08_Volume_Rendering_in_yt

September 26, 2014

1 A Brief Demo of Volume Rendering

This shows a small amount of volume rendering. Really, just enough to get your feet wet!

```
In []: import yt
       ds = yt.load("IsolatedGalaxy/galaxy0030/galaxy0030")
```

To create a volume rendering, we need a camera and a transfer function. We'll use the `ColorTransferFunction`, which accepts (in log space) the minimum and maximum bounds of our transfer function. This means behavior for data outside these values is undefined.

We then add on "layers" like an onion. This function can accept a width (here specified) in data units, and also a color map. Here we add on four layers.

Finally, we create a camera. The focal point is `[0.5, 0.5, 0.5]`, the width is 20 kpc (including front-to-back integration) and we specify a transfer function. Once we've done that, we call `show` to actually cast our rays and display them inline.

```
In []: tf = yt.ColorTransferFunction((-28, -24))
       tf.add_layers(4, w=0.01)
       cam = ds.camera([0.5, 0.5, 0.5], [1.0, 1.0, 1.0], (20, 'kpc'), 512, tf, fields=["density"])
       cam.show()
```

If we want to apply a clipping, we can specify the `clip_ratio`. This will clip the upper bounds to this value times the standard deviation of the values in the image array.

```
In []: cam.show(clip_ratio=4)
```

There are several other options we can specify. Note that here we have turned on the use of ghost zones, shortened the data interval for the transfer function, and widened our gaussian layers.

```
In []: tf = yt.ColorTransferFunction((-28, -25))
       tf.add_layers(4, w=0.03)
       cam = ds.camera([0.5, 0.5, 0.5], [1.0, 1.0, 1.0], (20.0, 'kpc'), 512, tf, no_ghost=False)
       cam.show(clip_ratio=4.0)
```