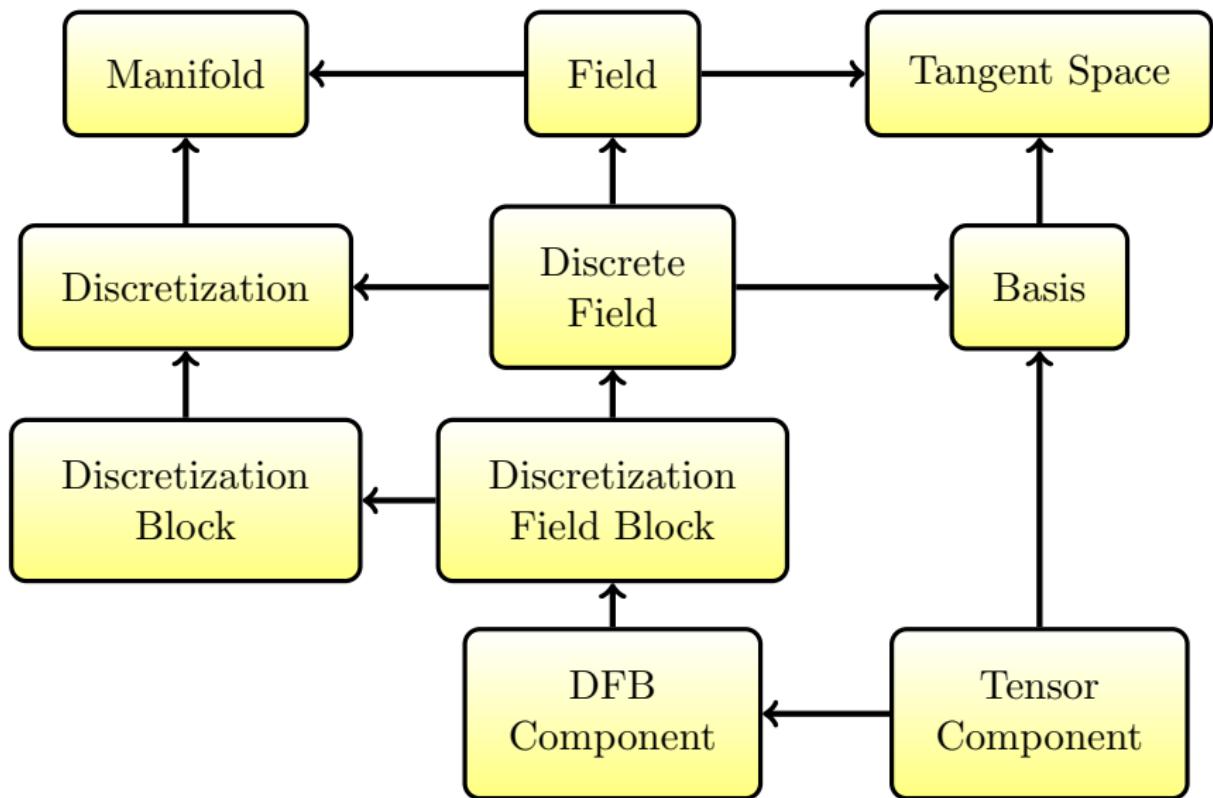
Schnetter *et al* 2006 *Class. Quantum Grav.* **23** S553
Pollney *et al* 2011 *Phys. Rev. D* **83** 044045

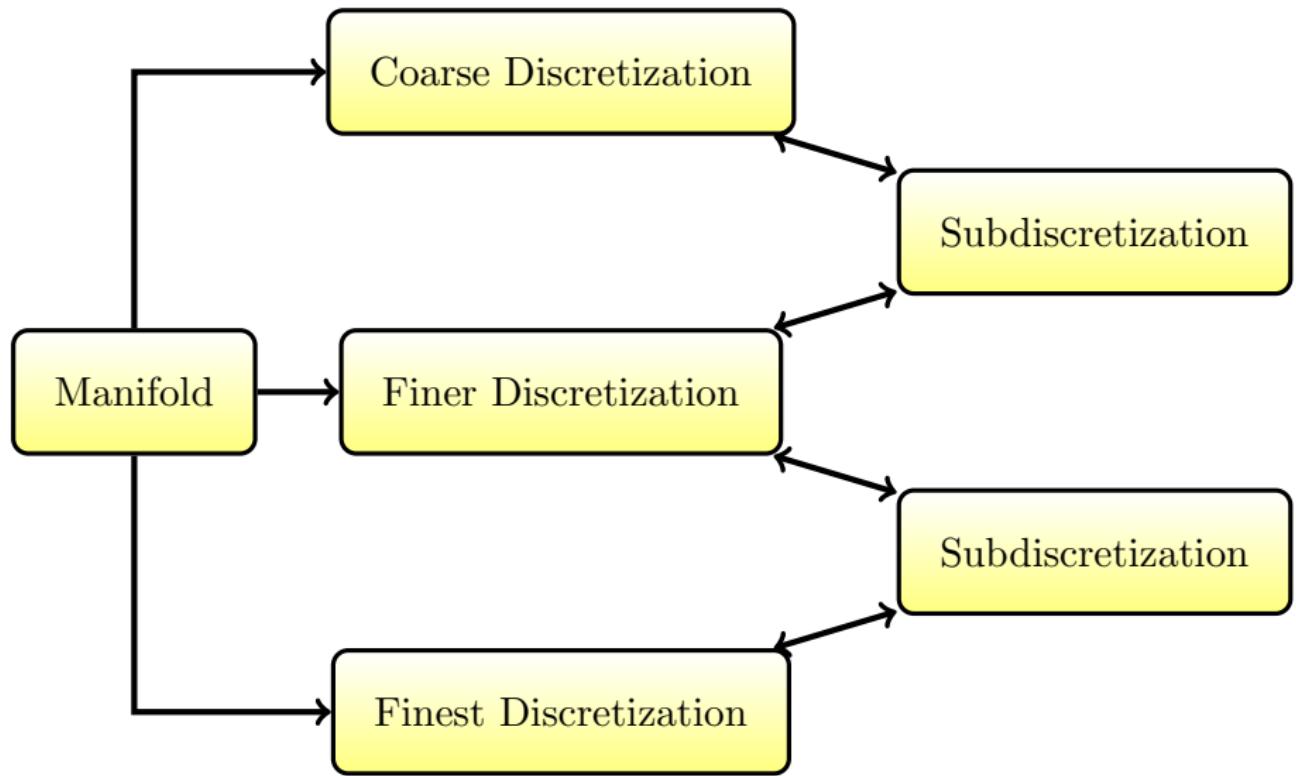
Other efforts

- Fiber Bundle by Werner Benger
- PostCactus by Wolfgang Kastaun
- Simulation Tools by Ian Hinder and Barry Wardell
- scidata by David Radice
- rugutils by Federico Guercilena
- pycactus by Roberto De Pietri, Francesco Maione, and Frank Löffler



Projects as Directed Graphs





SimulationIO Utilities

- `sio-list` is a utility for listing the contents of a file

```
sio-list file.s5
```

- `sio-convert-carpet-output` converts a sequence of hdf5 files to an s5 file.

```
sio-convert-carpet-output out.s5 file_*.h5
```

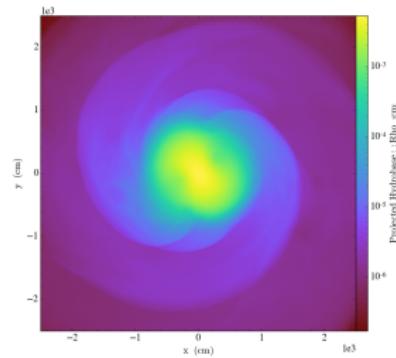
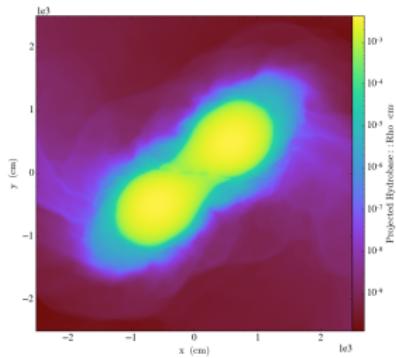
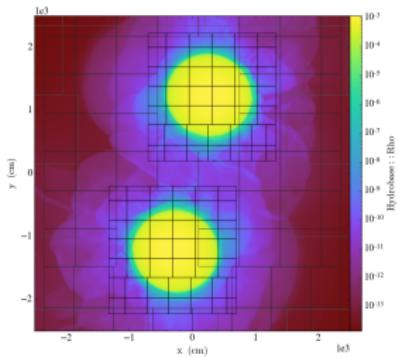
SimulationIO Python API

```
import pysimulationio
project,file_handle = SIO.readProject(filename)
field = project.fields['rho']
dcf = field.discretefields['rho']
for dcfb in \
dcf.discretefieldblocks.itervalues():
    for dcfbc in \
        dcfb.discretefieldblockcomponents:
        c = dcfbc.values()[0]
        dset = c.getData_Set()
        data = np.array(dset.read_double())
file_handle.close()
```

yt

volume rendering

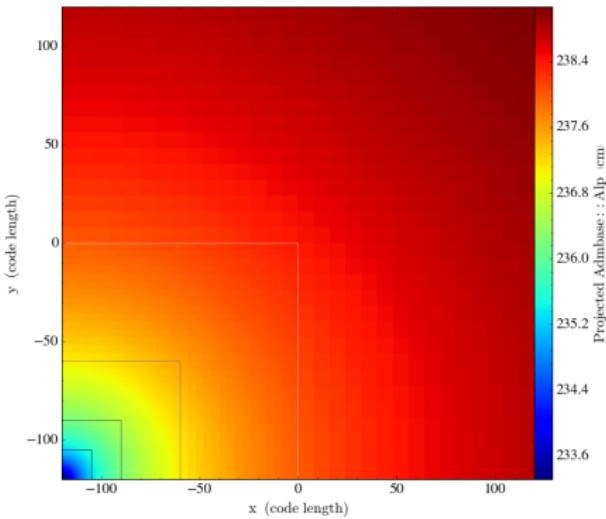
Projections and Slices



Simulation due to David Radice

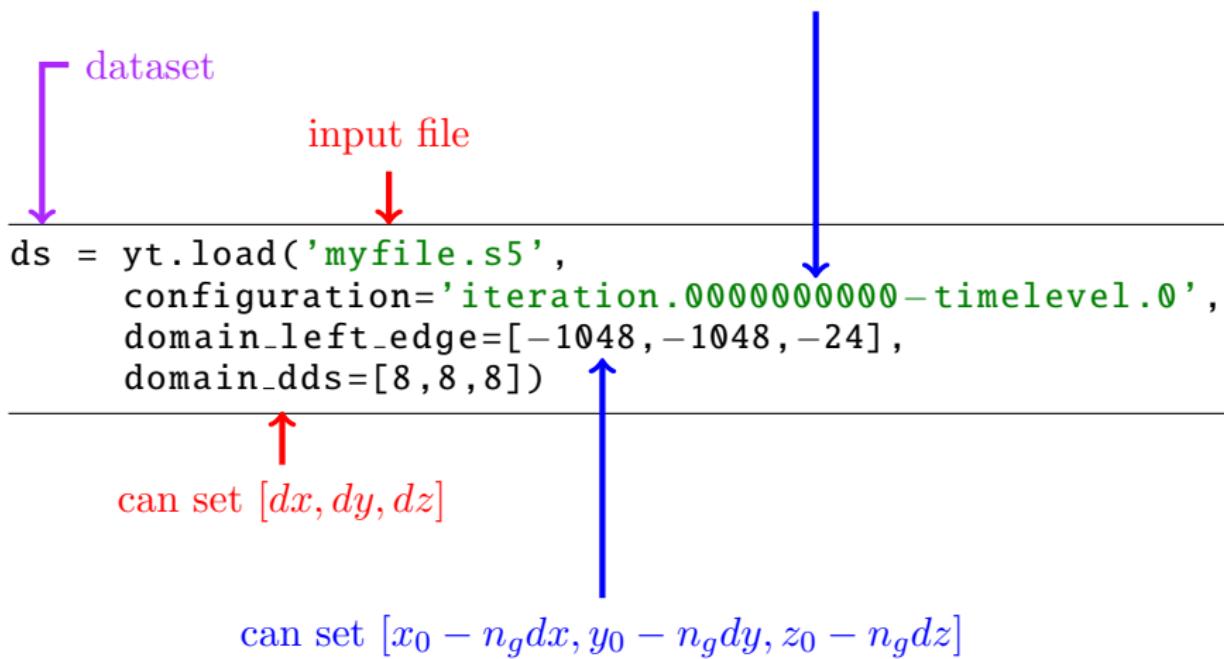
Loading SimulationIO Data

```
ds = yt.load(fpath,
    centering='vertex',
    Configuration=conf_name)
p = yt.ProjectionPlot(ds,
    'z','ADMBASE::alp',
    center=center,
    width=width)
p.set_log('ADMBASE::alp',
          False)
p.set_map(field='all',
          cmap='jet')
p.annotate_grids()
p.show()
```



Anatomy of yt.load

configuration is combo of iteration, timelevel



Parallelism is Easy

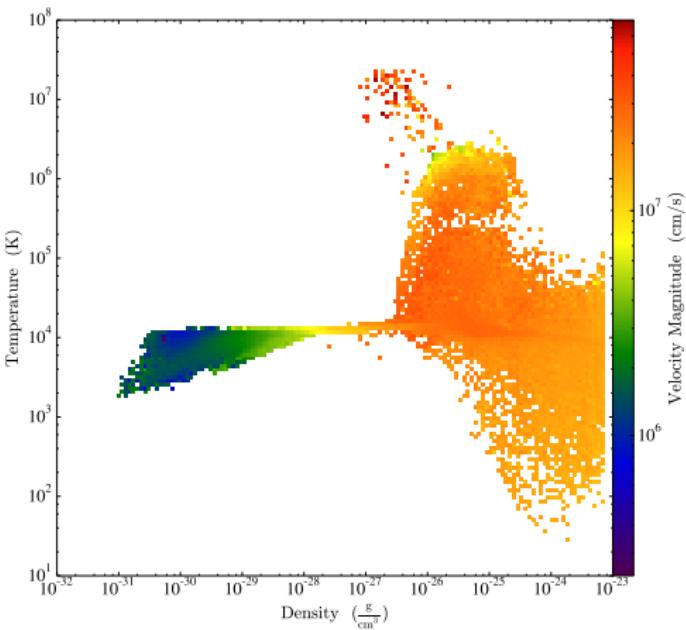
often all you need!

```
import yt
yt.enable_parallelism()
ds = yt.load(fpath, configuration=conf_name)
prj = yt.OffAxisProjectionPlot(ds, orthog_vec,
                                center, width=width)
if yt.is_root():
    prj.save(plot_name)
```

selects root process

Data Analysis

```
import yt
source = "./galaxy0030"
d = "density"
t = "temperature"
vm = "velocity_magnitude"
ds = yt.load(source)
ad = ds.all_data()
yt.PhasePlot(ad,d,t,vm)
```



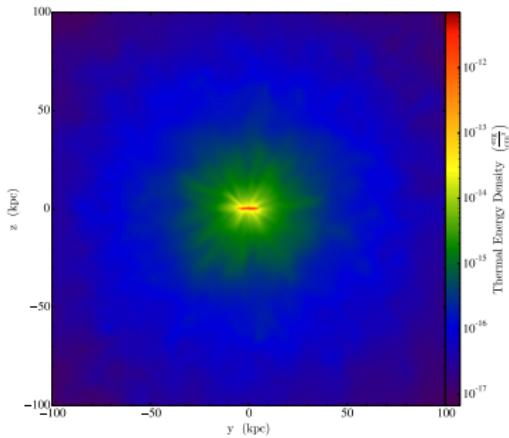
<http://yt-project.org/data/>

With Python, yt is Easily Extendable

```
# function defining
# new quantity
fname = "thermal_energy_density"
def therm_en_dens(field,data):
    n = data['gas',
              'number_density']
    kT = data['gas', 'kT']
    return (3/2)*n*kT

# add it to the dataset
ds.add_field(("gas", fname),
             units="erg/cm**3",
             function=therm_en_dens)

# plot
ad = ds.all_data()
yt.ProjectionPlot(...)
```



<http://yt-project.org/data/>

Extensions are Encouraged!

Author	Commit	Message
 Nathan Goldb... b3c23ff M	b3c23ff M	Merged in brittonsmith/yt (pull request #2136) Adding Gizmo
 Britton Smith 436b6de	436b6de	Removing explicit test failure.
 Britton Smith bbf532b	bbf532b	Fixing table.
 Britton Smith 52544f4	52544f4	Changing tests a bit.
 Britton Smith 09a8105	09a8105	Wrong name.
 Britton Smith e092b58	e092b58	Change one more FIRE to Gizmo.
 Britton Smith ee3d232	ee3d232	Returning metallicity to field list.
 Britton Smith 5fa56c0	5fa56c0	Adding deposited versions of metal fields and gas aliases.
 Britton Smith 9eb20a2	9eb20a2	Fixing color on the bikeshed.
 Cameron Hum... e936e8f	e936e8f	Adding answer test entry for gizmo
 Cameron Hum... a33c311 M	a33c311 M	Merging with tip
 Cameron Hum... e7f0026	e7f0026	Updating docs removing fire entry.
 Cameron Hum... 3ce6fdc	3ce6fdc	Moving FIRE frontend to Gizmo frontend
 Cameron Hum... 63ec9dd	63ec9dd	Adding tests for FIRE frontend.
 Cameron Hum... 681b490	681b490	Adding FIRE and Gizmo to supported codes.
 Cameron Hum... a8fcda2	a8fcda2	Adding docs for loading Gizmo and FIRE data
 Britton Smith cd8afa0	cd8afa0	Removing old FIRE fields.
 Britton Smith 6b45307	6b45307	Updating headers.

How To Read The Source Code

If you just want to *look* at the source code, you may already have it on your computer. If you build yt using the install script, the source is available at `$YT_DEST/src/yt-hg`. See [Installing yt Using pip or from Source](#) for more details about to obtain the yt source code if you did not build yt using the install script.

The root directory of the yt mercurial repository contains a number of subdirectories with different components of the code. Most of the yt source code is contained in the `yt` subdirectory. This directory itself contains the following subdirectories:

frontends

This is where interfaces to codes are created. Within each subdirectory of `yt/frontends/` there must exist the following files, even if empty:

- `data_structures.py`, where subclasses of `AMRGridPatch`, `Dataset` and `AMRHierarchy` are defined.
- `io.py`, where a subclass of `IOHandler` is defined.
- `fields.py`, where fields we expect to find in datasets are defined
- `misc.py`, where any miscellaneous functions or classes are defined.
- `definitions.py`, where any definitions specific to the frontend are defined. (i.e., header formats, etc.)

fields

This is where all of the derived fields that ship with yt are defined.

geometry

This is where geometric helper routines are defined. Handlers for grid and oct data, as well as

Advantages

- SimulationIO abstracts away simulation details and deals with physics ideas like manifolds and submanifolds
- Both frameworks have a Python API:
 - Scripting interface
 - Easily extendable
- yt is community developed
 - Inclusive and Accessible
- yt parallelizes reliably and seems to scale well

Disadvantages

- No GUI to speak of
- Cactus does not (yet) output SIO files. Must convert.
- SimulationIO and corresponding reader very much in beta
- Missing features:
 - Multiblock
 - Correct Units (coming soon!)
 - Reliable vertex centering

Visualization Tasks

- Gravitational waves from a BBH in-spiral
 - Visualize the ψ_4 at a fixed time and look for the quadrupole moment of the waves in 3D.
 - Visualize the total radiated power at a fixed time.
 - Plot lapse of the black holes at a fixed time.
 - Plot the instantaneous motion of the black holes by overlaying the shift vector
- A single neutron star
 - Experiment with volume plotting the full data
 - Experiment with the vertex-centred symmetric data. What goes wrong when you use cell-centring? What about volume rendering? Plot the grid structure.
- A binary neutron stars
 - Plot the grid structure of the neutron stars at a fixed time
 - Experiment with projections vs. slicing vs. volume plotting. What goes wrong when you make a volume plot?