
yt: A Deep Dive
November 6, 2011

yt-project.org

Quick outline

- Codes `yt` can handle
- Command-line tools
- Basic data-handling
- Basic plots
- Data objects
- Advanced plots
- Fun

Supported Codes!

- Enzo
- Nyx / Castro / Orion / Boxlib
- FLASH
- Athena (sort of!)
- RAMSES (partial)
- ART (sort of!)

What *can* `yt` support?

Starting Up `yt`

- Command-line
- Interactive
- Scripting

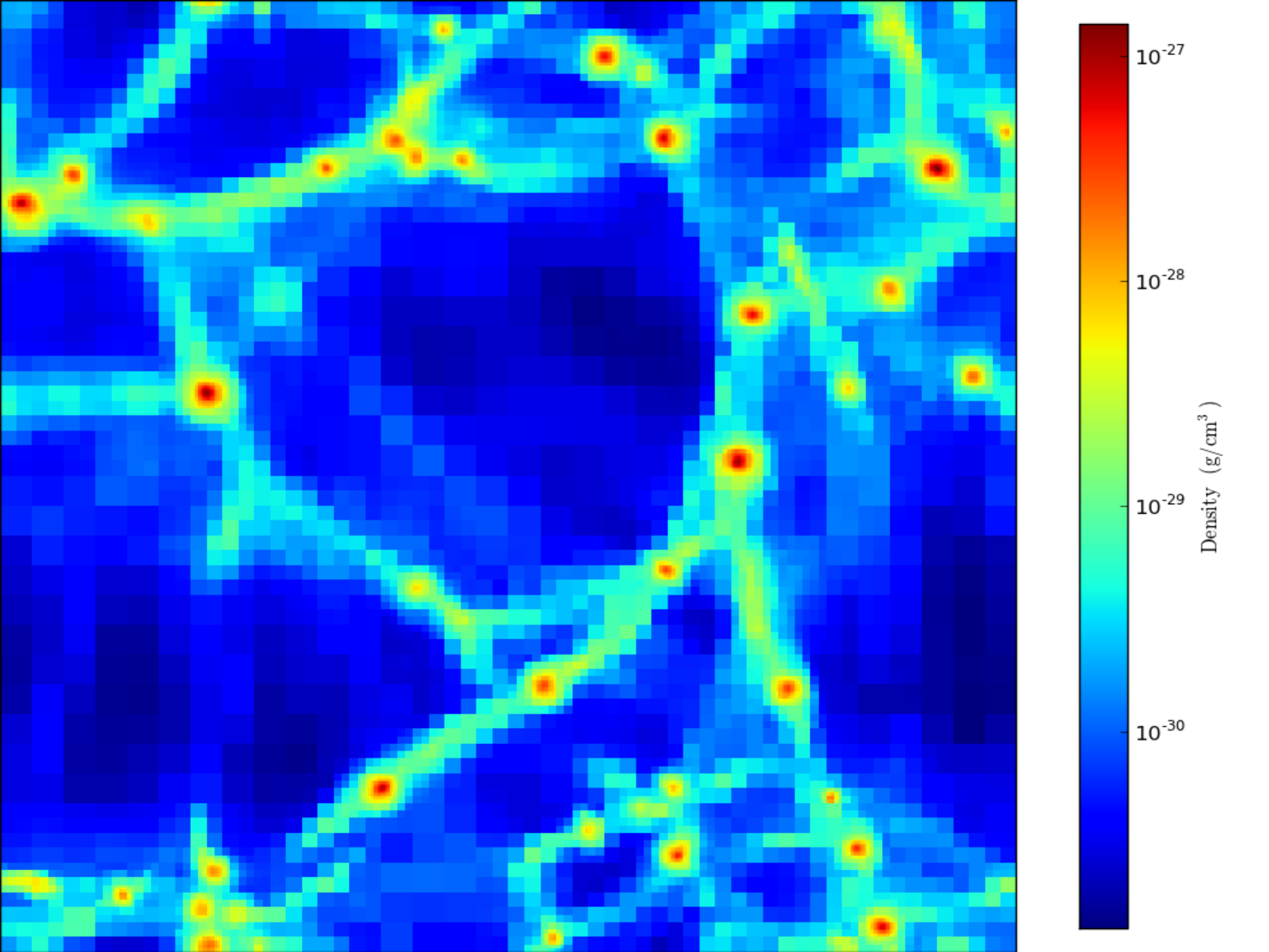
Simple, command-line toos

yt stats DD0023/DD0023

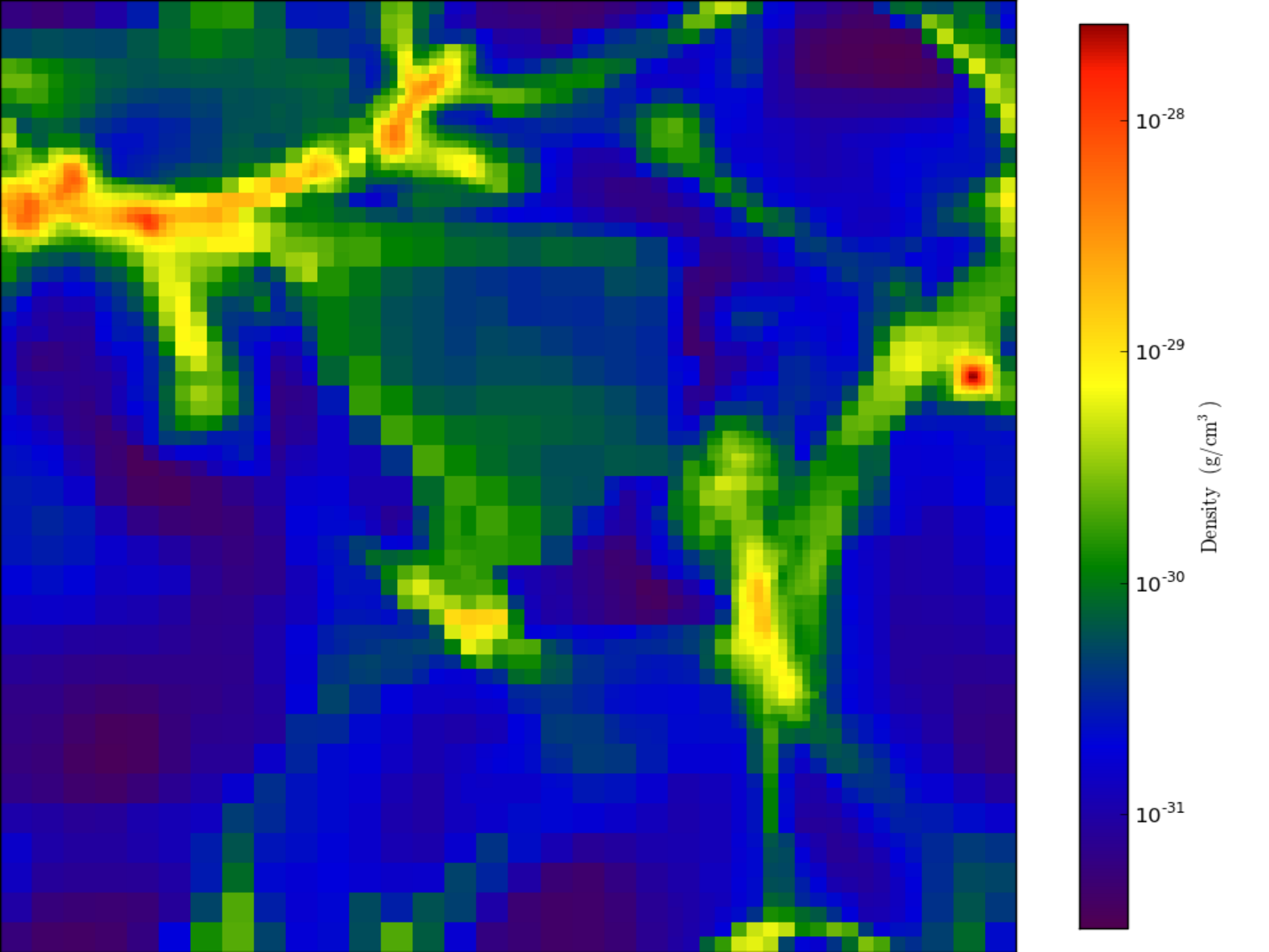
```
level # grids     # cells
---------------------------
  0      1       32768
  1     46       77712
  2     46       38288
  3     40       16976
  4     20        7480
---------------------------
       153      173224


t = 2.29708470e+02 = 4.36096127e+17 s = 1.38285175e+10 years

Smallest Cell:
Width: 1.953e-03 1
Width: 1.953e-03 unitary
Width: 6.250e-02 mpch
Width: 6.250e-02 mpchcm
Width: 8.878e-02 mpc
Width: 8.878e-02 mpccm
Width: 9.961e-02 aye
Width: 6.250e+01 kpch
Width: 6.250e+01 kpchcm
Width: 8.878e+01 kpc
Width: 8.878e+01 kpccm
Width: 6.250e+04 pch
Width: 6.250e+04 pchcm
Width: 8.878e+04 pc
Width: 8.878e+04 pccm
Width: 1.289e+10 auh
Width: 1.289e+10 auhcm
Width: 1.831e+10 au
Width: 1.831e+10 aucm
Width: 2.773e+12 rsunh
```

```
Usage:
    yt COMMAND [ARGS...]
    yt help [COMMAND]

Options:
    -h, --help  show this help message and exit

Commands:
    bootstrap_dev     Bootstrap a yt development environment
    bugreport         Report a bug in yt
    halos             Run HaloProfiler on one dataset
    help (?, h)       give detailed help on a specific sub-command
    hop               Run HOP on one or more datasets
    hubsubmit         Submit a mercurial repository to the yt Hub (http://h...
    instinfo          Get some information about the yt installation
    load              Load a single dataset into an IPython instance
    mapserver         Serve a plot in a GMaps-style interface
    pastebin          Post a script to an anonymous pastebin
    pastebin_grab     Print an online pastebin to STDOUT for local use. Pas...
    pasteboard        Place a file into your pasteboard.
    pasteboard_grab   Download from your or another user's pasteboard
    plot              Create a set of images
    render            Create a simple volume rendering
    rpdb              Connect to a currently running (on localhost) rpd ses...
    serve             Run the Web GUI Reason
    stats             Print stats and maximum density for one or more datasets
    update            Update the yt installation to the most recent version
    upload_image      Upload an image to imgur.com.  Must be PNG.
```

```
yt plot -p -g Density -f Density -a 0 DD0023/DD0023
```


10^{-27}
10^{-28}
10^{-29}
10^{-30}
Density (g/cm^3)

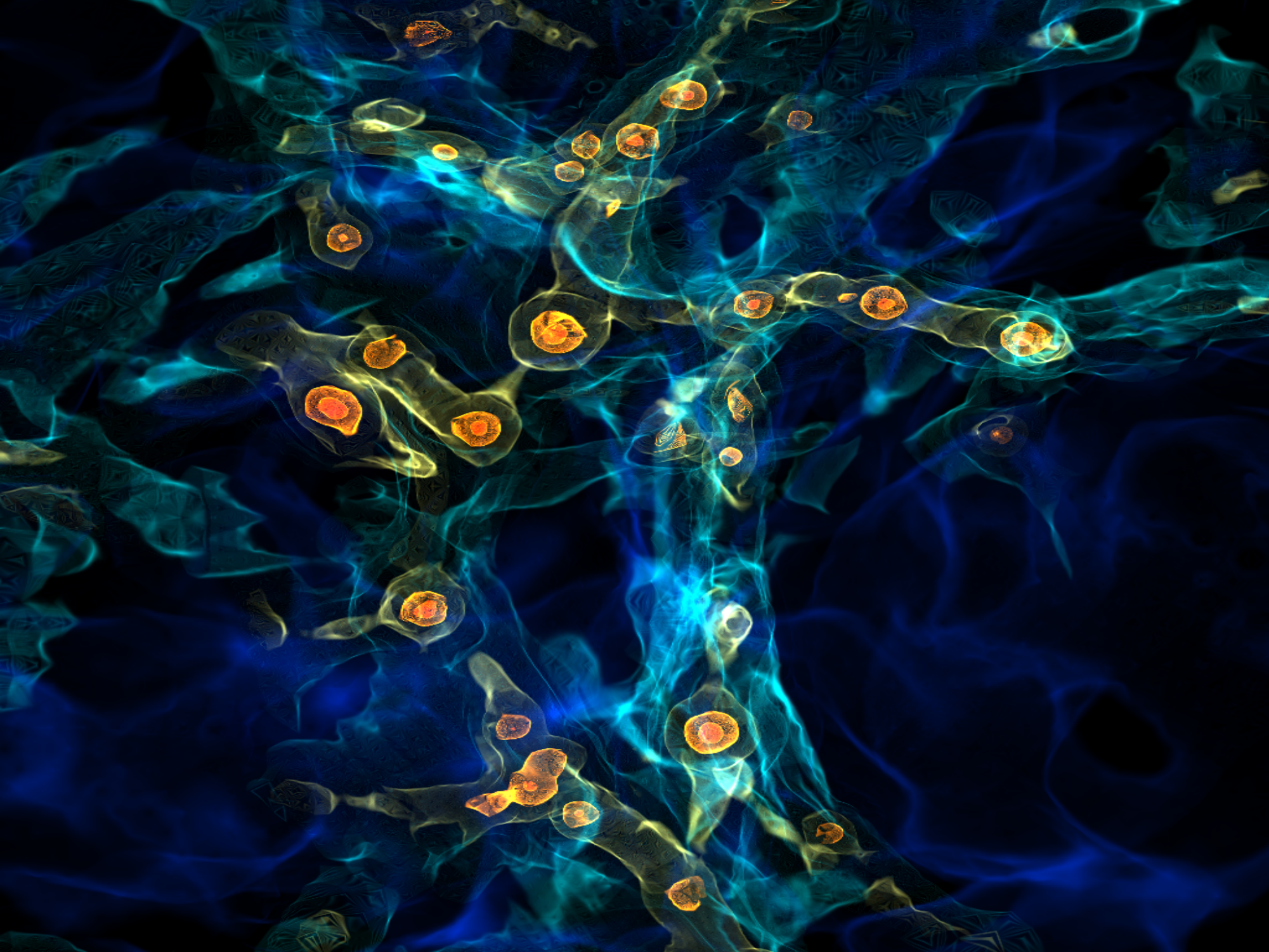
```
yt plot --colormap algae -f Density -a 0 DD0023/DD0023
```


10^{-28}
10^{-29}
10^{-30}
10^{-31}
Density (g/cm^3)

yt plot --help

```
Create a set of images

Usage:
    yt plot [ARGS...]

Options:
    -h, --help          show this help message and exit
    -w WIDTH, --width=WIDTH
                        Width in specified units
    -u UNIT, --unit=UNIT
                        Desired units
    -b BASENAME, --basename=BASENAME
                        Basename of parameter files
    -p, --projection    Use a projection rather than a slice
    -c CENTER, --center=CENTER
                        Center, space separated (-1 -1 -1 for max)
    -z ZLIM, --zlim=ZLIM
                        Color limits (min, max)
    -a AXIS, --axis=AXIS
                        Axis (4 for all three)
    -f FIELD, --field=FIELD
                        Field to color by
    -g WEIGHT, --weight=WEIGHT
                        Field to weight projections with
    -s SKIP, --skip=SKIP
                        Skip factor for outputs
    --colormap=CMAP     Colormap name
    -o OUTPUT, --output=OUTPUT
                        Folder in which to place output images
    --show-grids        Show the grid boundaries
    --time              Print time in years on image
```

```
yt render --enhance --pixels=1024 DD0023
```



yt render --help

```
Create a simple volume rendering

Usage:
    yt render [ARGS...]

Options:
    -h, --help          show this help message and exit
    -w WIDTH, --width=WIDTH
                        Width in specified units
    -u UNIT, --unit=UNIT
                        Desired units
    -c CENTER, --center=CENTER
                        Center, space separated (-1 -1 -1 for max)
    --enhance           Enhance!
    -o OUTPUT, --output=OUTPUT
                        File in which to place output
    -f FIELD, --field=FIELD
                        Field to color by
    --colormap=CMAP     Colormap name
    --contours=CONTOURS
                        Number of Contours for Rendering
    --viewpoint=VIEWPOINT
                        Viewpoint, space separated
    --pixels=PIXELS     Number of Pixels for Rendering
    --up=UP             Up, space separated
    -r VALRANGE, --range=VALRANGE
                        Range, space separated
    -l, --log           Take the log of the field?
    --contour_width=CONTOUR_WIDTH
                        Width of gaussians used for rendering.
```

Basic data-handling

How to get "into" `yt`

- Run a script with `py-yt` or `python`
- Load a dataset with `yt load`
- Run `iyt`

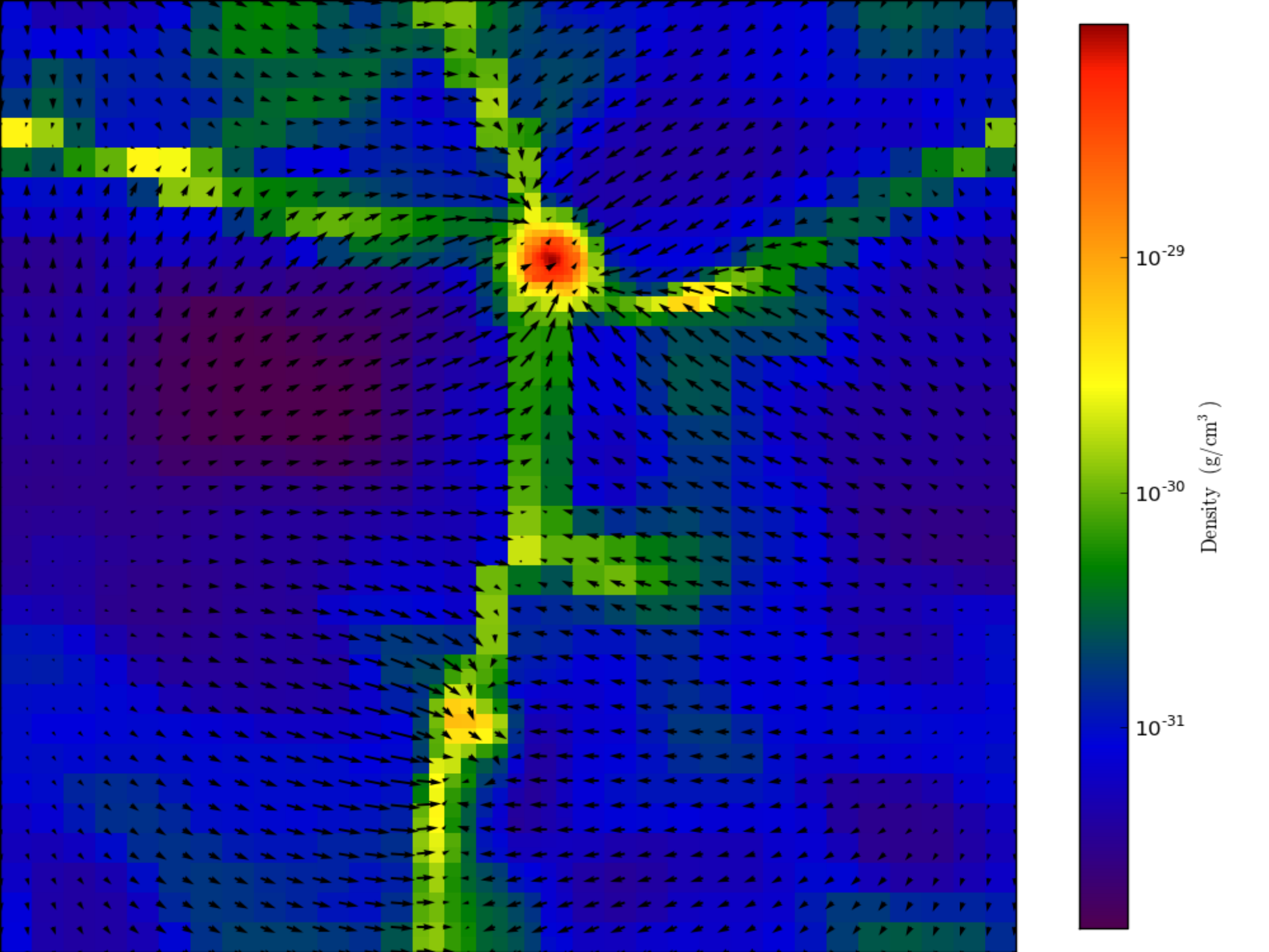
For these examples, we'll edit a script, then run it with `py-yt`

```
from yt.mods import *

pf = load("DD0023/DD0023")
pf.h.print_stats()
```

```
level # grids     # cells
------------------------------
  0      1       32768
  1     46       77712
  2     46       38288
  3     40       16976
  4     20        7480
------------------------------
       153      173224


t = 2.29708470e+02 = 4.36096127e+17 s = 1.38285175e+10 years

Smallest Cell:
Width: 1.953e-03 1
Width: 1.953e-03 unitary
Width: 6.250e-02 mpch
Width: 6.250e-02 mpchcm
Width: 8.878e-02 mpc
Width: 8.878e-02 mpccm
Width: 9.961e-02 aye
Width: 6.250e+01 kpch
Width: 6.250e+01 kpchcm
Width: 8.878e+01 kpc
Width: 8.878e+01 kpccm
Width: 6.250e+04 pch
Width: 6.250e+04 pchcm
Width: 8.878e+04 pc
```

Now, let's make some plots.

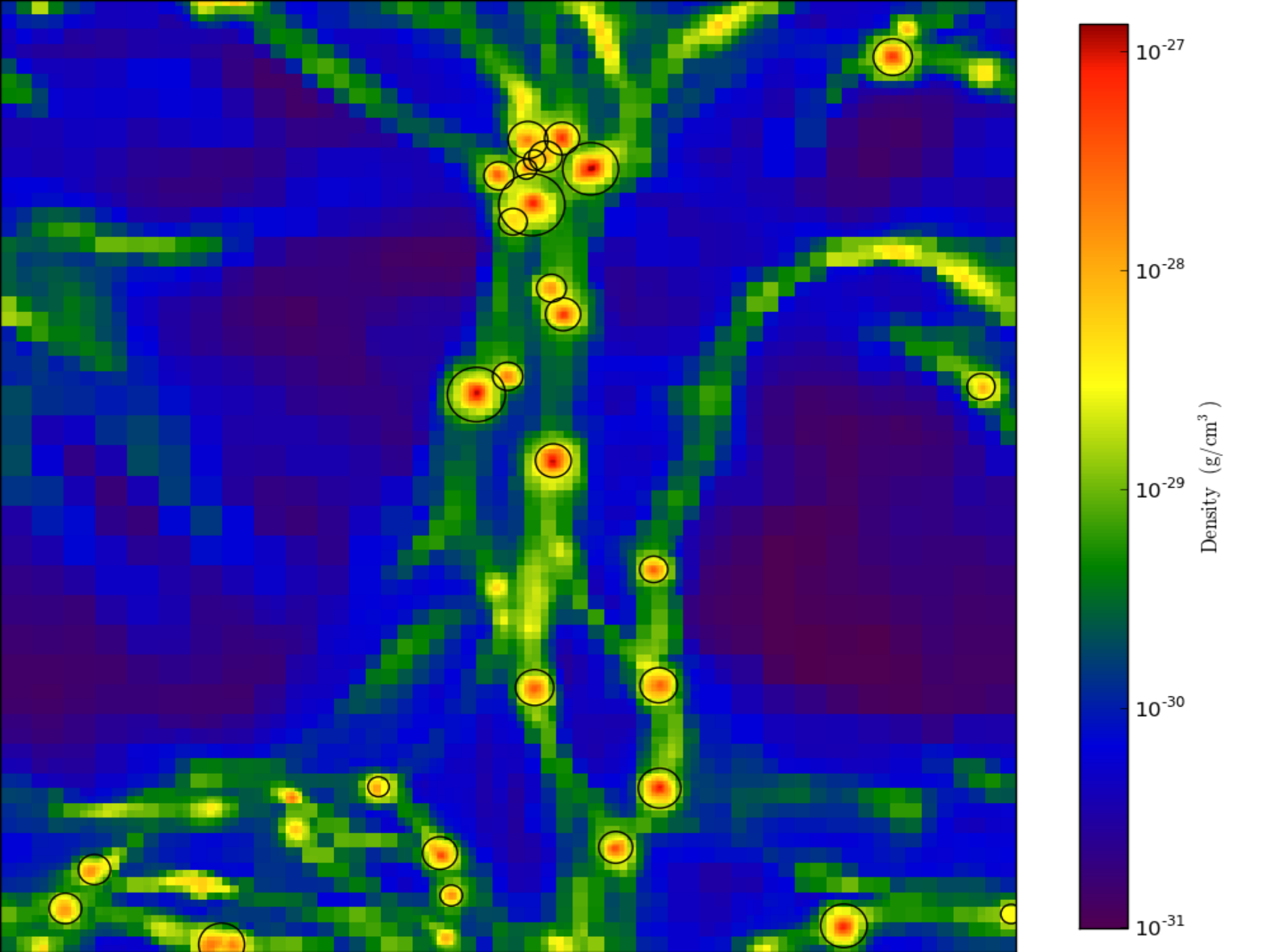
```
from yt.mods import *

pf = load("DD0023/DD0023")

pc = PlotCollection(pf, [0.5, 0.5, 0.5])
p = pc.add_slice("Density", 2)
p.modify["velocity"]()
pc.save()
```


10^{-29}
10^{-30}
10^{-31}
Density (g/cm^3)

But it's cosmology, so can we find halos, too?

```
from yt.mods import *

pf = load("DD0023/DD0023") # load data
halos = HaloFinder(pf)

pc = PlotCollection(pf, [0.5, 0.5, 0.5])
p = pc.add_projection("Density", 1, "Density")
p.modify["hop_circles"](halos)

pc.save()
```

```
Copying arrays for 32768 particles
Building Tree...
Finding Densities...
Finding Densest Neighbors...
Grouping...
Merging Groups...
Writing Output...
All Done!
```


10^{-27}
10^{-28}
10^{-29}
10^{-30}
10^{-31}
Density (g/cm^3)

Let’s dig deeper into our data.

```
from yt.mods import *
pf = load("DD0095/DD0095")

grid1 = pf.h.grids[0]
print "%s : %s, %s" % (grid1, grid1.LeftEdge, grid1.ActiveDimensions)
print grid1["Density"].max()

for child in grid1.Children:
    print "  %s: %s" % (child, child.ActiveDimensions)
```

```
EnzoGrid_0001 : [ 0.  0.  0.], [64 64 32]
7.60466650304e-28
  EnzoGrid_40096: [10 10 14]
  EnzoGrid_40097: [12 10  4]
  EnzoGrid_40098: [6 6 6]
  EnzoGrid_40099: [6 4 6]
  EnzoGrid_40100: [2 8 8]
  EnzoGrid_40101: [10 12  8]
  EnzoGrid_40102: [4 6 6]
  EnzoGrid_40103: [10  8  8]
  EnzoGrid_40104: [14  8 16]
  EnzoGrid_40105: [12 20  6]
  EnzoGrid_40106: [ 8  8 10]
  EnzoGrid_40107: [6 8 8]
  EnzoGrid_40108: [10  8  6]
  EnzoGrid_40109: [18 10 14]
  EnzoGrid_40110: [ 8  6 14]
  EnzoGrid_40111: [ 8  8 12]
  EnzoGrid_40112: [ 8 12  8]
  EnzoGrid_40113: [8 6 6]
  EnzoGrid_40114: [6 6 6]
  EnzoGrid_40115: [12 10 10]
  EnzoGrid_40116: [14  8 10]
  EnzoGrid_40117: [20  6 12]
  EnzoGrid_40118: [8 6 8]
  EnzoGrid_40119: [ 8 12 12]
  EnzoGrid_40120: [4 6 4]
  EnzoGrid_40121: [10 12  8]
```

Simple Scripting: Data Objects

```
from yt.mods import *

pf = load("DD0023/DD0023")

sp = pf.h.sphere('max', 1.0/pf["mpc"])

print sp["Density"]
print sp["Density"].max(), sp["Density"].min()
```

```
[  2.52042852e-29   3.23860958e-29   2.59744070e-29 ...,   1.55367021e-28
   1.12697623e-28   9.40910694e-29]
2.24652781982e-27 1.89361383963e-29
```

```
from yt.mods import *

pf = load("DD0023/DD0023")

sp = pf.h.sphere('max', 1.0/pf["mpc"])

baryon_mass, particle_mass = sp.quantities["TotalQuantity"](
        ["CellMassMsun", "ParticleMassMsun"])

print "Total mass in sphere is %0.5e (gas = %0.5e / particles = %0.5e)" % \
            (baryon_mass + particle_mass, baryon_mass, particle_mass)
```

```
Total mass in sphere is 5.31028e+13 (gas = 7.99936e+12 / particles = 4.51034e+13)
```

```
from yt.mods import *
pf = load("DD0023/DD0023")
sp = pf.h.sphere('max', 1.0/pf["mpc"])

print sp.quantities.keys()
```

```
['MinLocation', 'StarAngularMomentumVector', 'IsBound', 'BulkVelocity', 'AngularMomentumVector', 'TotalQua
```

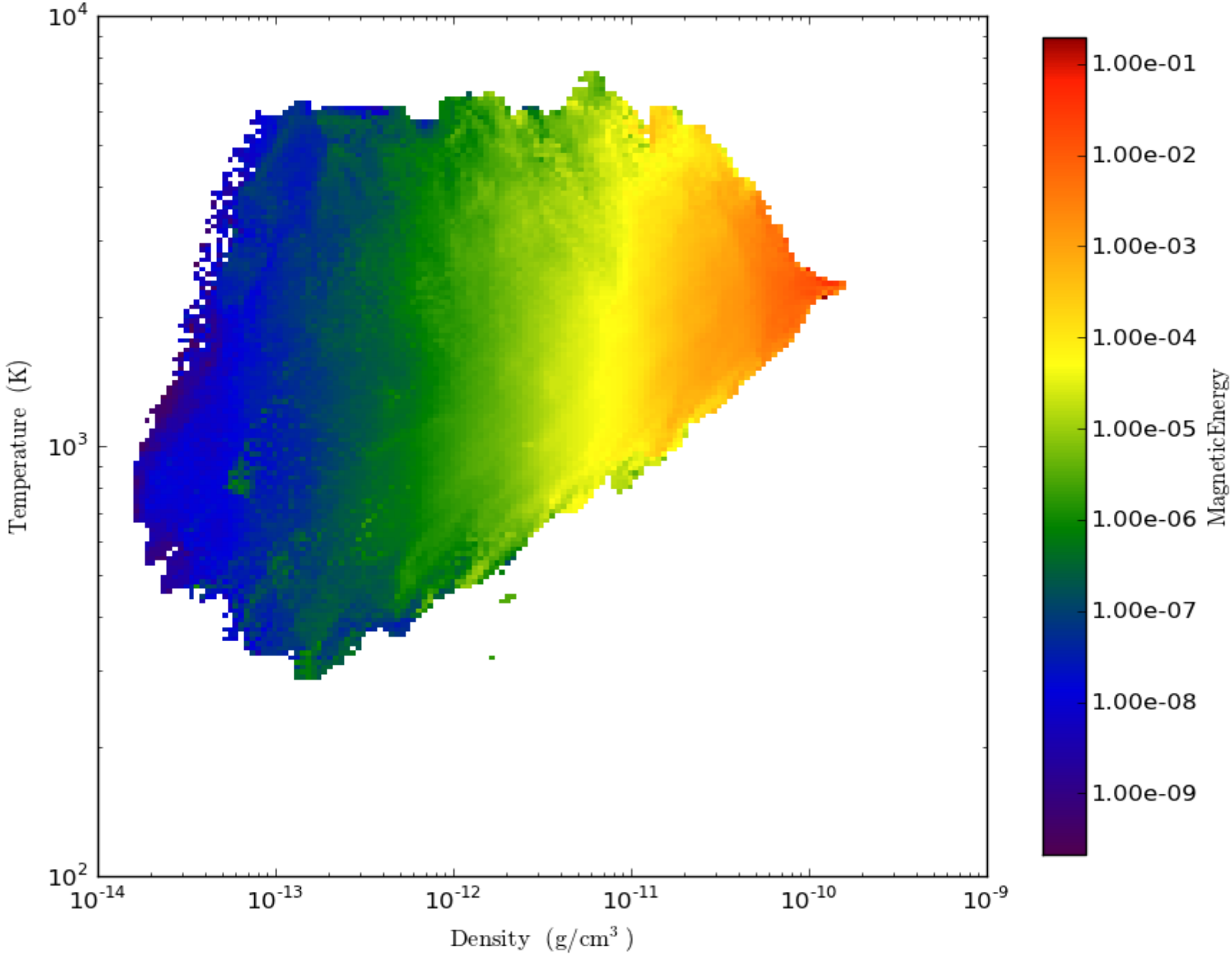
```
from yt.mods import *
pf = load("DD0087/DD0087")
sp = pf.h.sphere("max", 100.0/pf['au'])
L = sp.quantities["AngularMomentumVector"]()
print "Angular Momentum Vector:", L
```

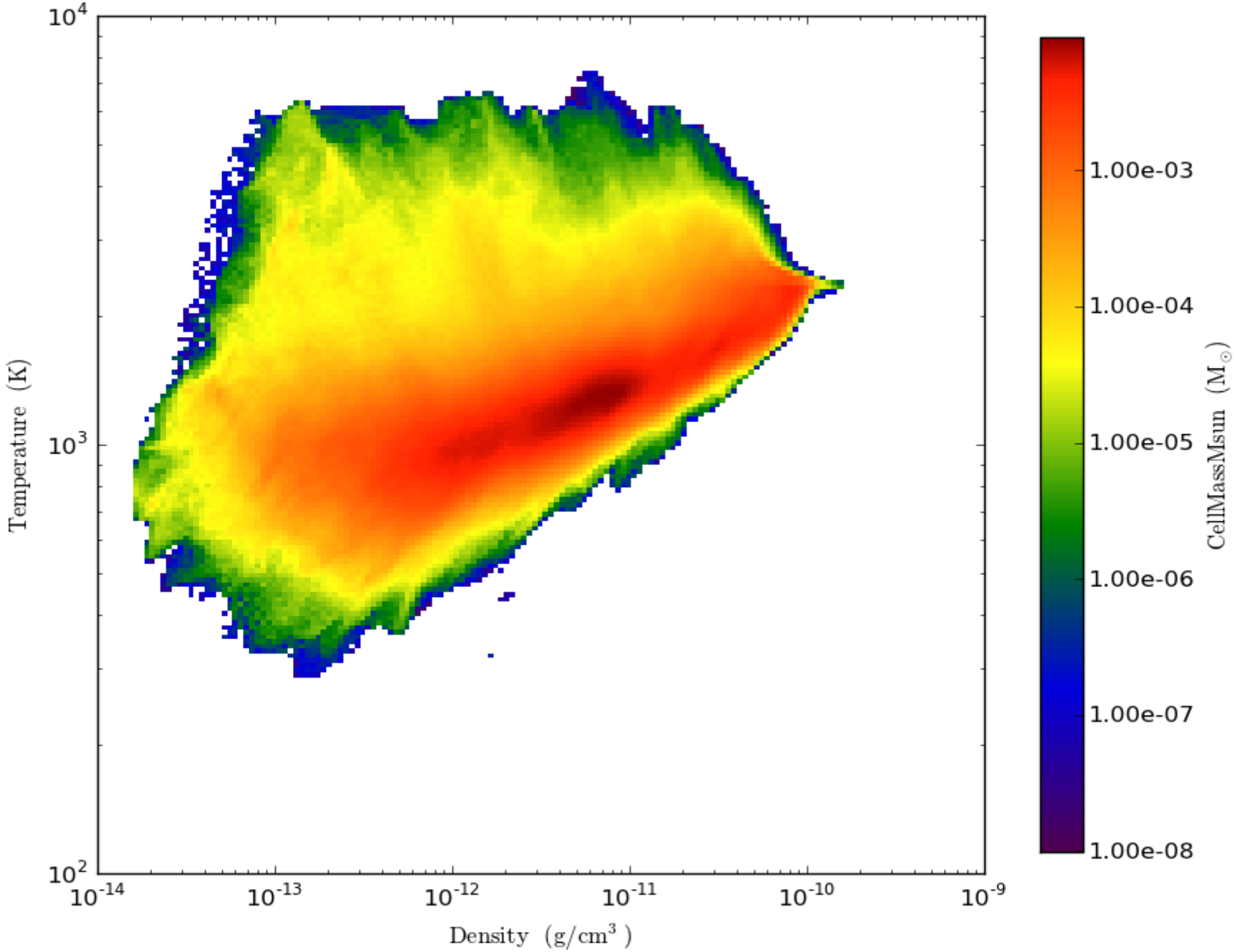
```
Angular Momentum Vector: [ 0.38855402  0.23911532 -0.88985934]
```

```
from yt.mods import *

@derived_field(name = "Dinosaurs", units = "TRexPerS")
def Dinosaurs(field, data):
    return (data["Density"]**(2./3) /
            data["Temperature"]**(0.5))

pf = load("DD0023/DD0023")
dd = pf.h.all_data()

dinosaurs, = dd.quantities["TotalQuantity"](["Dinosaurs"])
print "Total T-Rexes per Second: %0.3e" % (dinosaurs)
```

```
Total T-Rexes per Second: 2.382e-16
```

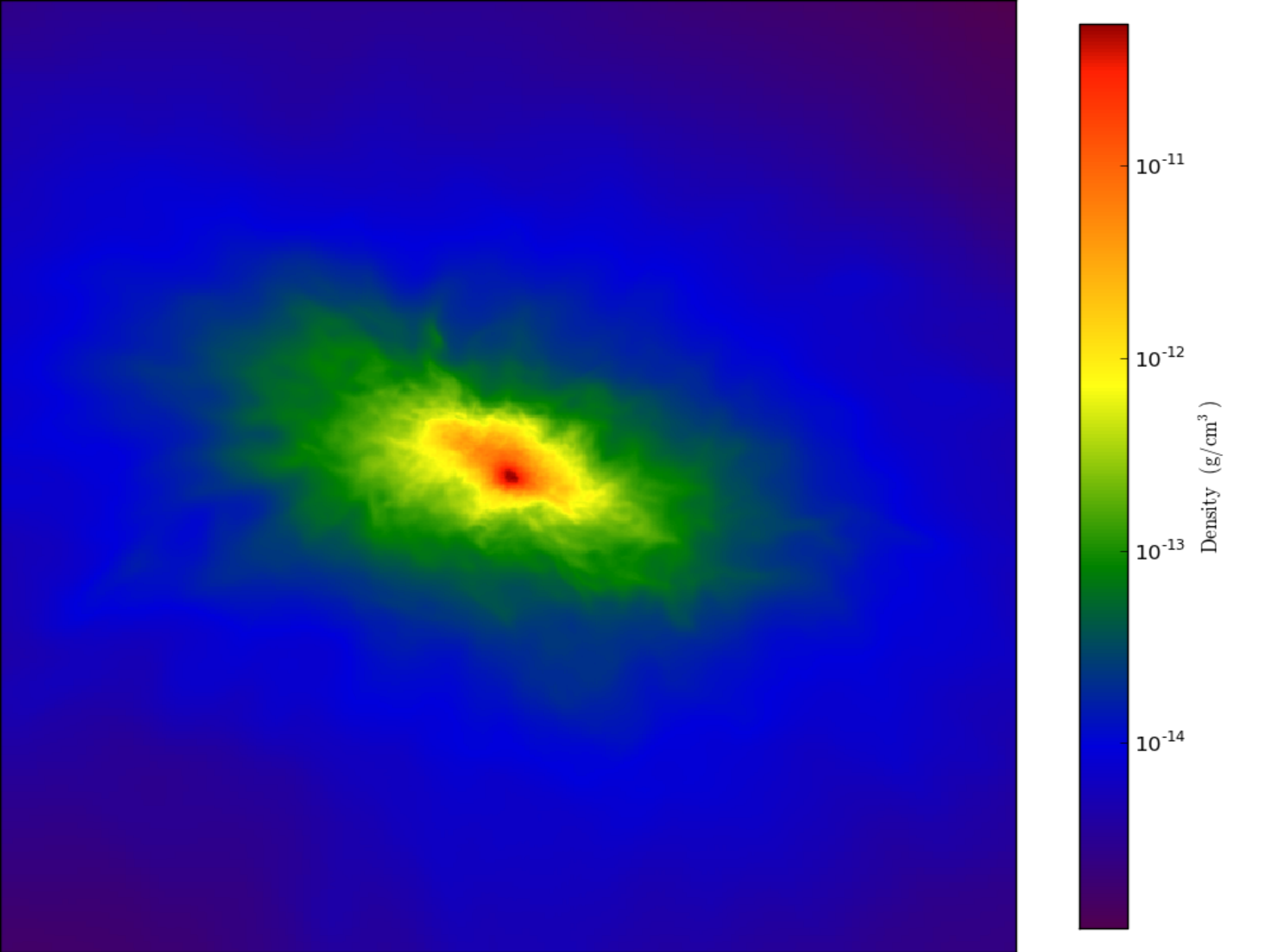
```
from yt.mods import *

pf = load("DD0087/DD0087")
v, c = pf.h.find_max("Density")
pc = PlotCollection(pf, c)

pc.add_phase_sphere(100.0, "au",
    ["Density", "Temperature", "MagneticEnergy"])

pc.add_phase_sphere(100.0, "au",
    ["Density", "Temperature", "CellMassMsun"],
    weight = None)

pc.save()
```


10^4
10^3
10^2
Temperature (K)
10^{-14}
10^{-13}
10^{-12}
10^{-11}
10^{-10}
10^{-9}
Density (g/cm^3)
1.00e-01
1.00e-02
1.00e-03
1.00e-04
1.00e-05
1.00e-06
1.00e-07
1.00e-08
1.00e-09
MagneticEnergy

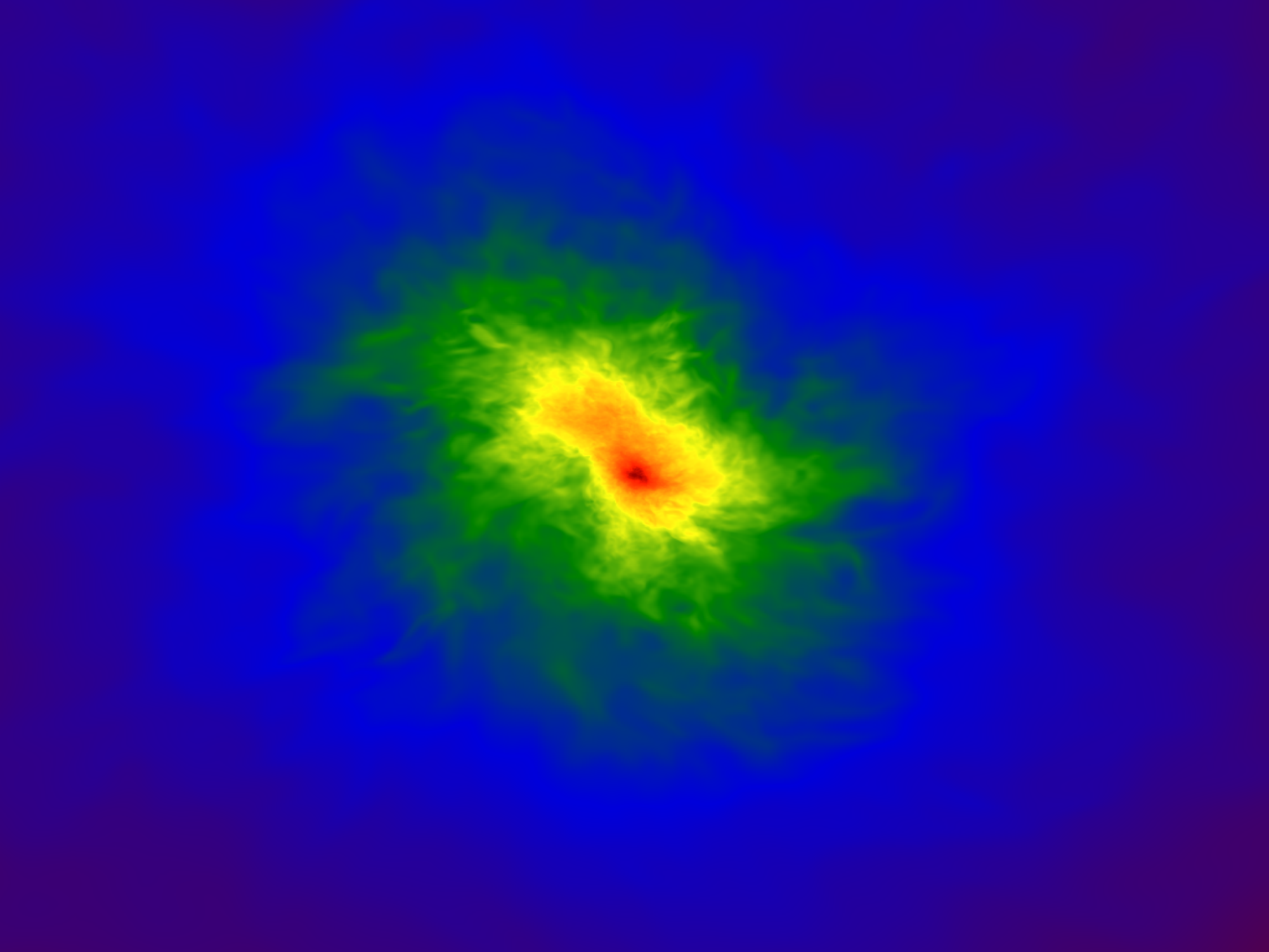

10^4
10^3
10^2
Temperature (K)
10^{-14}
10^{-13}
10^{-12}
10^{-11}
10^{-10}
10^{-9}
Density (g/cm^3)
1.00e-03
1.00e-04
1.00e-05
1.00e-06
1.00e-07
1.00e-08
CellMassMsun ($M_\odot$)

Advanced Plots

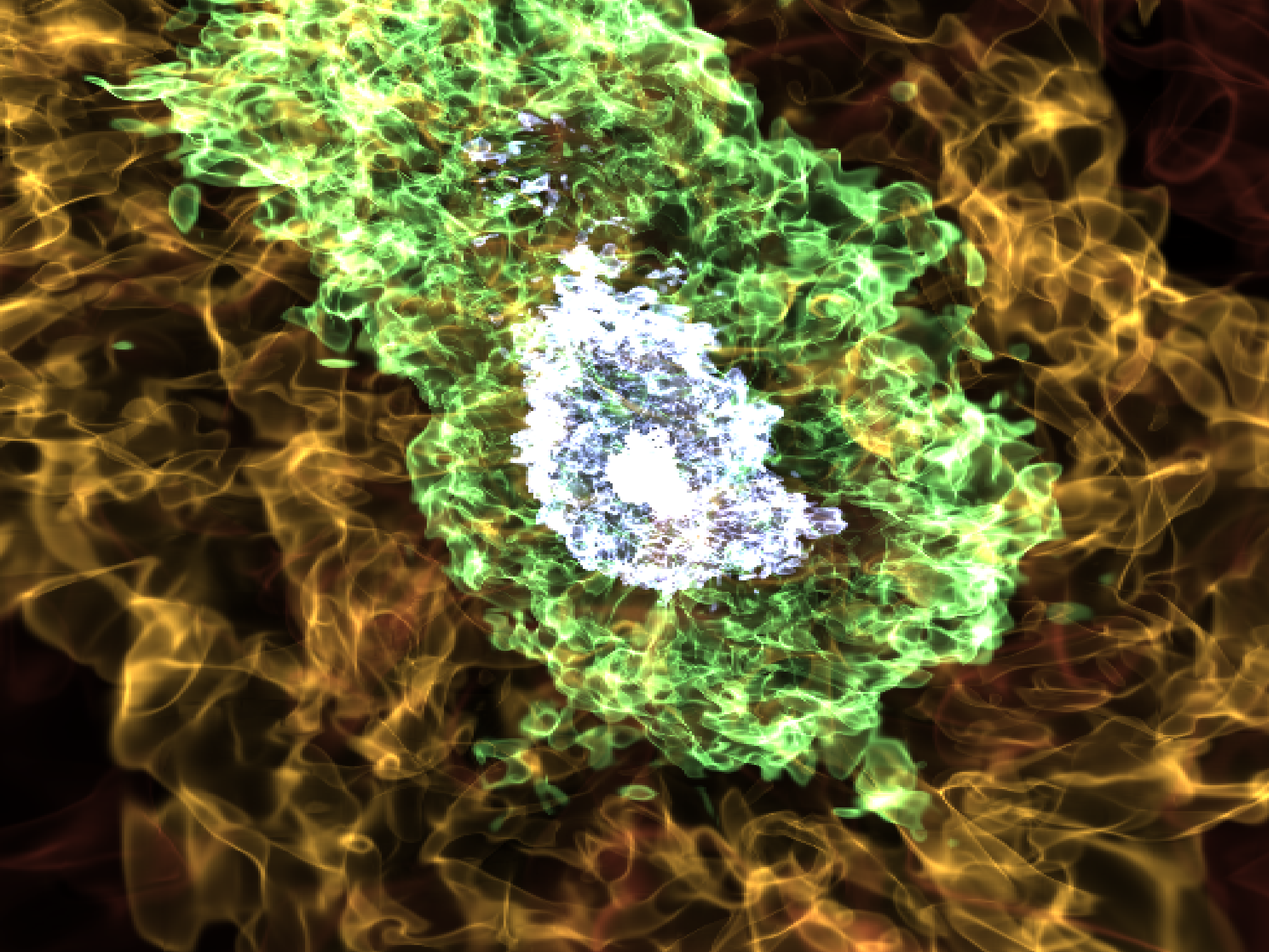
```
from yt.mods import *

pf = load("DD0087/DD0087")
v, c = pf.h.find_max("Density")
pc = PlotCollection(pf, c)
pc.add_projection("Density", 0, "Density")
pc.set_width(1000.0, 'au')
pc.save()
```


10^{-11}
10^{-12}
10^{-13}
10^{-14}
Density (g/cm^3)

```
from yt.mods import *

pf = load("DD0087/DD0087")
v, c = pf.h.find_max("Density")

sp = pf.h.sphere("max", (1000.0, 'au'))
L = sp.quantities["AngularMomentumVector"]()

image = off_axis_projection(pf, c, L, 1000.0/pf['au'],
        1024, "Density", "Density")
write_image(na.log10(image), "DD0087_offaxis.png")
```



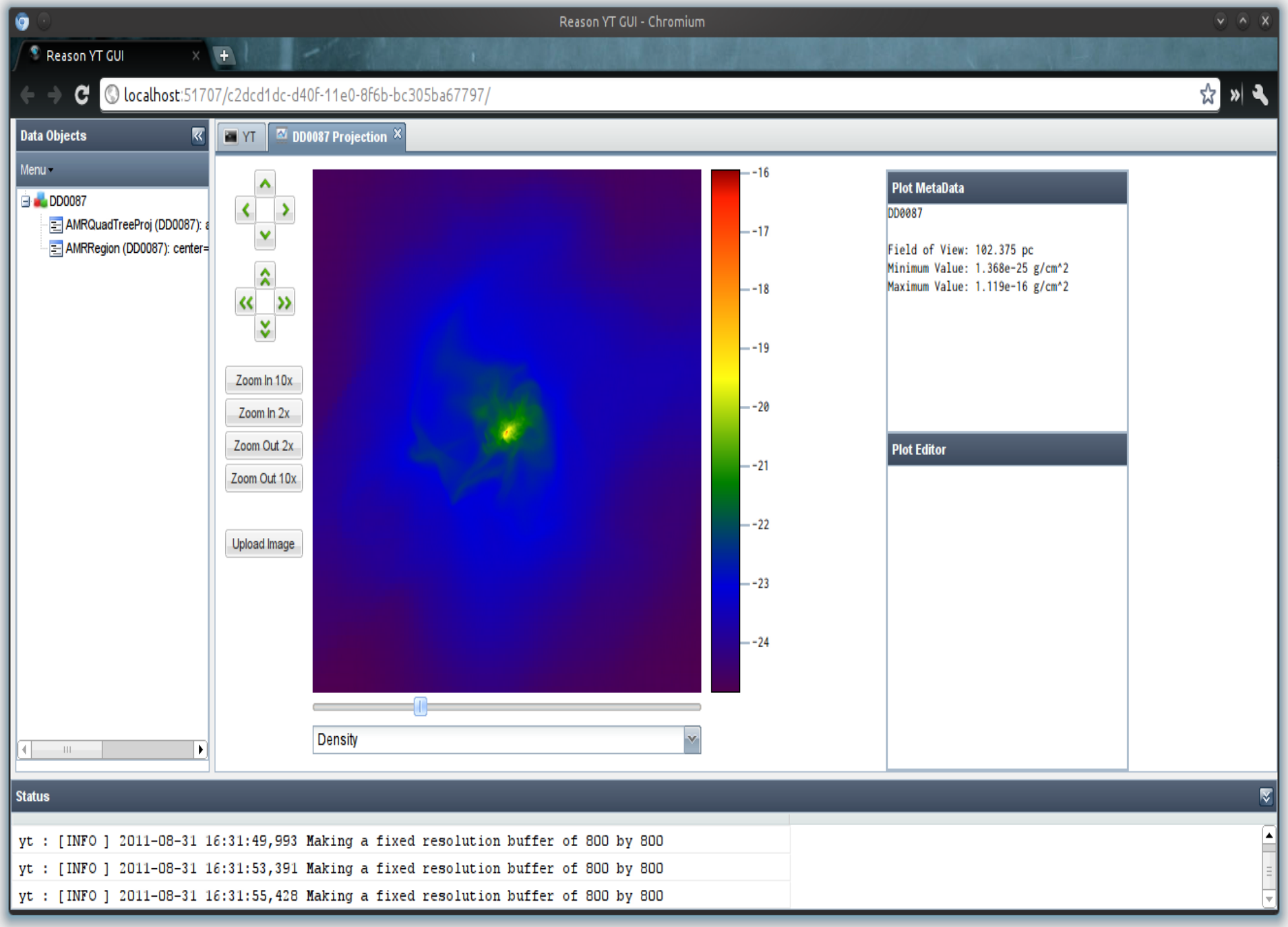
```
from yt.mods import *
pf = load("DD0087/DD0087")
v, c = pf.h.find_max("Density")
sp = pf.h.sphere(c, (250, 'au'))
L = sp.quantities["AngularMomentumVector"]()

tf = ColorTransferFunction((-14, -10))
tf.add_layers(6, colormap="kamae", w=0.001)

cam = pf.h.camera(c, L, 200.0/pf['au'], (512, 512), tf)
cam.snapshot("DD0087_isocontour.png", 2.0)
```



Fun!

yt mapserver

yt serve


Reason YT GUI - Chromium
Reason YT GUI
localhost:51707/c2dcd1dc-d40f-11e0-8f6b-bc305ba67797/
Data Objects
Menu
DD0087
AMRQuadTreeProj (DD0087): a
AMRRegion (DD0087): center=
YT
DD0087 Projection
Zoom In 10x
Zoom In 2x
Zoom Out 2x
Zoom Out 10x
Upload Image
-16
-17
-18
-19
-20
-21
-22
-23
-24
Density
Plot MetaData
DD0087
Field of View: 102.375 pc
Minimum Value: 1.368e-25 g/cm^2
Maximum Value: 1.119e-16 g/cm^2
Plot Editor
Status
yt : [INFO] 2011-08-31 16:31:49,993 Making a fixed resolution buffer of 800 by 800
yt : [INFO] 2011-08-31 16:31:53,391 Making a fixed resolution buffer of 800 by 800
yt : [INFO] 2011-08-31 16:31:55,428 Making a fixed resolution buffer of 800 by 800

Places to go from here

- Documentation: tutorial, advanced examples, cookbook, etc
- Support venues: `yt-users`, `yt-dev`, IRC
- `yt` Hub
- `yt` workshop, Chicago, January 24-26 2012!