yt graphics interface for MOCASSIN

K.Gesicki & K.Kowalik

Centre for Astronomy, Faculty of Physics, Astronomy and Informatics
Nicolaus Copernicus University, ul. Grudziadzka 5, 87-100 Torun, Poland
Krzysztof.Gesicki@astri.umk.pl Kacper.Kowalik@astri.umk.pl

http://yt-project.org/

from the web-page of the yt-project you can read that:

yt is a python package for analyzing and visualizing volumetric, multi-resolution data from astrophysical simulations, radio telescopes, and a burgeoning interdisciplinary community.

yt is more than a visualization package: it is a tool to seamlessly handle simulation output files to make analysis simple. **yt** can easily knit together volumetric data to investigate phase-space distributions, averages, line integrals, streamline queries, region selection, halo finding, contour identification, surface extraction and more.

yt is built on a stack of completely free and libre open source software, with no proprietary dependencies. Getting **yt** is as simple as running the installation script.

yt aims to provide a simple uniform way of handling volumetric data, regardless of where it is generated. **yt** currently supports FLASH, Enzo, Boxlib, Athena, arbitrary volumes, and support is in progress for Gadget, Tipsy, ART, RAMSES and MOAB. If your data isn't already supported, why not add it?

http://mocassin.world-traveller.org/

from the mocassin's web-page you can read that:

MOCASSIN is a fully 3D or 2D photoionisation and dust radiative transfer code which employs a Monte Carlo approach to the transfer of radiation through media of arbitrary geometry and density distribution.

The code can deal with arbitrary Cartesian grids of variable resolution, it has successfully been used to model complex density fields from SPH calculations and can deal with ionising radiation extending from Lyman edge to the X-ray. The dust and gas microphysics is fully coupled both in the radiation transfer and in the thermal balance.

The mocassinPlot driver produces an output file, containing the luminosities of each individual grid volume element in the required emission lines. This file, named plot.out, is written in a format which should be easily readable into a data visualisation software, such as IDL or PDL.

We asked ourselves: why not to use **yt** for 3D visualisation?

combining mocassin and yt

For the visualisation of results of **mocassin** we persuaded **yt** to read the output files. This has been accomplished in a form of a python script. Until **mocassin** will be fully supported by **yt** the input script should be run separately. The script is available upon request from the authors of this contribution.

Right now the input for **yt** requires a uniform grid, therefore all features of **mocassin** cannot be used, however the work on applying a variable resolution grid is in progress.

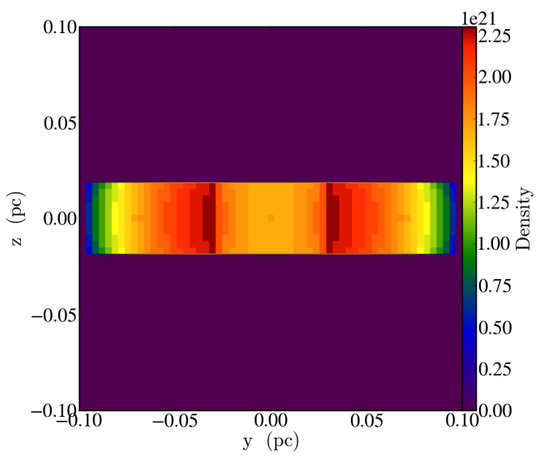
After parsing **mocassin** output data (files: grid0.out, grid1.out, plot.out) the user has access to a plethora of commands and routines written in *python* by **yt** team. The web page offers a lot of *quick links*, clear *quickstart guide* and extended documentation for users with different experience and requirements. We appreciated very much the special care about inexperienced users.

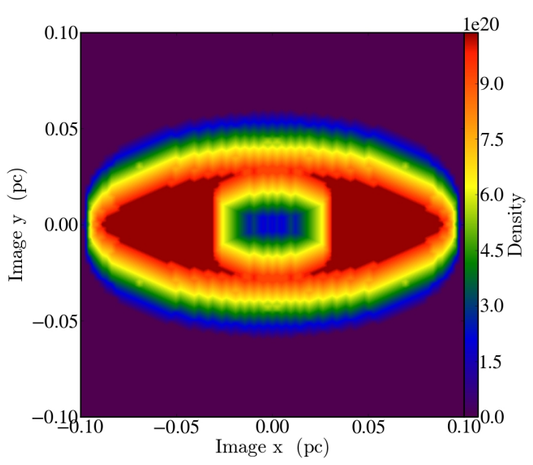
We present here the results of our first tests. These are not real nebulae models but a flat disk with uniform density, created for the purpose of learning new tool on a simple example.

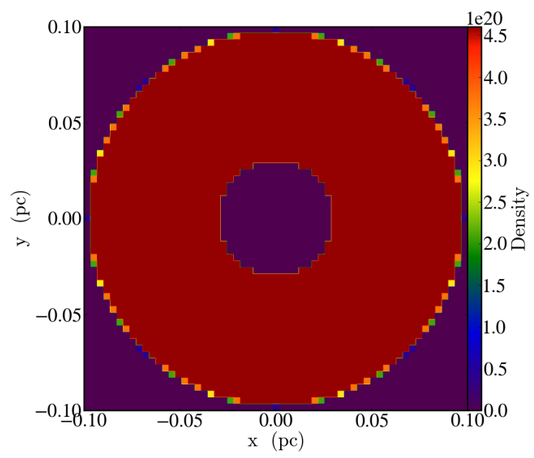
At right we show how it works: this simple python script reads our *load_mocassin* command, loads **mocassin** data and produces the images with *OffAxisProjectionPlot* command of **yt**.

Below we show these images presenting the disk at different angles and in different wavelengths.

```
*foo.py
#!/usr/bin/python
import matplotlib
matplotlib.rc("axes", linewidth=2.0)
matplotlib.rc("lines", linewidth=2.0)
matplotlib.rc("font", size=36)

from yt.mods import OffAxisProjectionPlot, ProjectionPlot
from parse_mocassin import load_mocassin

pf = load_mocassin('/home/gesicki/APN6poster/')
fields = ["Emission_Line_3", "Emission_Line_5", "Emission_Line_7"]
for field in fields:
    prj = OffAxisProjectionPlot(pf, [1.0, 1.0, 0.7], field,
                                center=pf.domain_center, width=(0.2, 'pc'),
                                depth=(0.5, 'pc'), north_vector=[0.0, 0.0, 1.0],
                                fontsize=28)
    prj.save()

field = "Density"
ProjectionPlot(pf, 'z', field, center=pf.domain_center,
               width=(0.2, 'pc'), fontsize=28).save()
OffAxisProjectionPlot(
    pf, [1.0, 1.0, 0.7], field,
    center=pf.domain_center, width=(0.2, 'pc'),
    depth=(0.5, 'pc'), north_vector=[0.0, 0.0, 1.0], fontsize=28
).save()
ProjectionPlot(pf, 'x', field, center=pf.domain_center,
               width=(0.2, 'pc'), fontsize=28).save()
Python  Szerokość tabulacji: 6  Wrsz 28, kol 46  WST
```

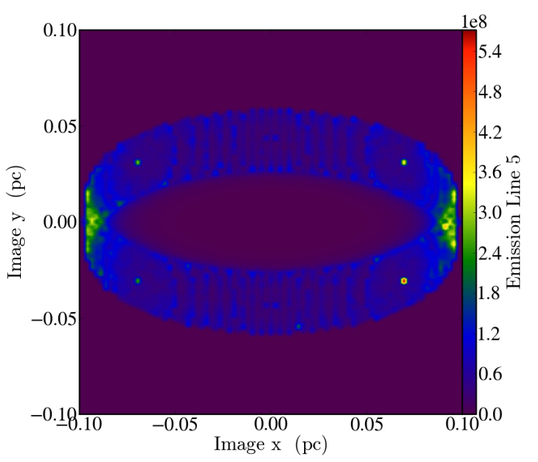


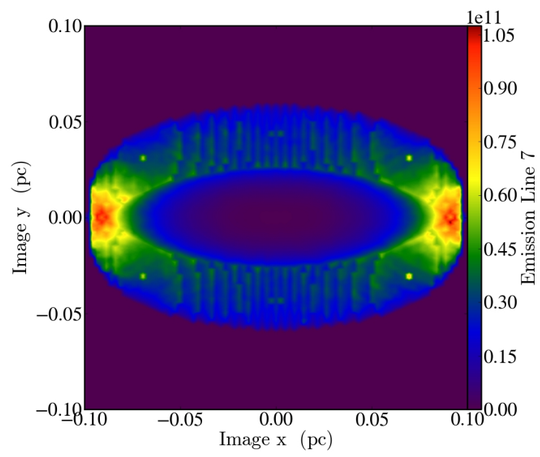


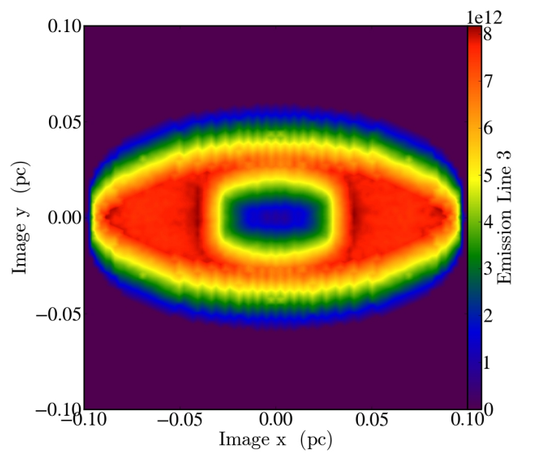


example of a simple flat disk
with uniform density, defined on
spatial grid 60x60x60

here the density structure
is plotted for the disk
seen at different angles







example of a simple flat disk,
with uniform density of 4000 cm^{-3},
ionized by the central black body
with temperature of 75 000 K

presented are (from left)
the monochromatic images at lines
[O I] 6300, [N II] 6583, [O III] 5007

Acknowledgements:
This publication was supported by the Polish National Science Centre (NCN) through the grant 2011/03/B/ST9/02552