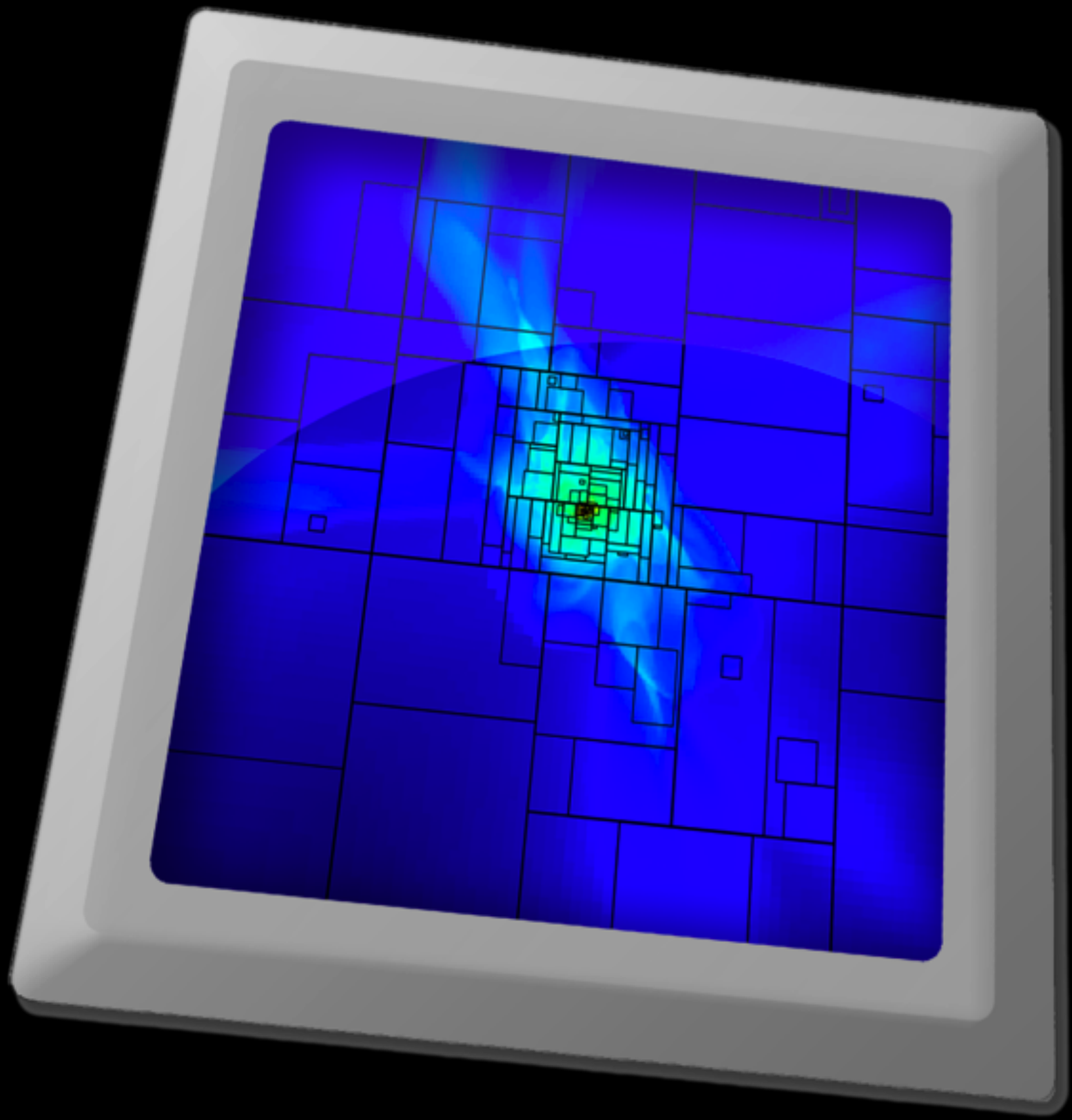
Simulating and Analyzing the First Stars in the Universe

Matthew Turk (NSF / Columbia)
and the Enzo and yt collaborations

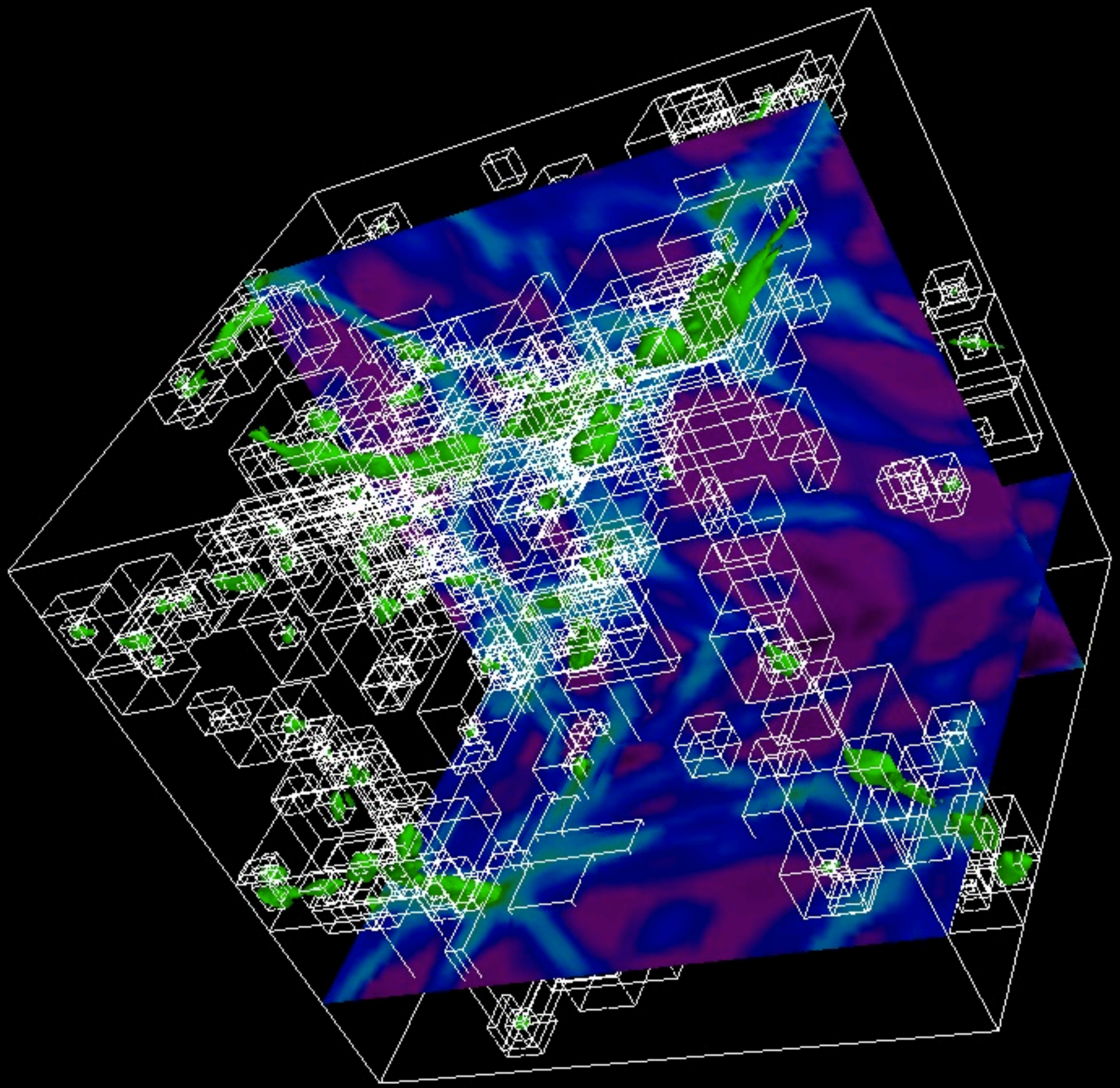
Analysis.



yt

astro-ph/1011.3514

yt.enzotools.org



yt has been designed to address physical, not computational, entities.

Enzo, Orion, CASTRO, FLASH

Chombo, Tiger, ART, RAMSES

yt is designed to be the *lingua franca*
of astrophysical codes.

Objects

Orthogonal Rays
Non-orthogonal Rays

1D

Slices
Oblique Slices
Projections

2D

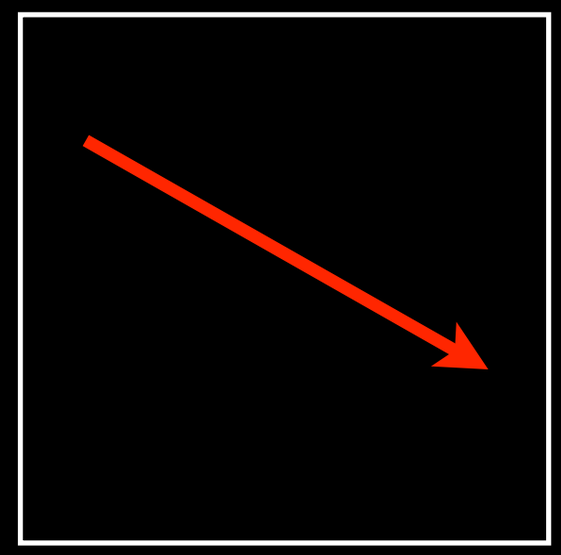
Spheres
Rectangular Prisms
Disks/Cylinders
Inclined Boxes
Clumps
Extracted Regions

3D

Objects

All respect unified interface:

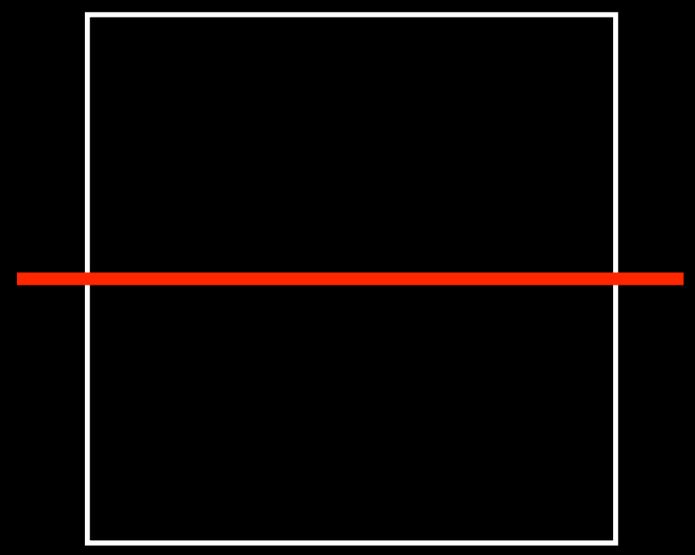
```
from yt.mods import *
pf = load(“DataDump0155.dir/DataDump0155”)
ray = pf.h.ray([0.1, 0.2, 0.5],
               [0.4, 0.9, 0.1])
print ray[“Density”]
```



Objects

All respect unified interface:

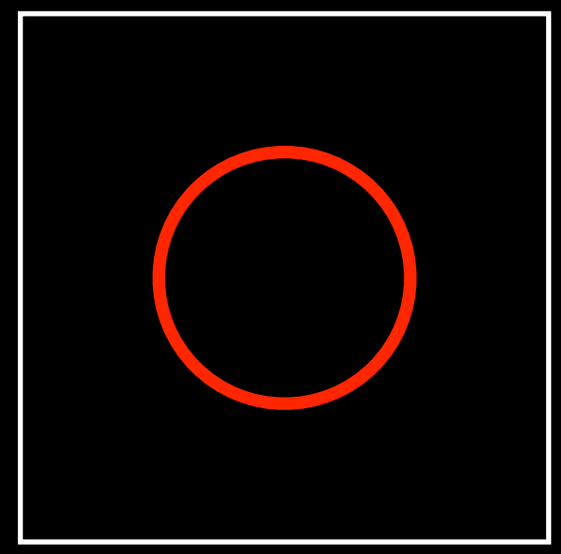
```
from yt.mods import *
pf = load(“DataDump0155.dir/DataDump0155”)
sl = pf.h.slice(0, 0.5)

print sl[“Density”]
```



Objects

All respect unified interface:

```
from yt.mods import *
pf = load(“DataDump0155.dir/DataDump0155”)
sp = pf.h.sphere(100.0/pf[‘au’], 0.5)

print sp[“Density”]
```



```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
v, c = pf.h.find_max("Density")
```

```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
v, c = pf.h.find_max("Density")
```

Load yt,
populate our namespace,
read configuration.

```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
v, c = pf.h.find_max("Density")
```

Read parameter file,
create grid objects,
set up units.

```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
v, c = pf.h.find_max("Density")
```

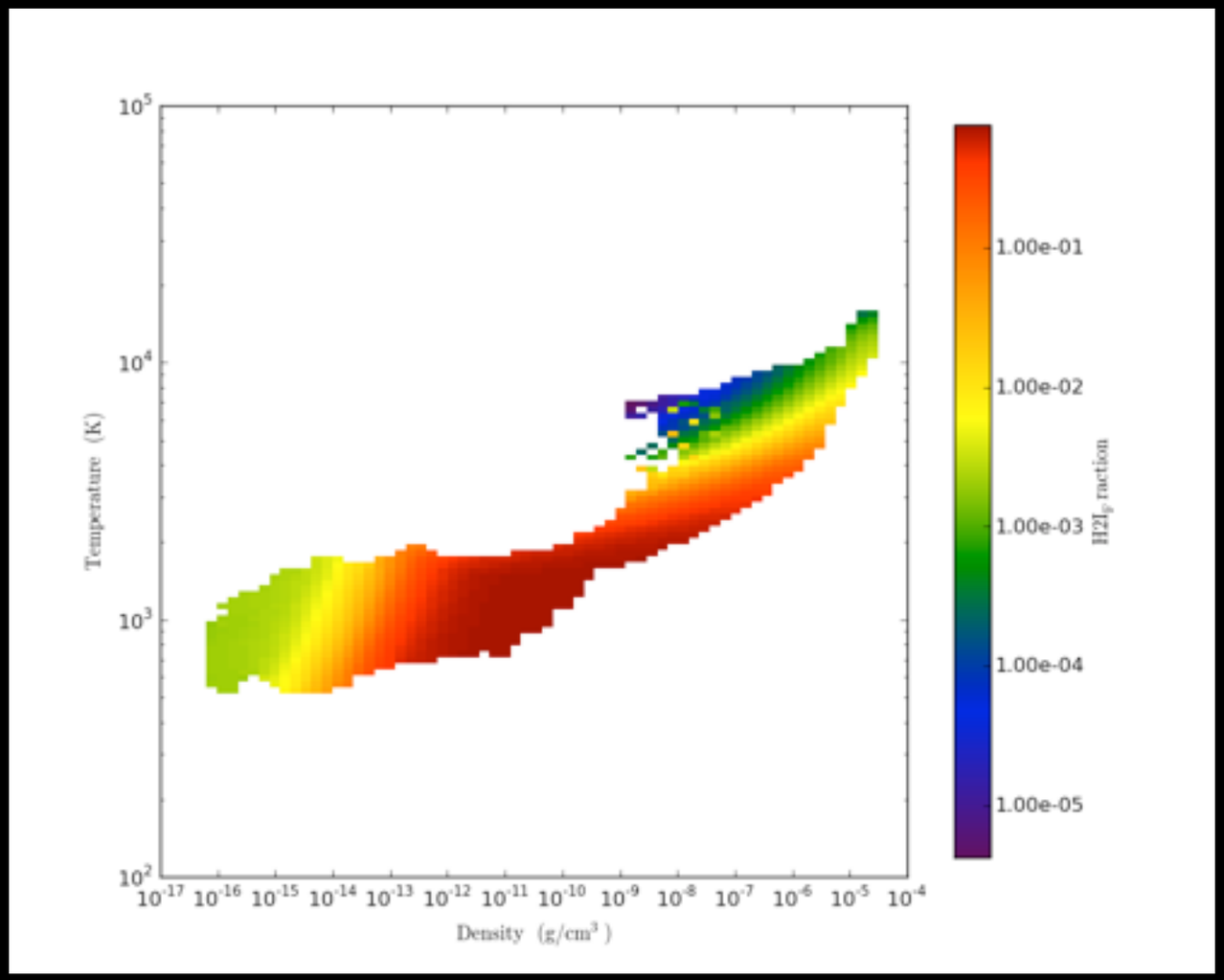
Address the most dense point.

Adding new fields should be easy.

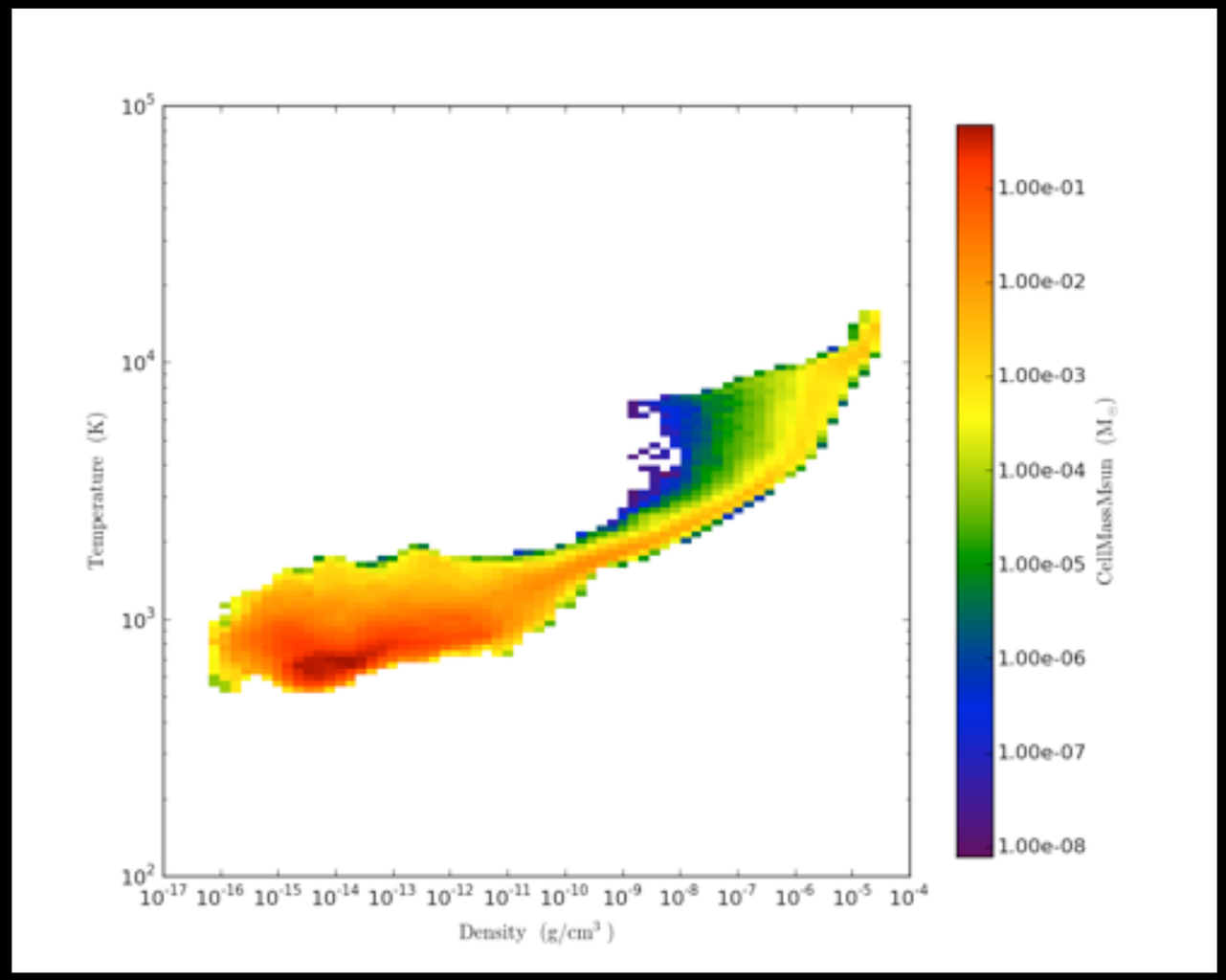
```
from yt.mods import *

@derived_field("Pressure")
def Pressure(field, data):
    return (data.pf["Gamma"] - 1.0) * \
       data["Density"]*data["ThermalEnergy"]
```

Scripts should be simple and clear.

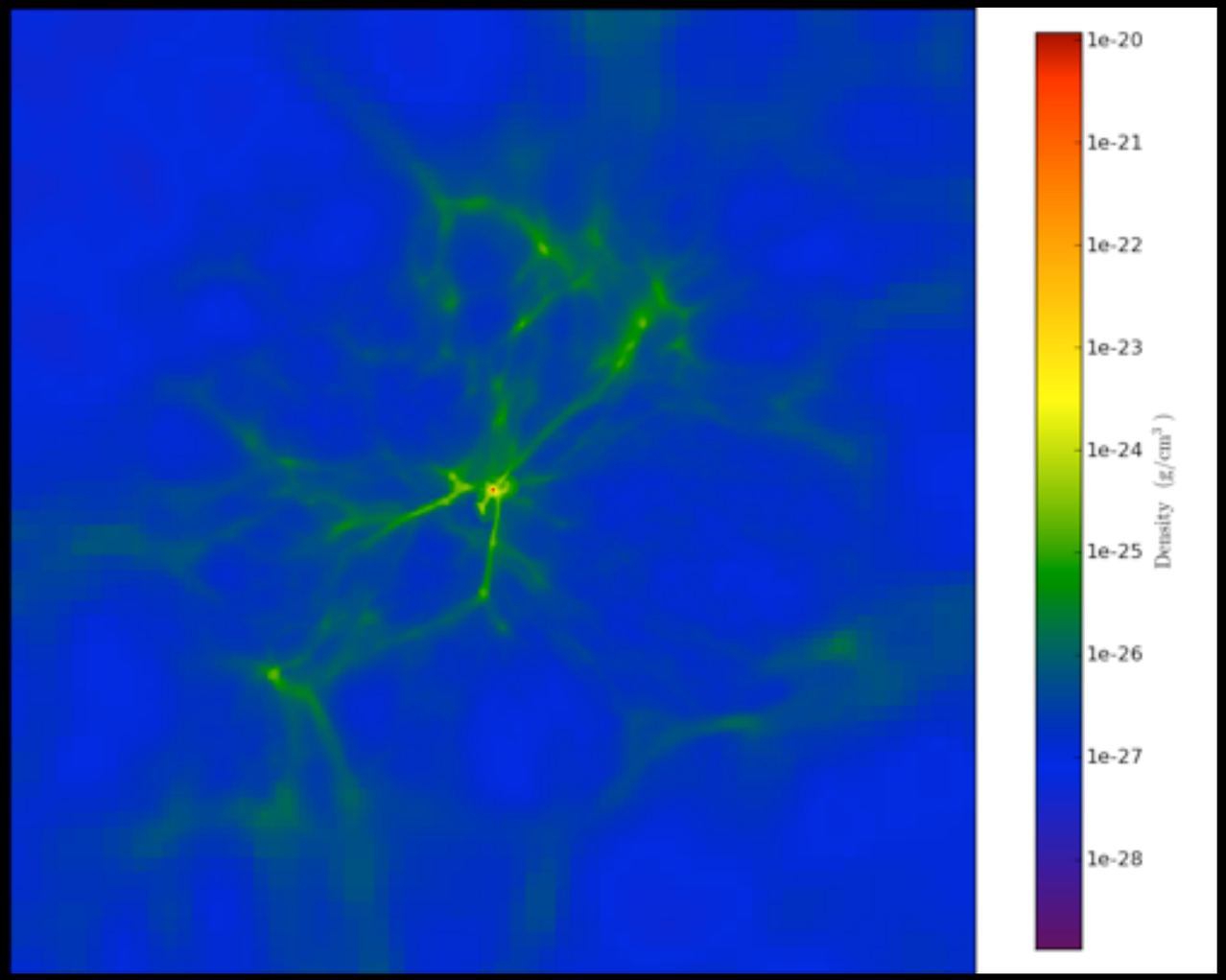
```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
pc = PlotCollection(pf)
pc.add_phase_sphere(1000.0, 'au',
 ["Density", "Temperature", "H2I_Fraction"])
pc.save()
```



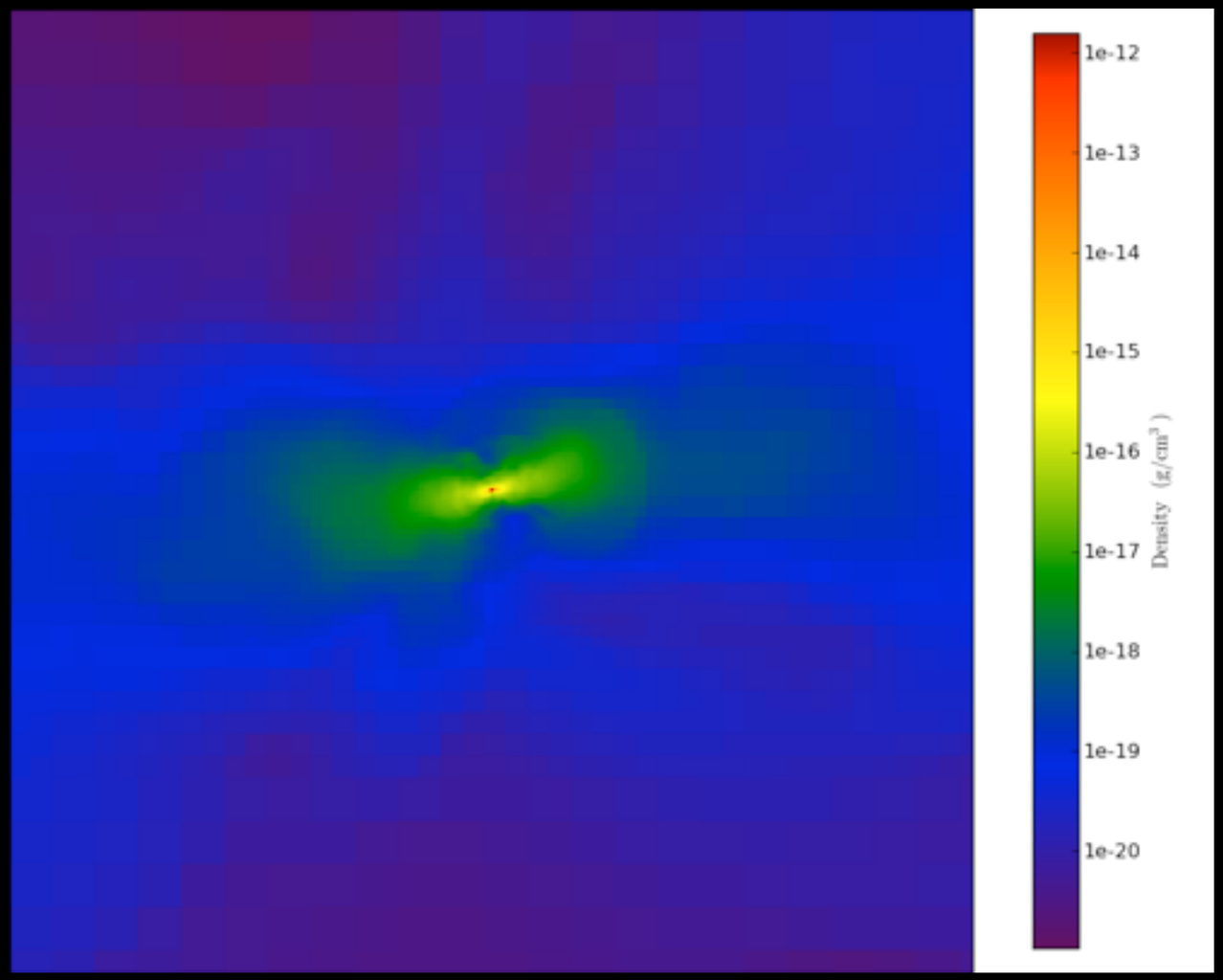
```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
pc = PlotCollection(pf)
pc.add_phase_sphere(1000.0, 'au',
 ["Density", "Temperature",
  "CellMassMsun"], weight = None)
pc.save()
```



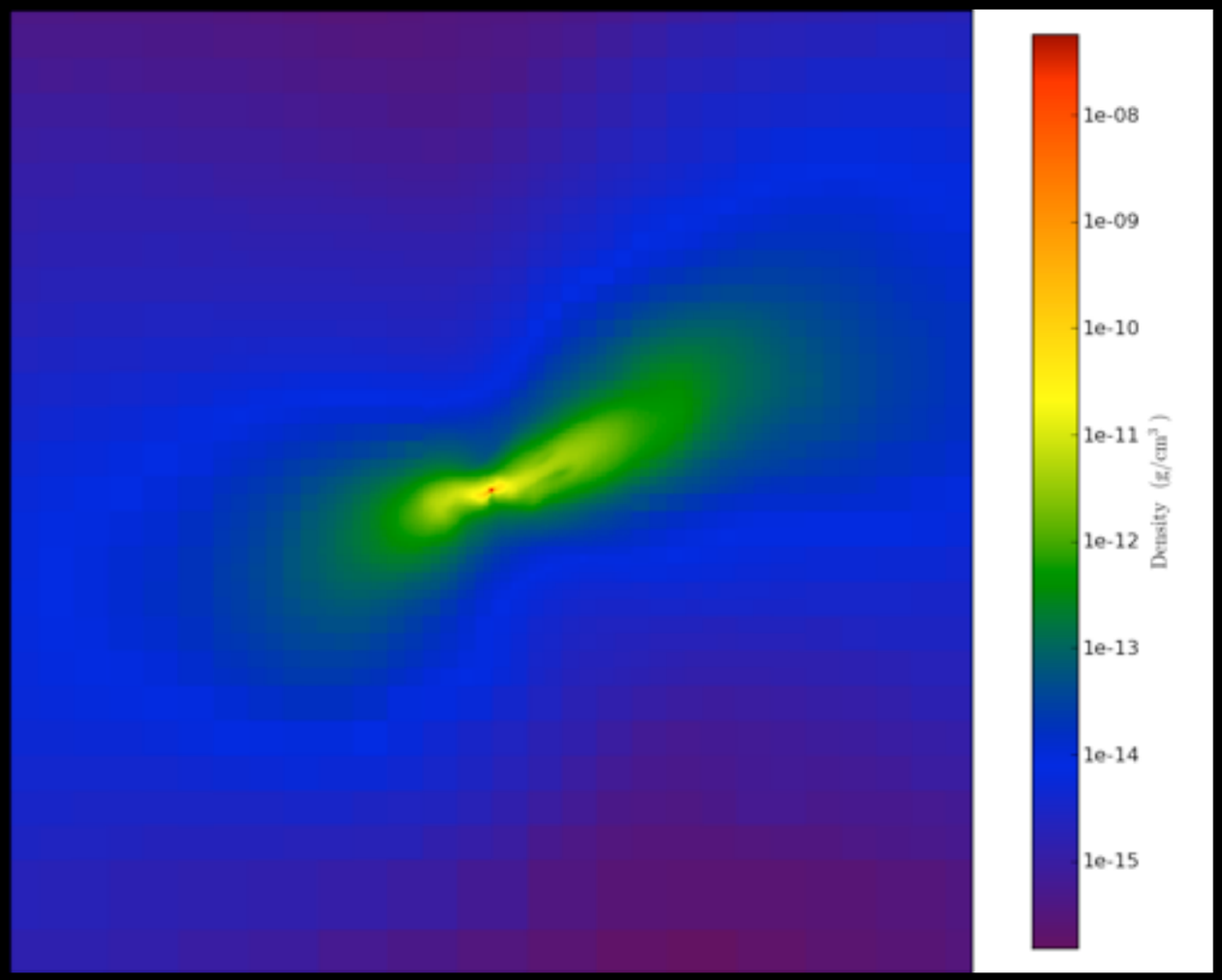
```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
pc = PlotCollection(pf)
pc.add_slice("Density", 0)
pc.set_width(0.5, 'unitary')
pc.save()
```



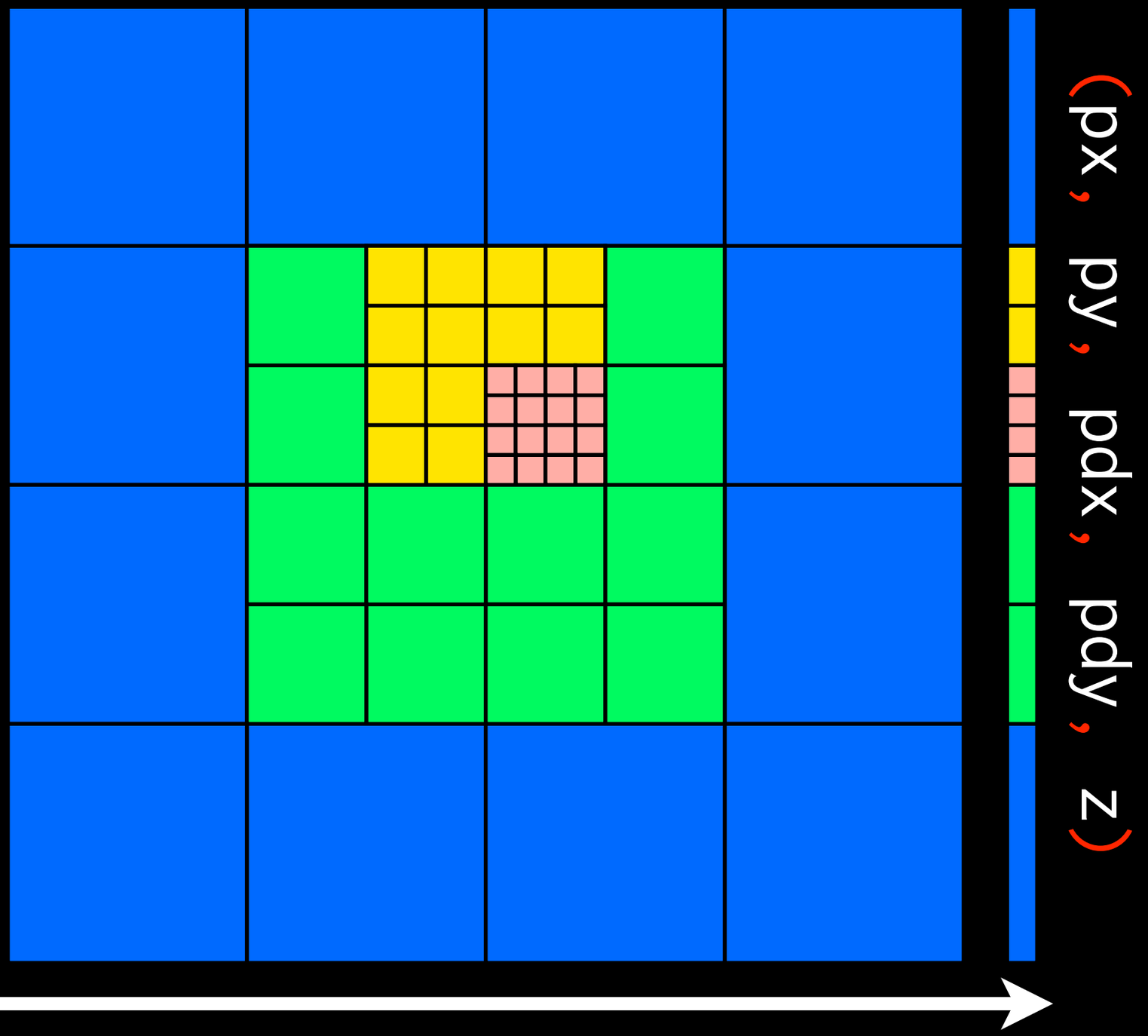
```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
pc = PlotCollection(pf)
pc.add_slice("Density", 0)
pc.set_width(1.0, 'pc')
pc.save()
```



```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
pc = PlotCollection(pf)
pc.add_slice("Density", 0)
pc.set_width(1000.0, 'au')
pc.save()
```



Projections



(px, py, pdx, pdy, z)

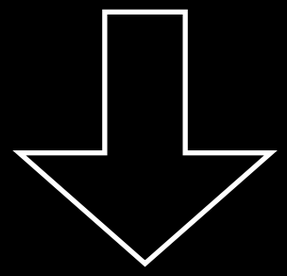
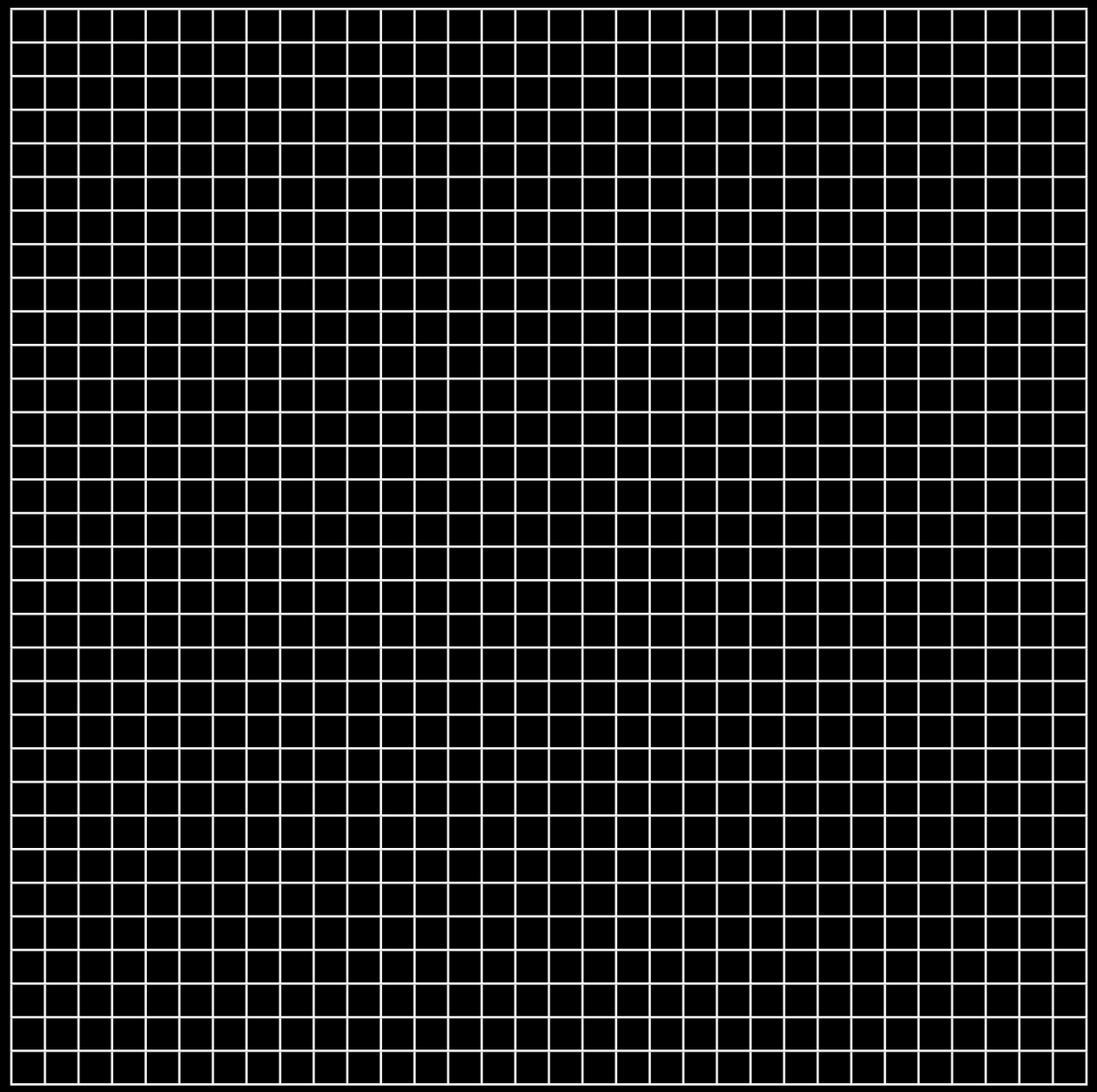


Image Buffer



(px, py, pdx, pdy, z)

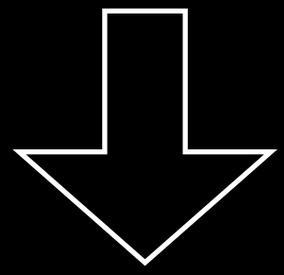
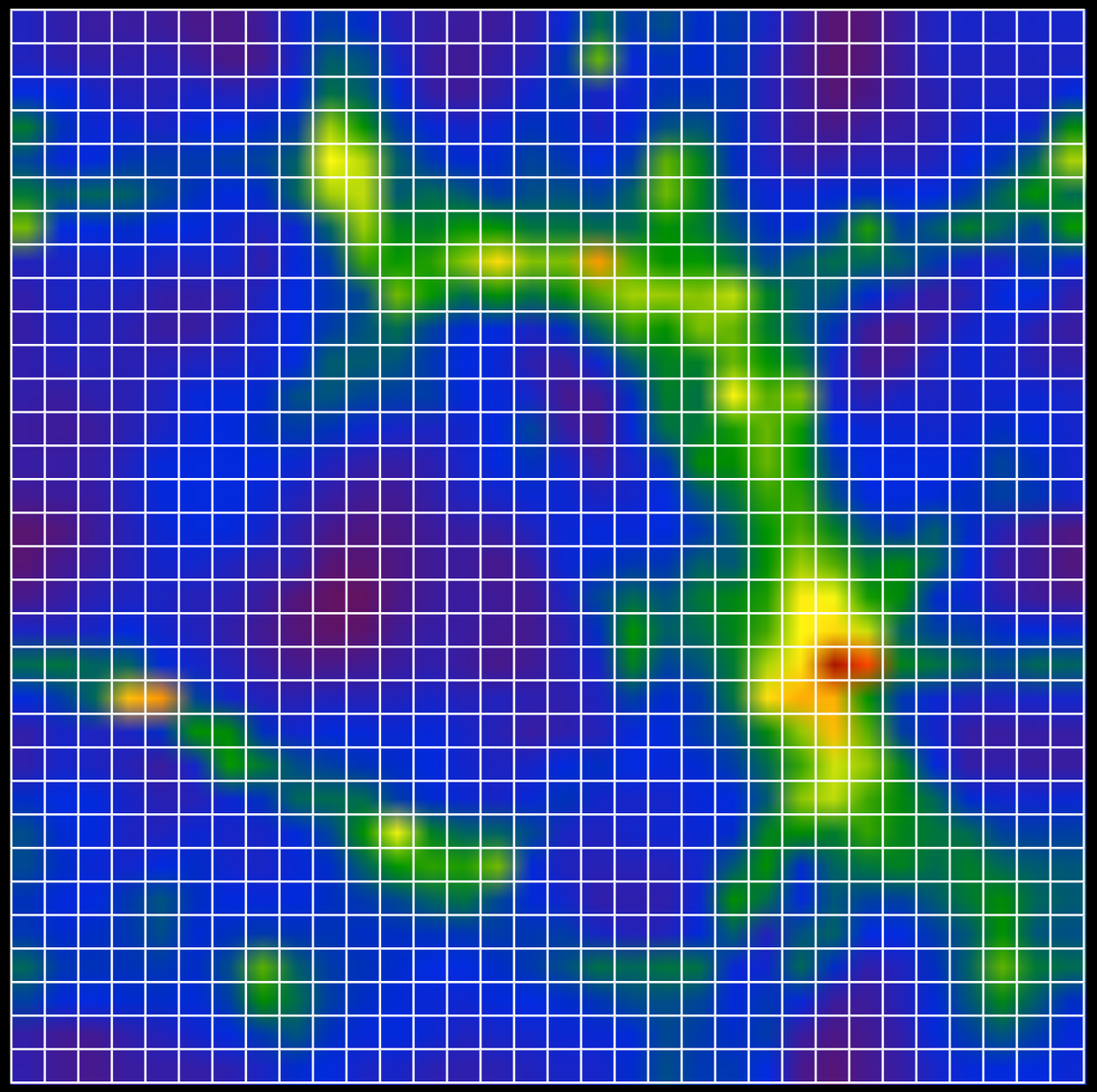
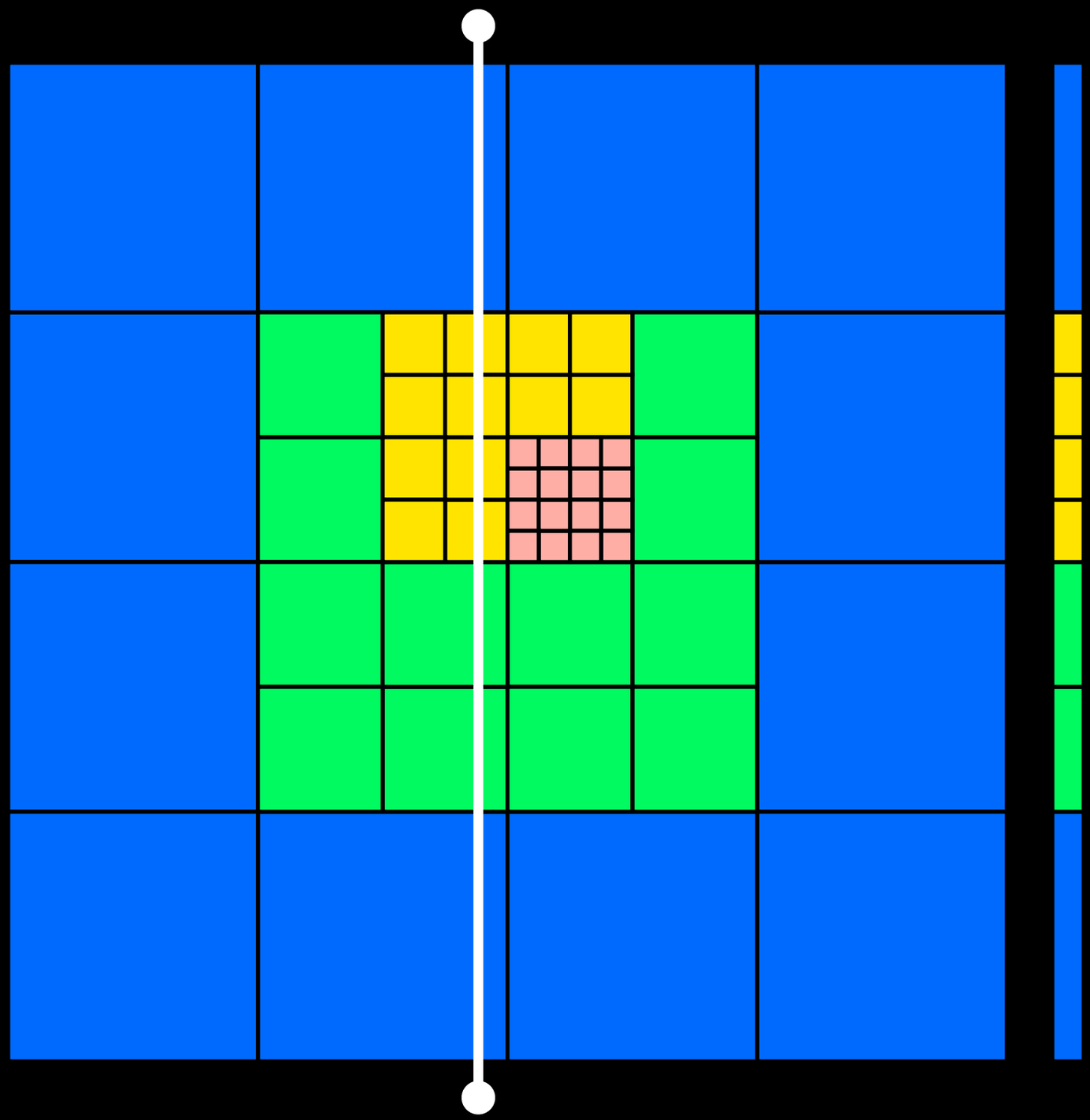


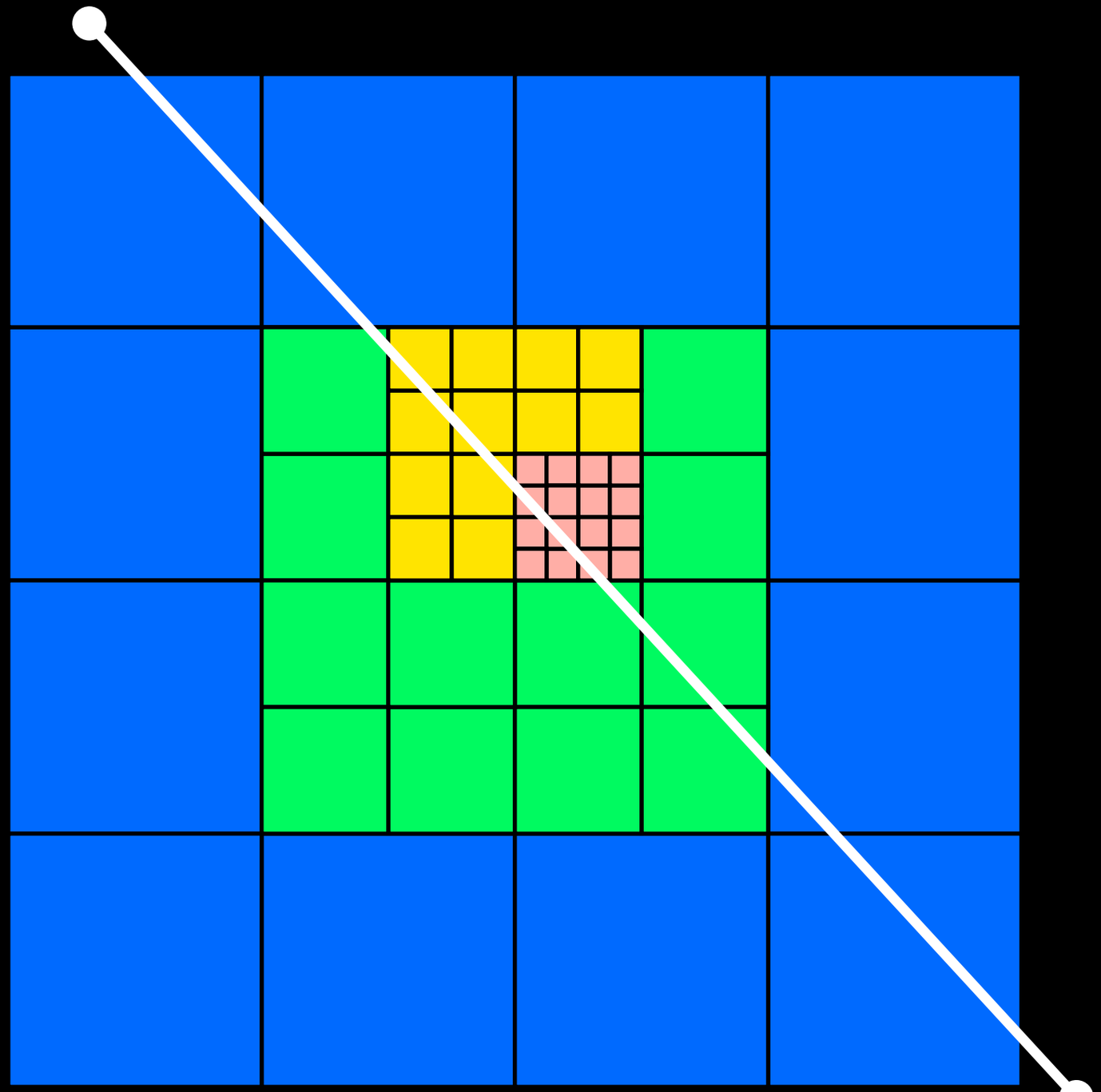
Image Buffer



Slices



Oblique Slices



Three Halo Finders:

Standard HOP

Friends of Friends

Parallel HOP

```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
halos = HaloFinder(pf)
halos.write_out("my_halos.txt")
```

```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
halos = HaloFinder(pf)

bvs = []
for halo in halos:
    sp = halo.get_sphere()
    bv = halo.quantities["BulkVelocity"]()
    bvs.append(bv)
```

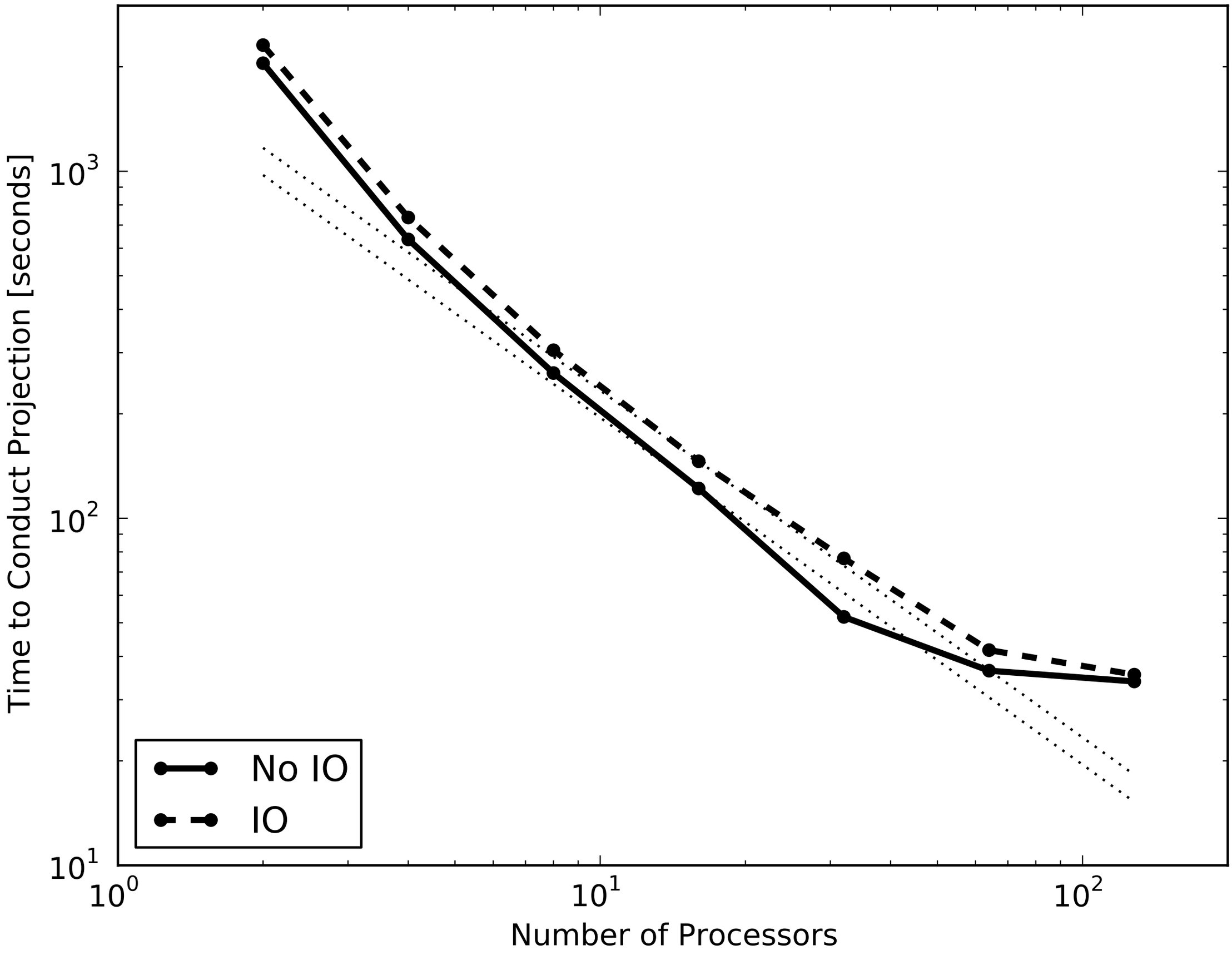
```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
halos = HaloFinder(pf)
pc = PlotCollection(pf)
bvs = []
for halo in halos:
    sp = halo.get_sphere()
    pc.add_phase_object(sp,
        ["Density", "Temperature",
         "Electron_Fraction"])
    bvs.append(bv)
```

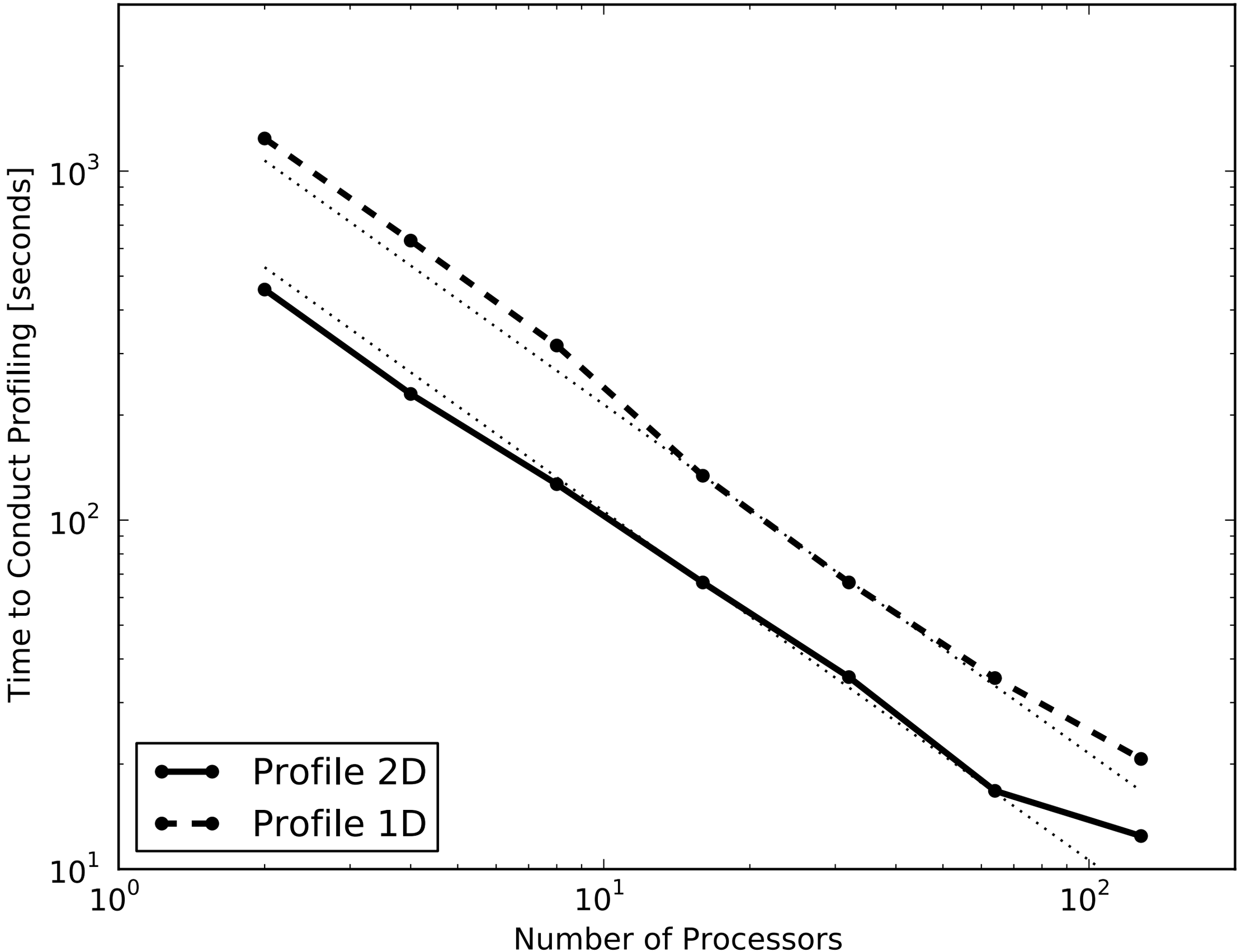
Parallelism

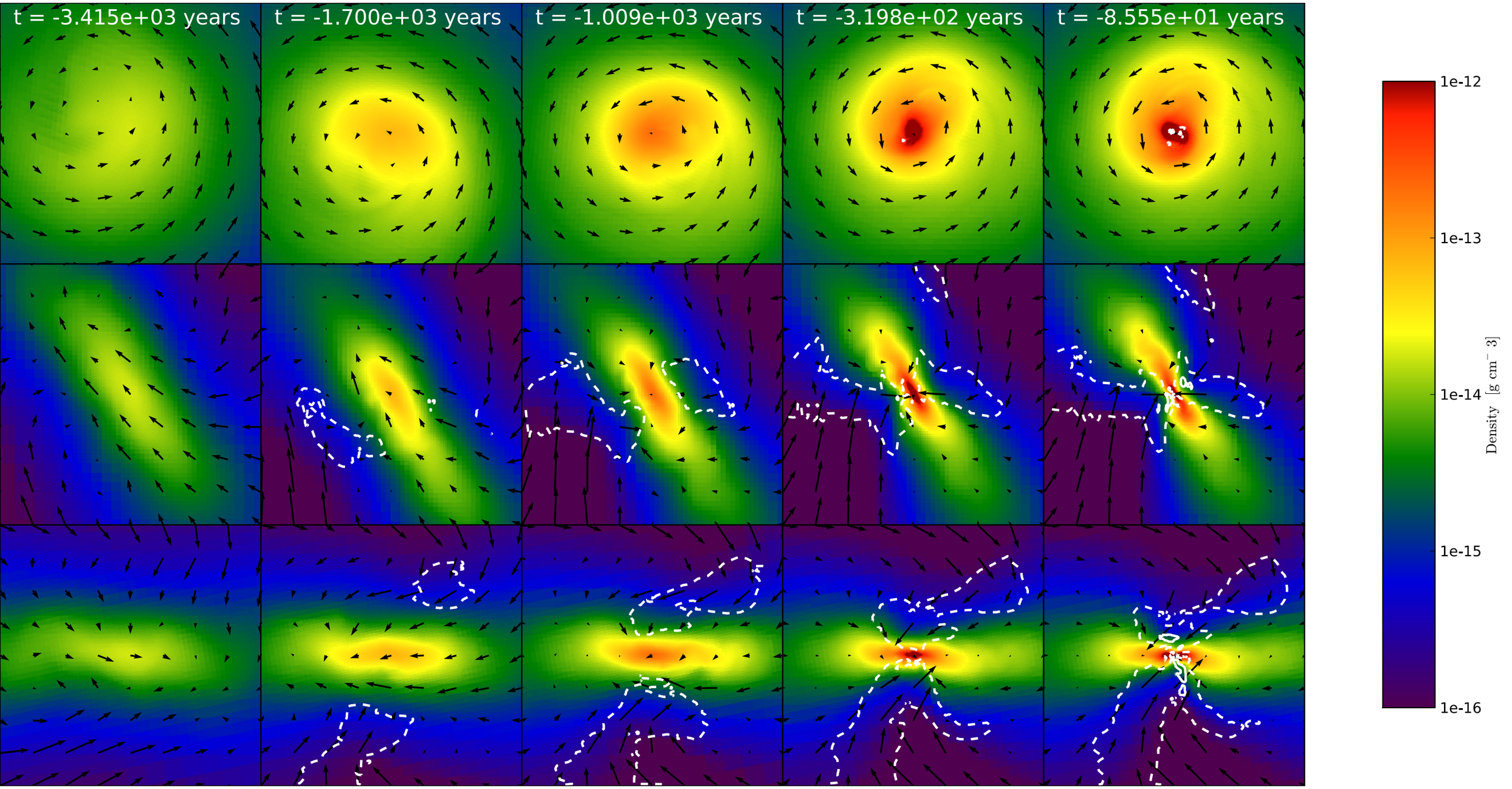
Embarassingly Parallel	Spatial Decomposition
Quantities Profiles Slices Projections Volume Rendering	Halo Finding

Simulated Observations

X-Ray Emissivity
SED Generation
Thermal gas emission
Sunrise export
SZ Compton, Kinematic SZ
Light Cones
Light Rays


Time to Conduct Projection [seconds]
10^3
10^2
10^1
10^0
10^1
10^2
Number of Processors
No IO
IO


Time to Conduct Profiling [seconds]
10^3
10^2
10^1
10^0
10^1
10^2
Number of Processors
Profile 2D
Profile 1D


t = -3.415e+03 years
t = -1.700e+03 years
t = -1.009e+03 years
t = -3.198e+02 years
t = -8.555e+01 years
1e-12
1e-13
1e-14
1e-15
1e-16
Density [g cm^-3]

Density [g/cm^{-3}]

10^{-18} 10^{-16} 10^{-14} 10^{-12} 10^{-10}

Temperature [K]

10^{3}

$cos(\theta)$ (Declination from L)

−1.0 −0.5 0.0 0.5 1.0

Density [g/cm^{-3}]

10^{-16} 10^{-14} 10^{-12} 10^{-10}

Temperature [K]

10^{3}

Radius [AU]

10^{4} 10^{3} 10^{2} 10^{1} 10^{0} 10^{-1}

10^{16} 10^{18} 10^{20} 10^{22}

S [ergs cm^{2}]

−6 −4 −2 0 2 4 6

Radial Velocity [km/s]

400 200 0 −200 −400

AU

−400 −200 0 200 400

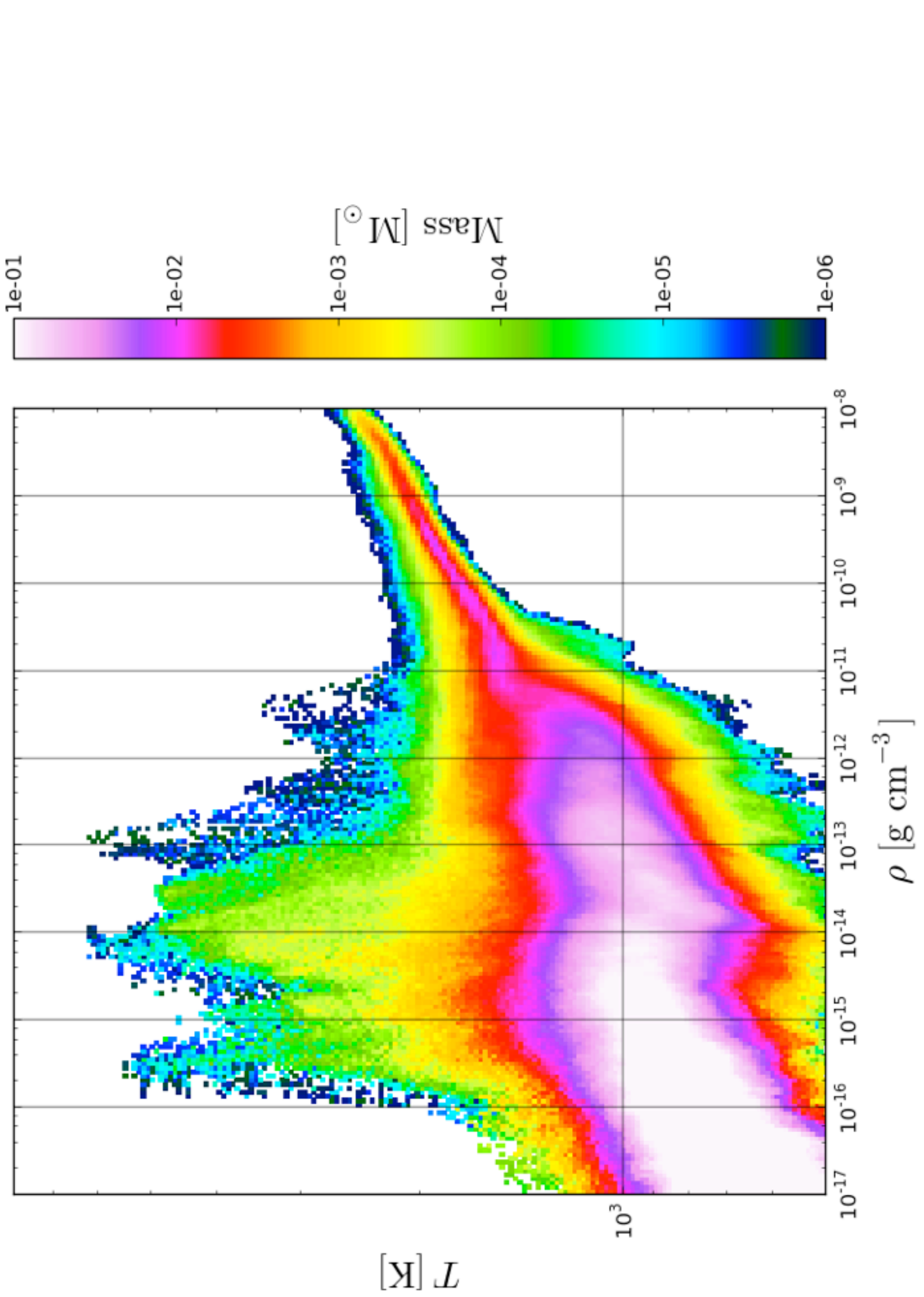
AU

1e-07 1e-06 1e-05 1e-04 1e-03 1e-02 1e-01 1

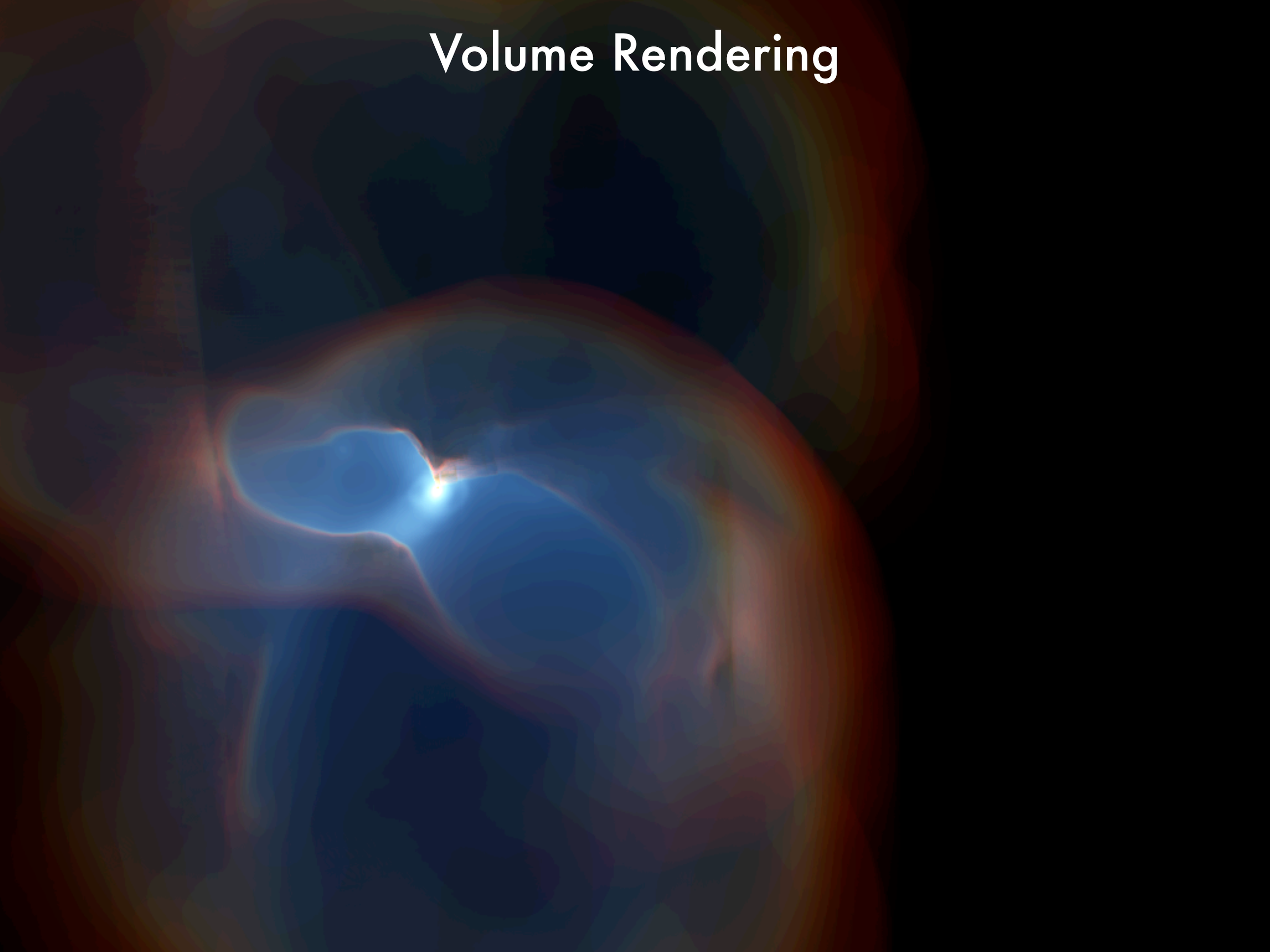
Mass [$M_{\odot}$]

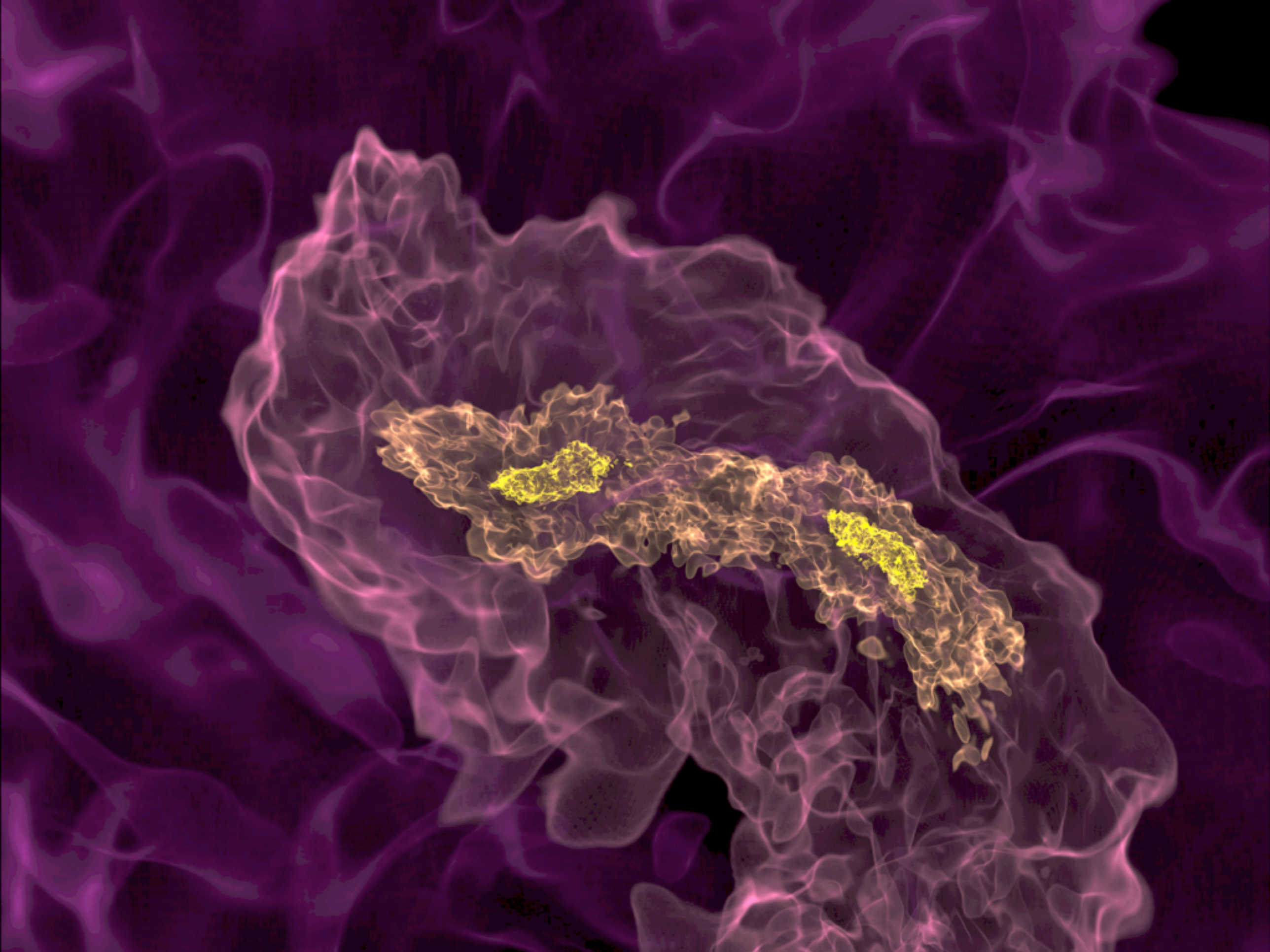
1e-07 1e-06 1e-05 1e-04 1e-03 1e-02 1e-01

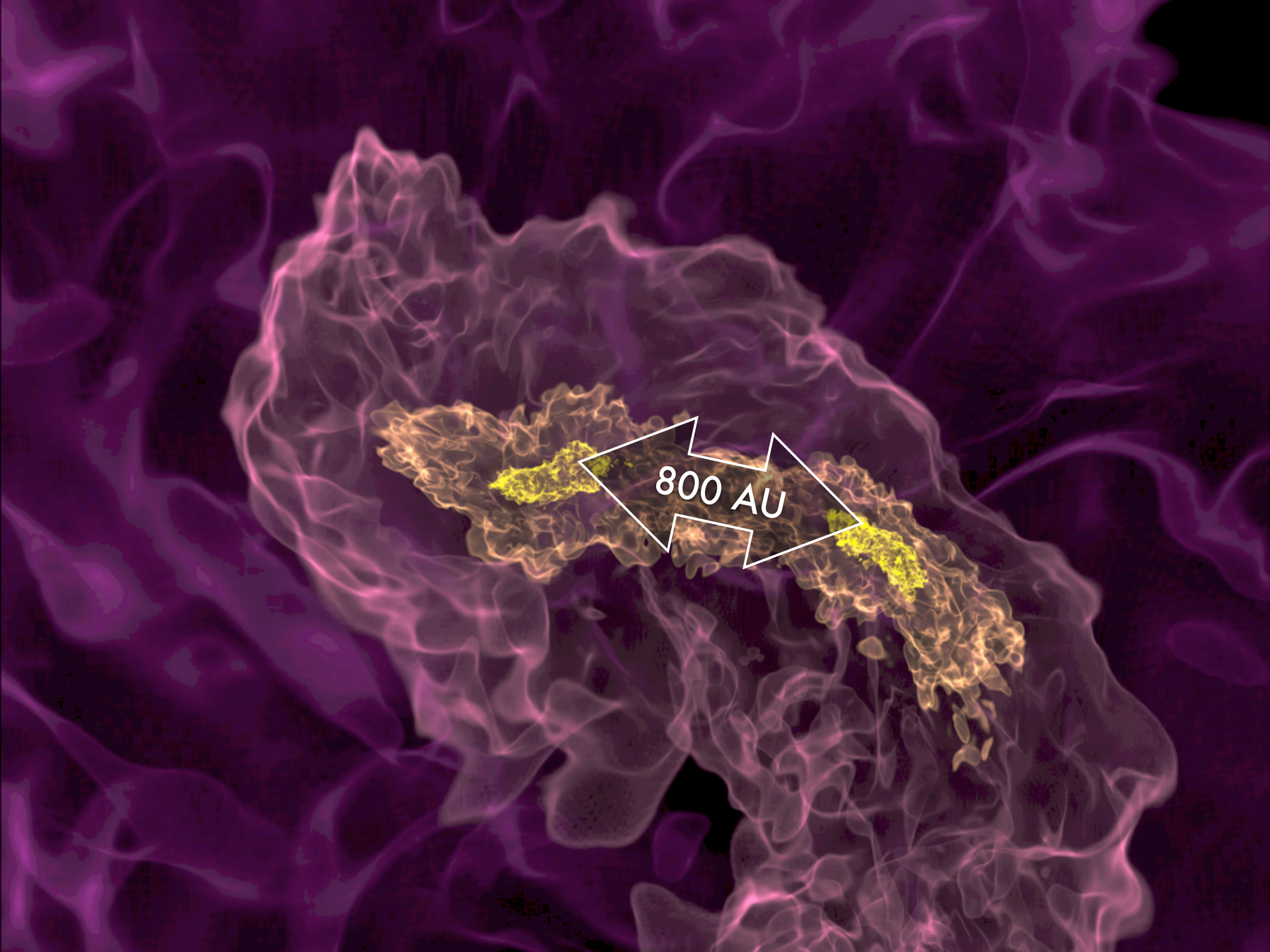
H_2/ρ

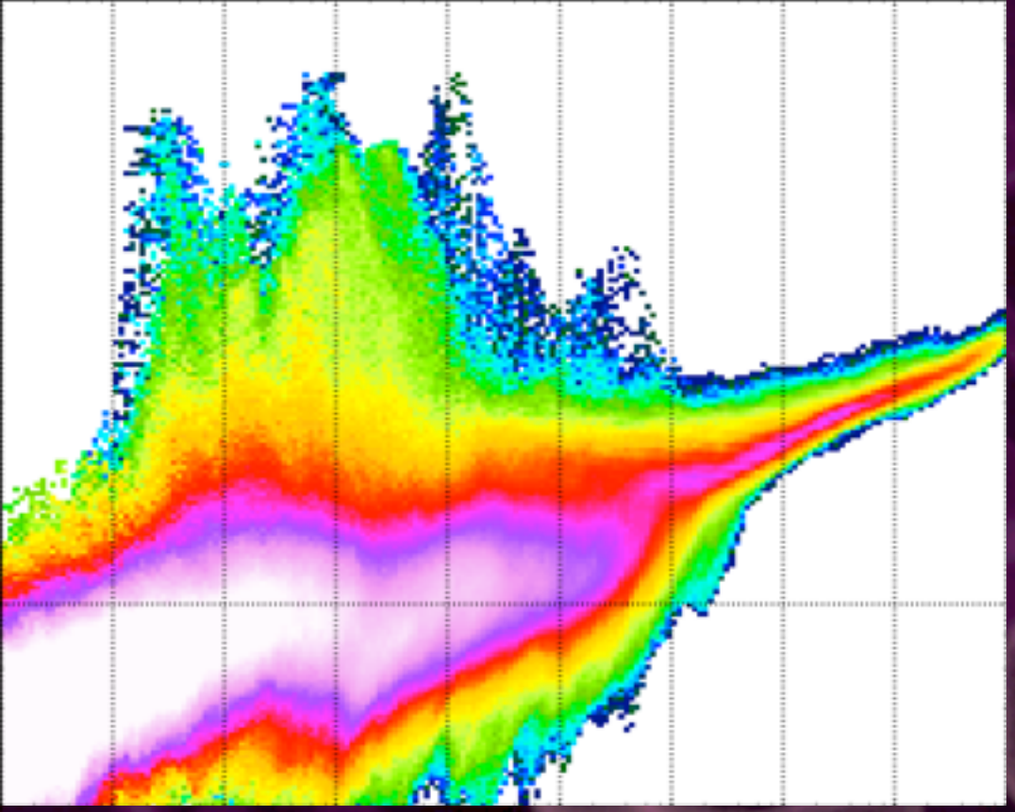

Mass [M$_\odot$]
1e-01
1e-02
1e-03
1e-04
1e-05
1e-06
T [K]
10^3
ρ [g cm^{-3}]
10^{-8}
10^{-9}
10^{-10}
10^{-11}
10^{-12}
10^{-13}
10^{-14}
10^{-15}
10^{-16}
10^{-17}

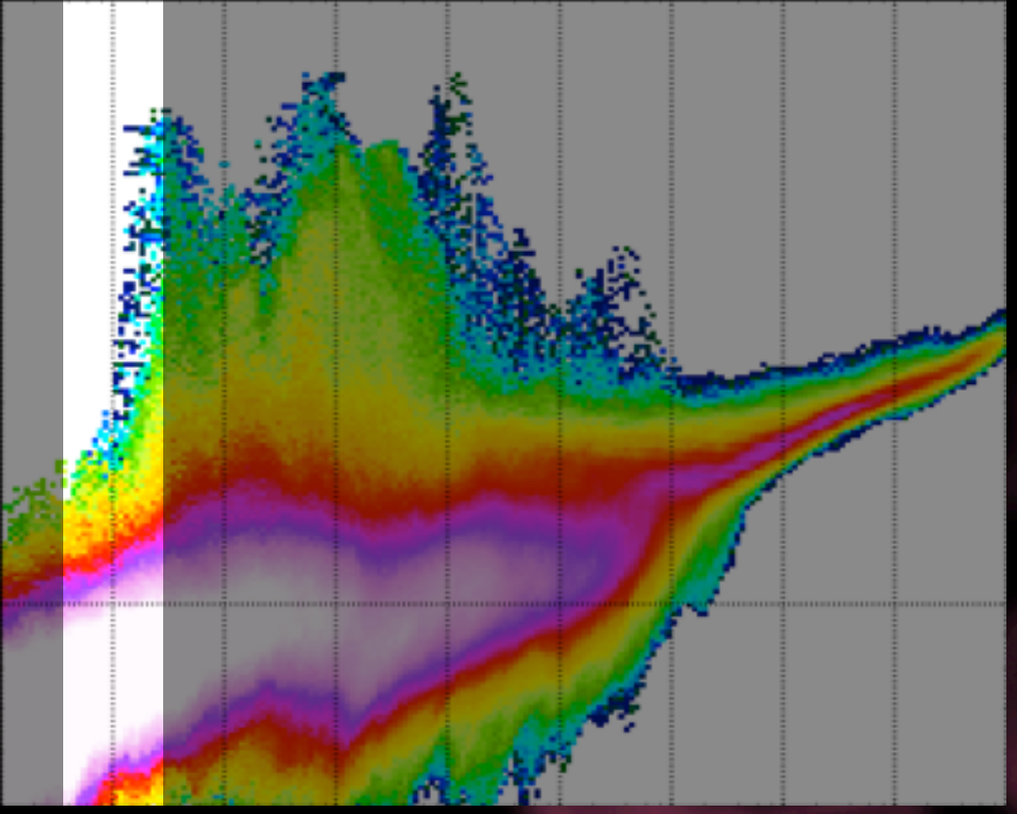
Volume Rendering



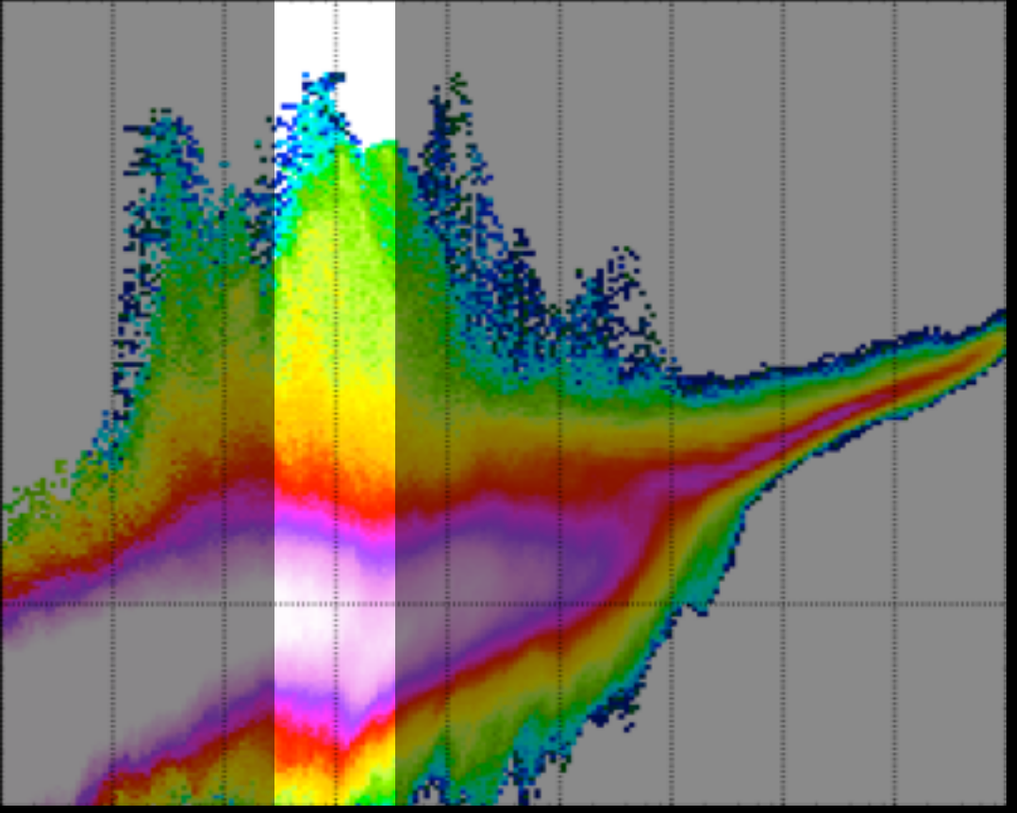



800 AU

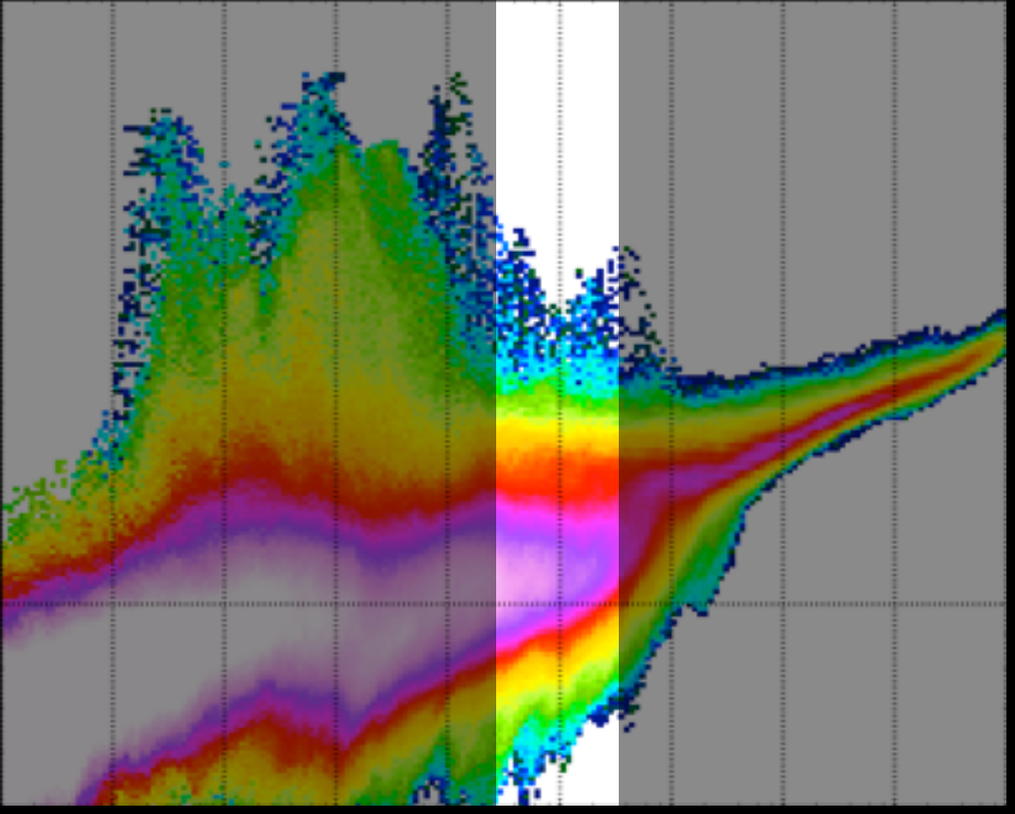


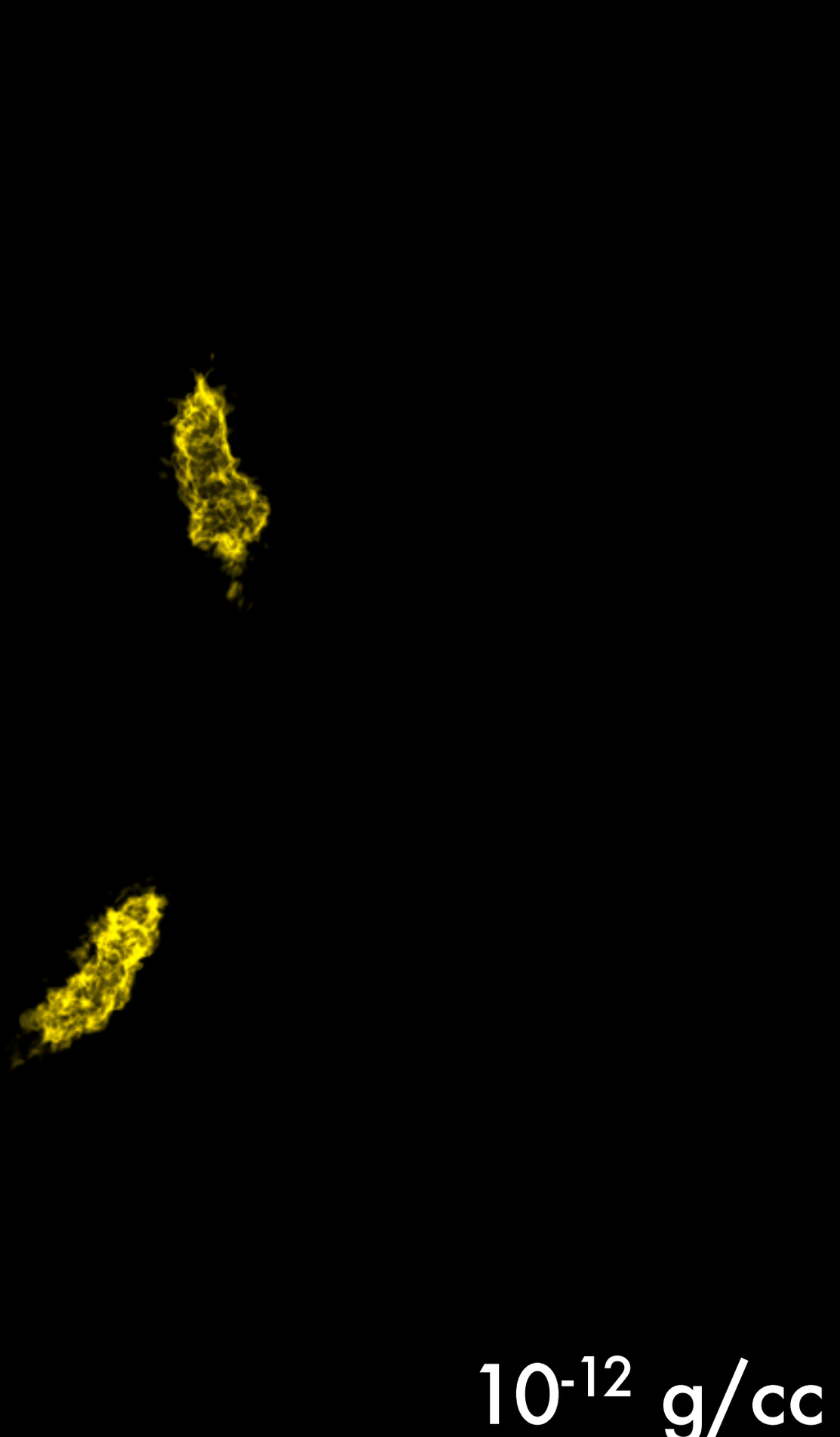


10^{-16} g/cc

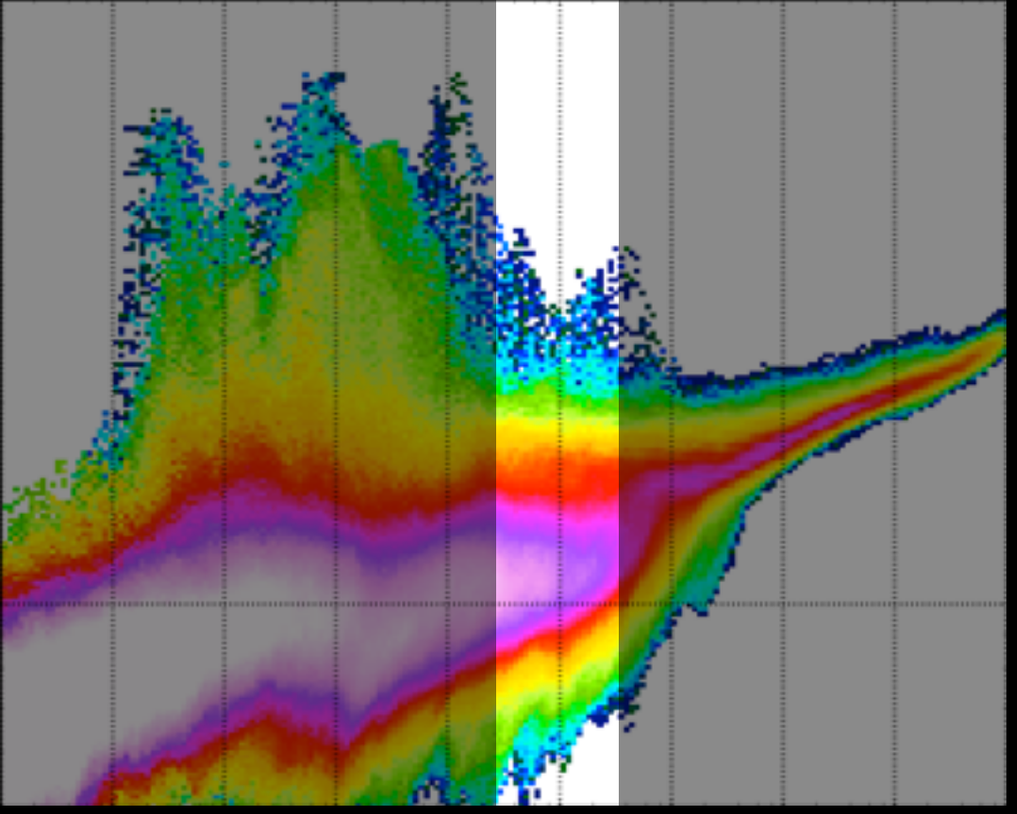


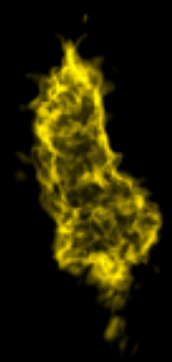
10^{-14} g/cc



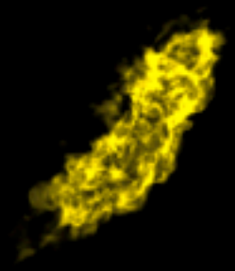


10^{-12} g/cc





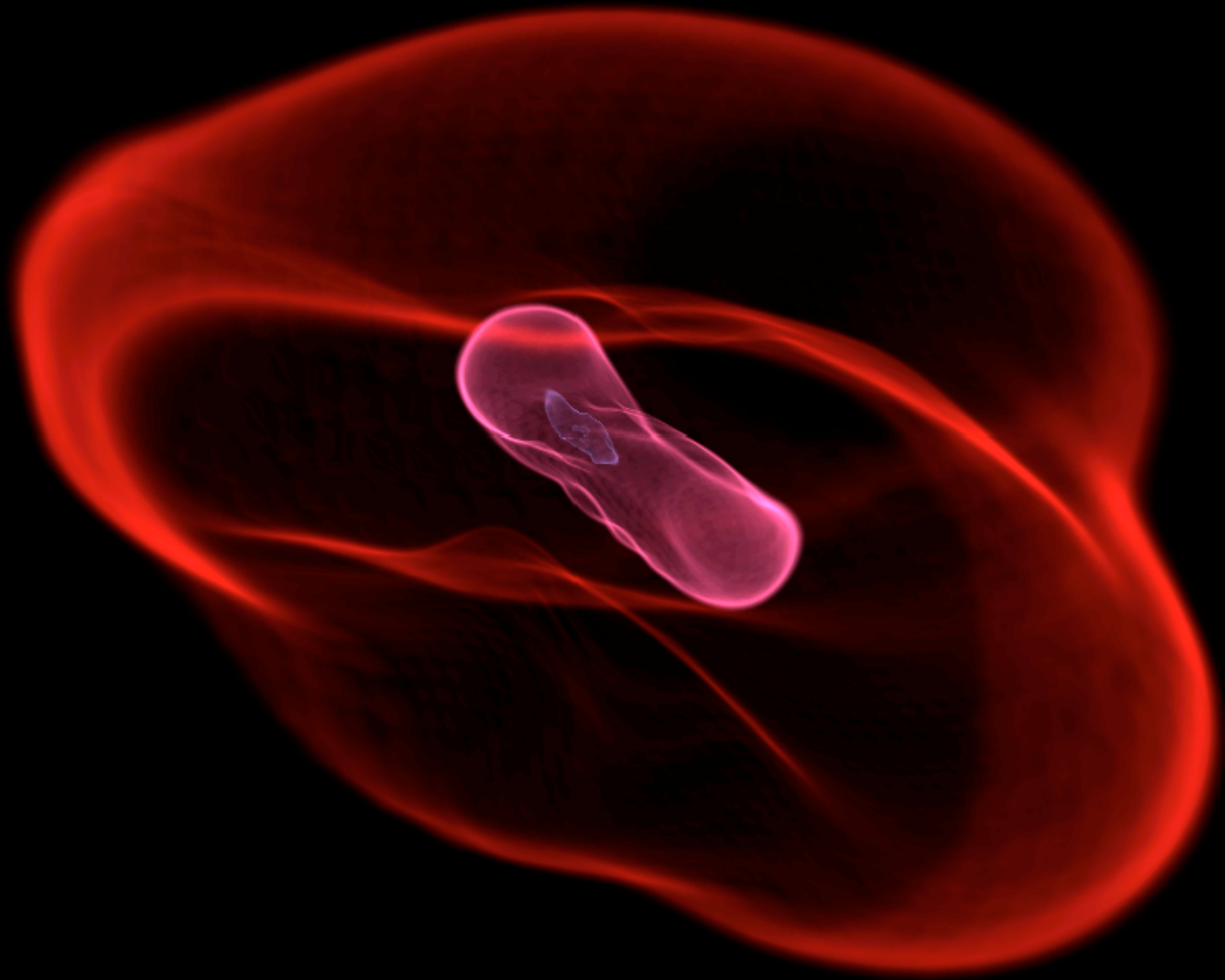
6.3 M⊙



10 M⊙

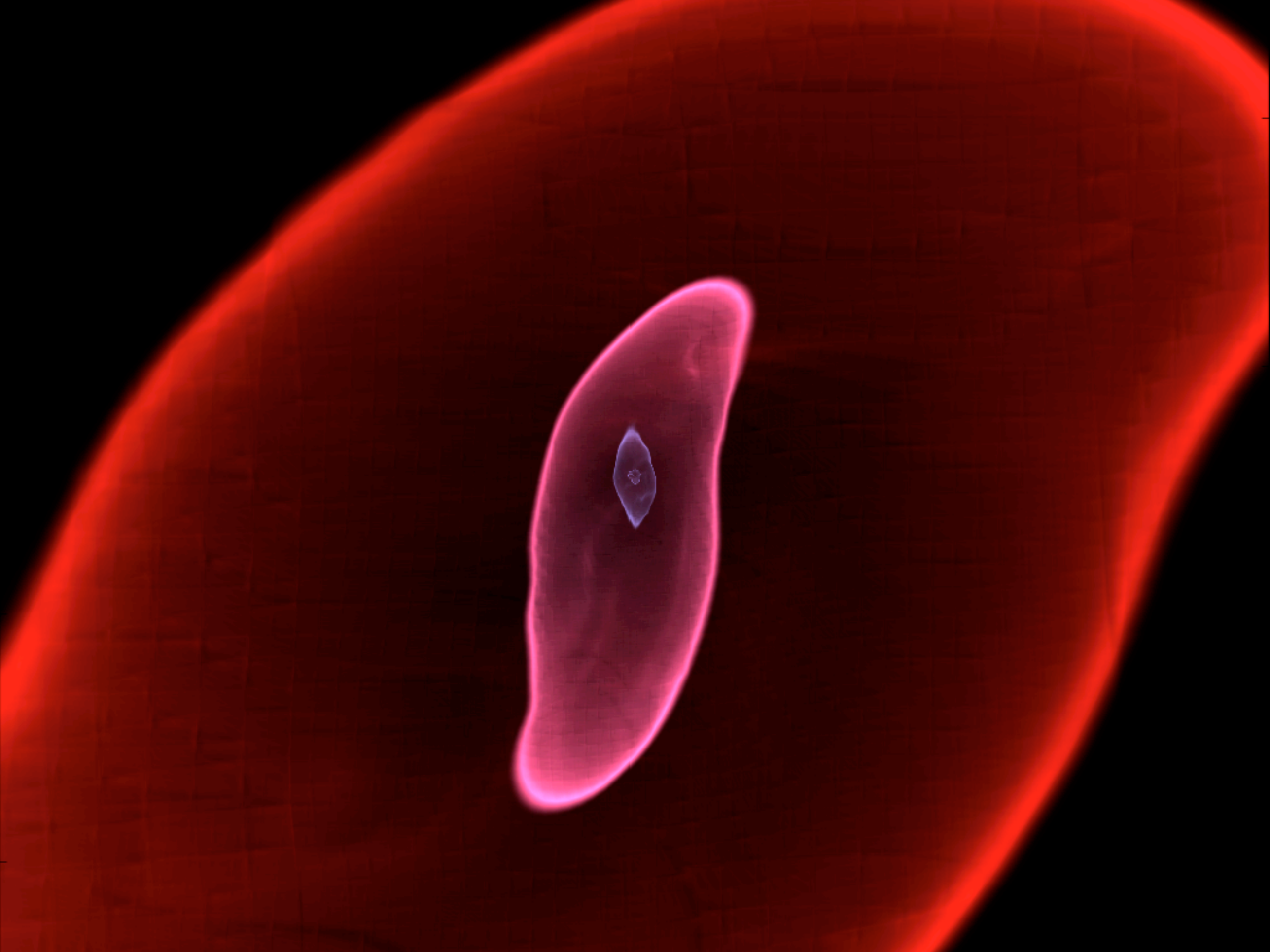
10^{-12} g/cc

```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
v, c = pf.h.find_max("Density")
L = [1.0, 1.0, 1.0]
W = 1000.0/pf['au']
tf = vr.ColorTransferFunction((-14.0, -4))
tf.add_layers(8)
cam = pf.h.camera(c, L, W, 1024, tf)
cam.snapshot()
```

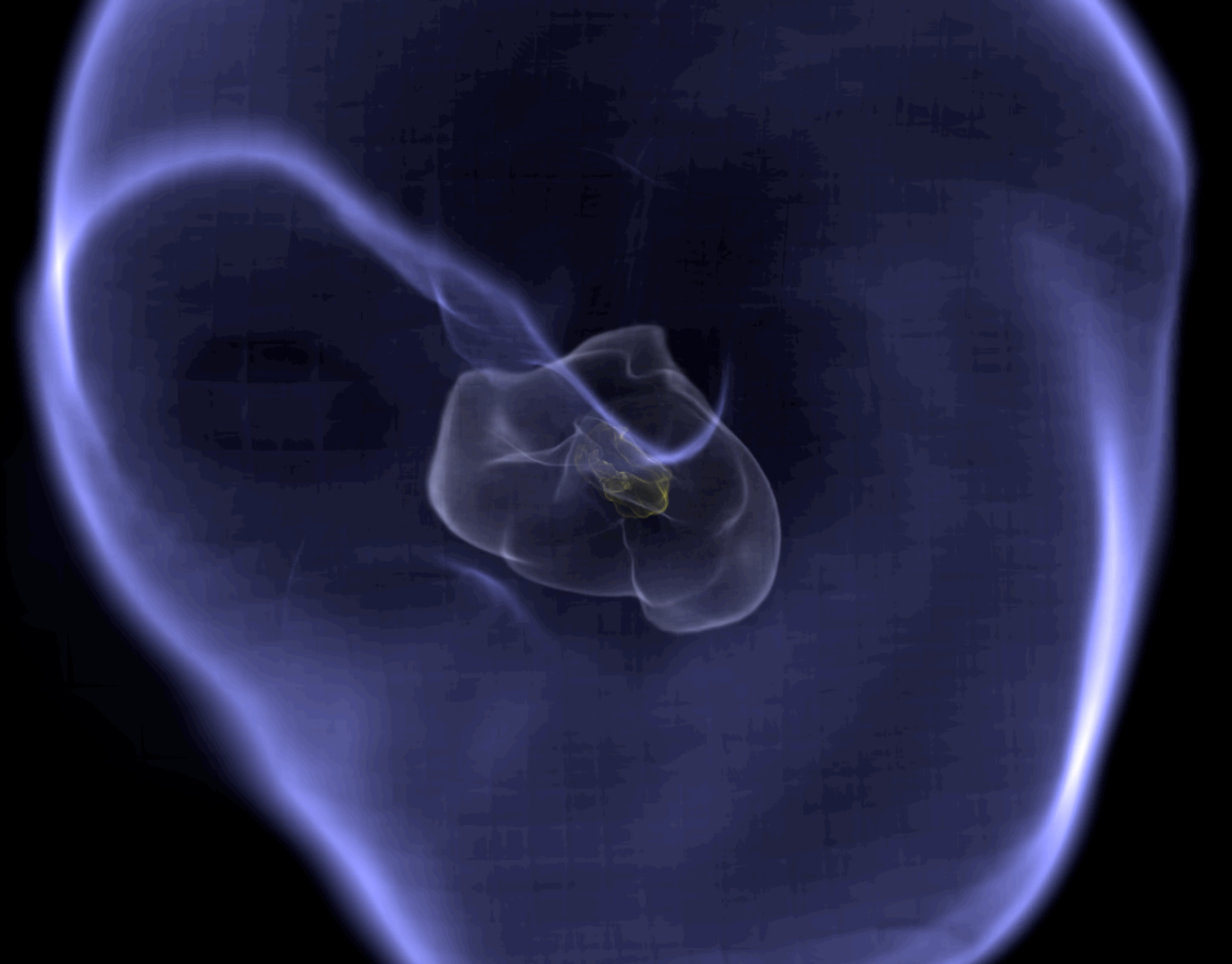


```
from yt.mods import *
pf = load("DataDump0155.dir/DataDump0155")
v, c = pf.h.find_max("Density")
sp = pf.h.sphere(c, 1000.0/pf['au'])
L = sp.quantities["AngularMomentumVector"]()
W = 1000.0/pf['au']
```

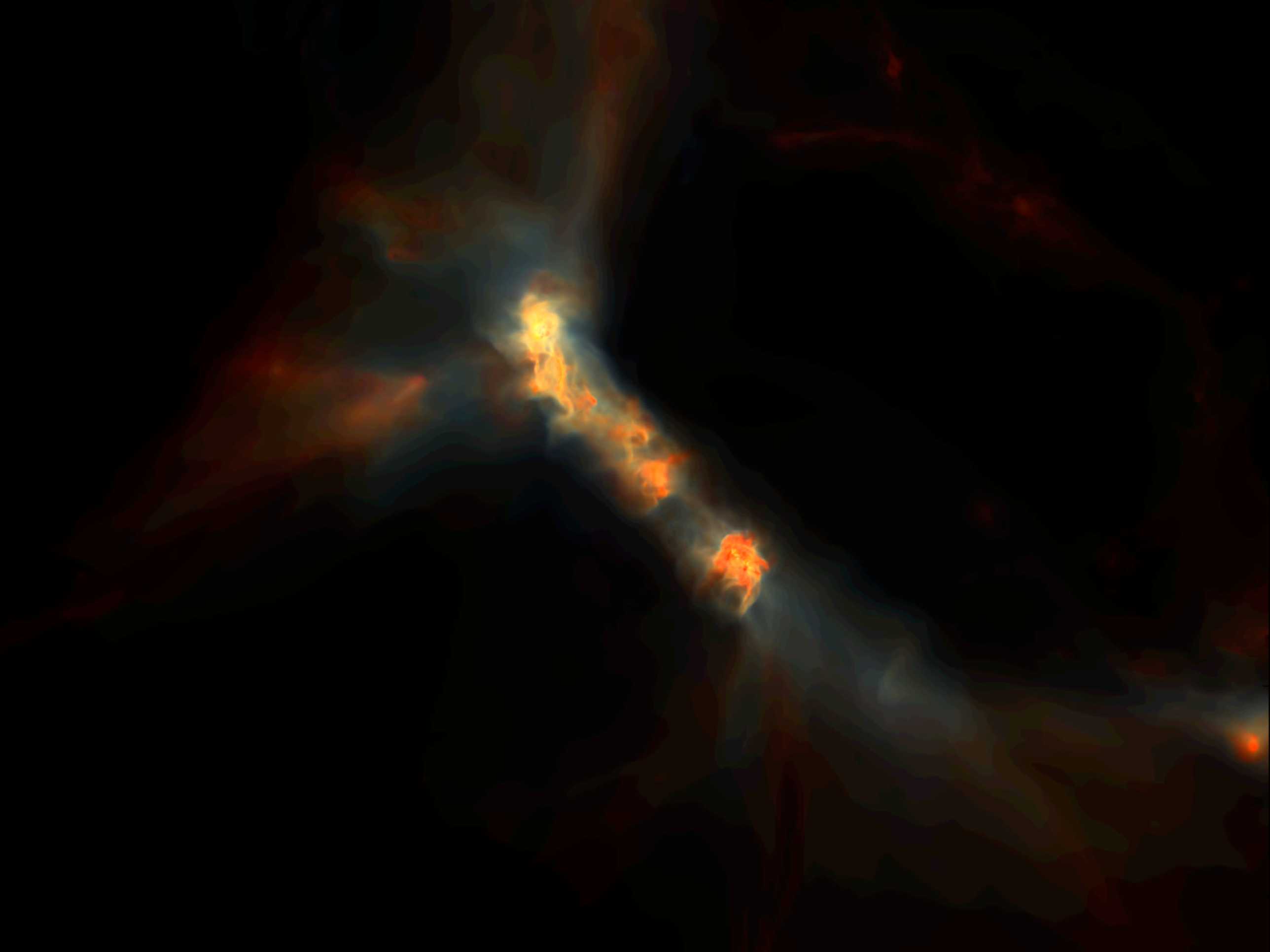
```
tf = vr.ColorTransferFunction((-14.0, -4))
tf.add_layers(8)
cam = pf.h.camera(c, L, W, 1024, tf)
cam.snapshot()
```

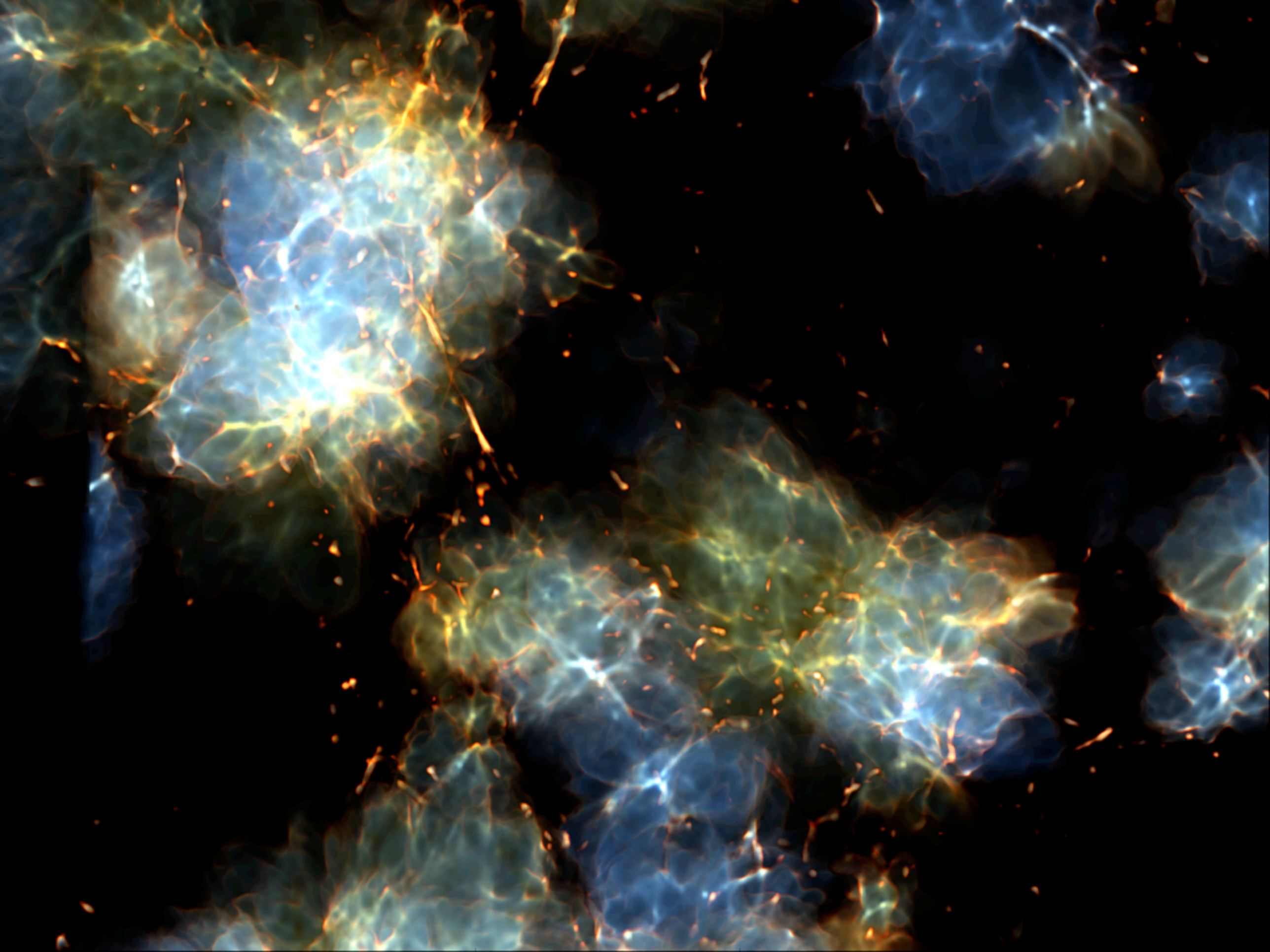


```
cam.zoom(100.0)
cam.snapshot()
```

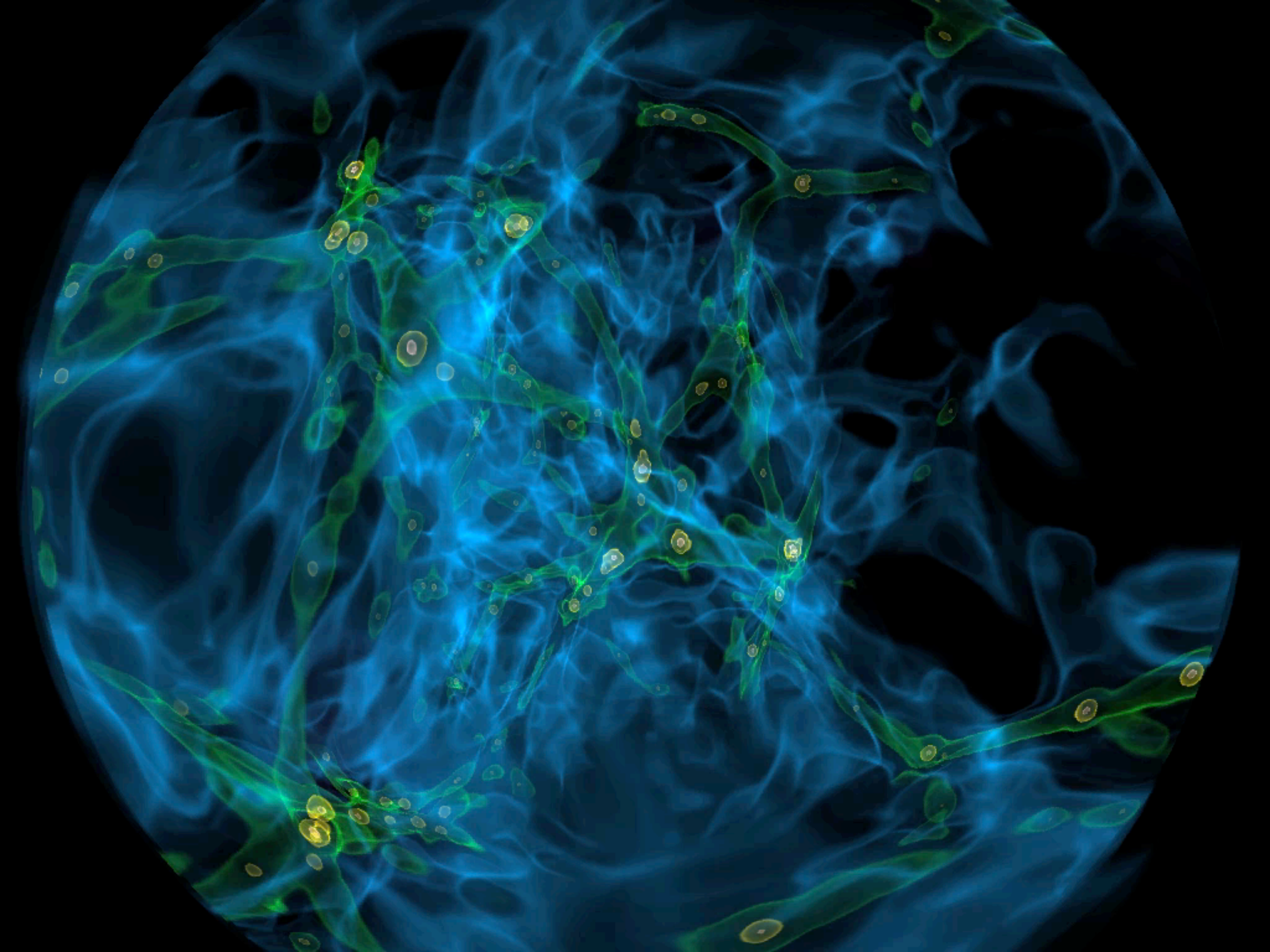


```
from yt.mods import *
pf = load(“DD1701/DD1701”)
v, c = pf.h.find_max(”Density”)
L = [1.0, 1.0, 1.0]
W = 100.0/pf['mpc']

tf = vr.PlanckTransferFunction()
cam = pf.h.camera(c, L, W, 1024, tf)
cam.snapshot()
```

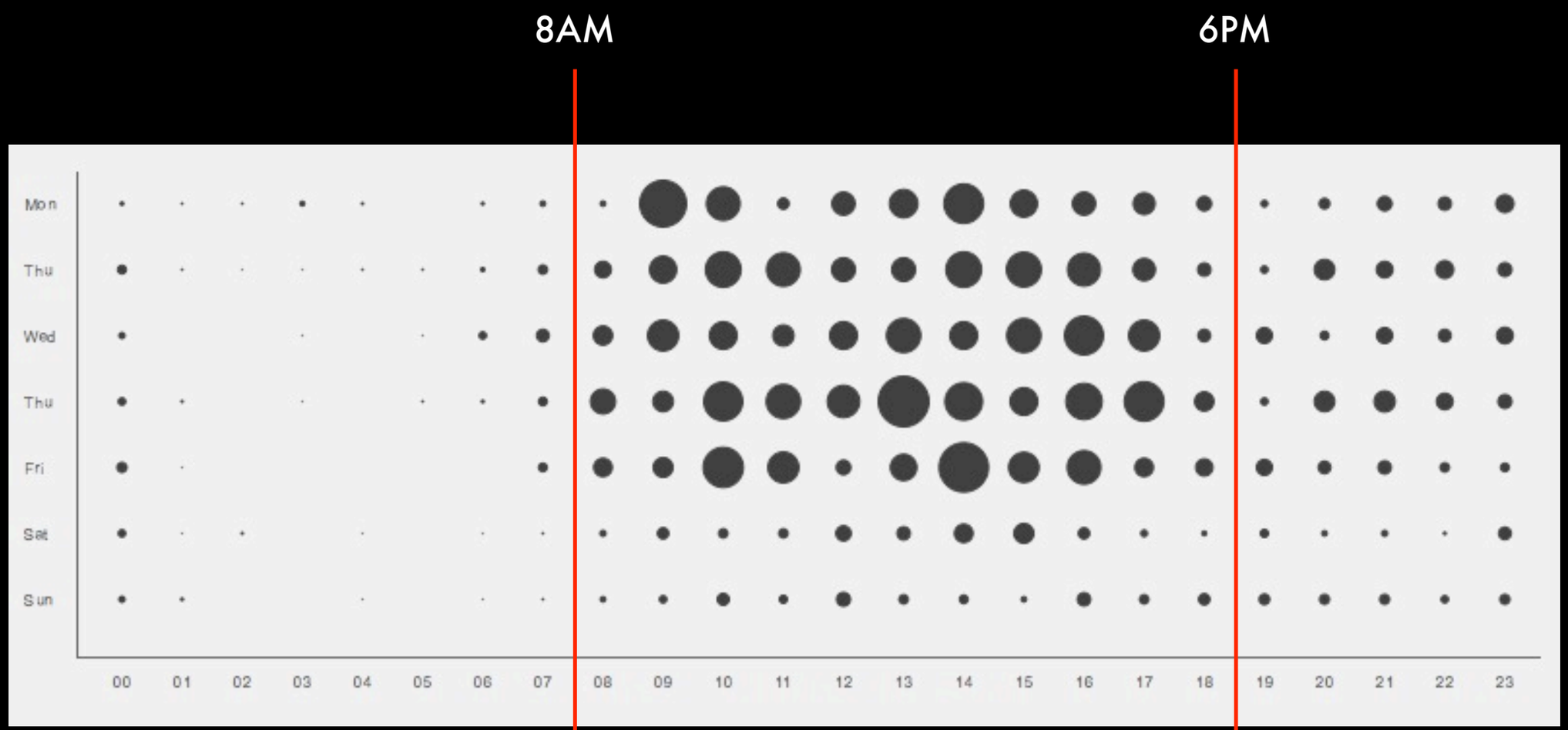




Off-axis Projection



71,000 lines of code
Python, Cython, C
12 contributors (60+ users)
Contributors from 7 institutions



What's next?

initialization


better astrophysical
object selection

co-scheduled & in situ viz

(hooking up to the hydrant)

Thank you.

Tom Abel
David Collins
Oliver Hahn
Cameron Hummels
Ji-hoon Kim
Christopher Moody
Michael Norman
Brian O'Shea
Stella Offner
Jeff Oishi
Devin Silvia
Sam Skillman
Stephen Skory
Britton Smith
Matthew Turk
John Wise
John ZuHone

yt.enzotools.org