Astronomical Data Analysis Software and Systems XXI
ASP Conference Series, Vol. 1000
Pascal Ballester, Daniel Egret, and Nuria P. F. Lorente, eds.
©2012 Astronomical Society of the Pacific

Glnemo2, an Interactive 3D Visualization Program for N-body Data

J. C. Lambert

Laboratoire Astrophysique de Marseille, CNRS UMR6110, Université de Provence, France

Abstract. Glnemo2 aims at giving a new user experience to visualize the results of N-body simulations and a new approach in the analysis process. It is particularly timely, because astrophysicists have now access to a lot of computing power to run N-body simulations from single multi-processor machines up to small clusters.

1. Introduction

Glnemo2 is an interactive 3D visualization program which displays particle positions of the different components (gas, stars, disk, dark mater halo, bulge) of an N-body snapshot. It's a very useful tool for everybody running N-body simulations from isolated galaxies to cosmological simulations. It can show quickly a lot of information about data by revealing shapes, dense areas, formation of structures such as spirals arms, bars, peanuts or clumps of galaxies.

Glnemo2 has been designed to meet the requirements of the user, with simplicity in mind, easy to install, easy to use with an interactive and responsive graphical user interface, powerful with a fast 3D engine and generic with the possibility to load different kinds of input files. Glnemo2 is part of the NEMO project[1] from where it gets its philosophy and the way it works from the command line.

Glnemo2 has a growing worldwide user community who daily use it for checking the evolution of galaxy simulations, to produce images which illustrate scientific articles or to make scientific movies which explain some results. It's also used for teaching N-body simulations in some universities in Barcelona, Strasbourg and Tokyo.

2. Graphical User Interface

Glnemo2 uses an intuitive and interactive graphical user interface (GUI) based on Nokia's QT4 API,[2] which intensively uses widgets and dialog boxes. Many operations can be applied from the GUI such as 3D view rotation around one of the X,Y or Z axes, zooming operations to move quickly inside the data, counting particles within a rectangle selected with the mouse.

[1]NEMO project home page: `http://carma.astro.umd.edu/nemo`

[2]QT project home page: `http://qt.nokia.com`

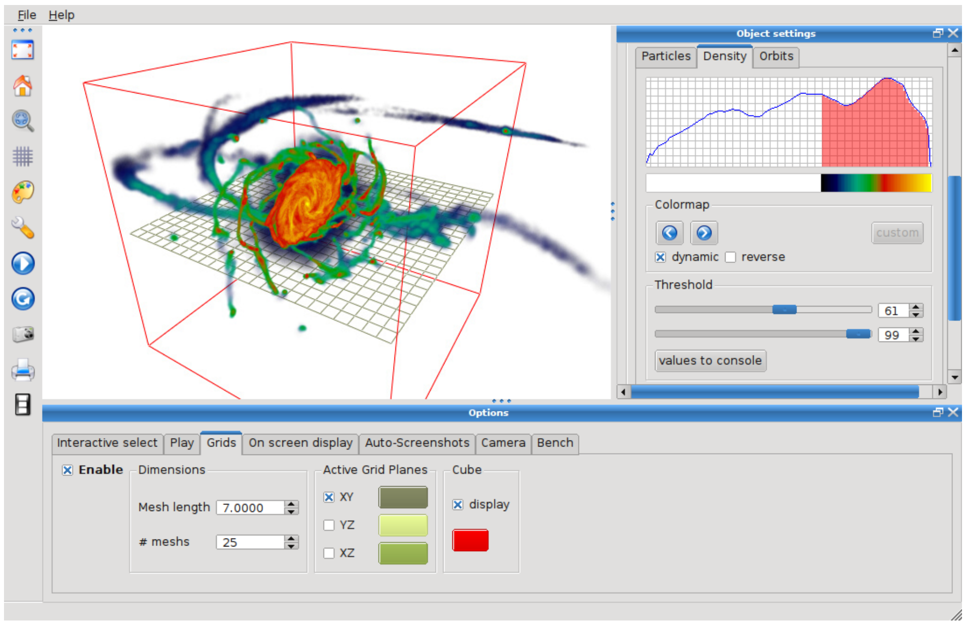
Each component of a simulation (disk, gas, stars, dark matter halo, bulge) can be loaded selectively and are identified as new objects from glnemo2. Each of these objects can be individually modified and displayed. Object properties like size and color of the particles and velocity vectors can be interactively modified.

You can follow the evolution of a simulation by just clicking on the play button of the interface. QT4 API is multi-threaded and some parts of the glnemo2 code, like data-loading, run also in a parallel thread, thus making the GUI always responsive even during heavy loading.

You can trace orbits of selected particles over time to follow their history.

You can easily create movies from a dedicated interface which records every user event (keyboard, mouse) and saves each rendered frame on disk.

There is a dedicated interface for components providing density fields, such as gas particles from GADGET[3] and RAMSES[4] simulations. From this interface the density threshold can be modified in real time (as seen on the right part of Figure 1) which dramatically reveals the shape of the selected component.



Threshold: 61, 99

[✓ dynamic] reverse

[✓ Enable]

Mesh length 7.0000

meshs 25

[✓ XY] YZ XZ

[✓ display]

Figure 1. Glnemo2 Graphical User Interface

3. 3D Engine

Glnemo2 uses a powerful 3D engine which takes into account the power of the graphics card to accelerate the rendering. Millions of particles can be rendered in real time. The 3D engine is multi-threaded to always ensure a responsive GUI. It uses the latest

[3]GADGET project: `http://www.mpa-garching.mpg.de/gadget/`

[4]RAMSES project: `http://web.me.com/romain.teyssier/Site/RAMSES.html`

OpenGL technologies such as vertex buffer objects (VBO[5]) to use directly the GPU's main memory, frame buffer objects (FBO[6]) to render off-screen screen-shots to create movies, and GLSL[7] shader to use the multi-core power of the graphic card. GLSL shaders are dedicated programs which run on top of the GPU and in parallel on every core of the graphics card. Basically the following algorithm is used: a Gaussian texture is applied around each particle. The size of the texture depends on the size of the nearest neighbor of the current particle. Color and transparency are coded according to the local density value of the particle.

4. Generic Data Loading

Glnemo2 uses a plug-in mechanism to load data in a generic way, based on an object-oriented virtual class. Adding a new input file format is just a matter of writing a new plug-in following the specifications of the main virtual loading class. This makes the application really versatile about what type of files can be read. The loading engine detects automatically the input file format and all the components of the snapshot. Disk, gas, dark matter halo, bulge and stars can be loaded and displayed individually.
Here is a list of input file formats natively supported by the application:

- Nemo files from the Nemo project,[1] maintained by Peter Teuben
- Gadget[2] 1, 2, 3, little-endian and big-endian from Volker Springel's simulation program
- Ramses[3] data, from Romain Teyssier's simulation program
- FTM files from Clayton Heller's SPH/N-body code
- PhiGrape[8] simulation files
- List of snapshot files stored in a file to watch the evolution of a simulation
- Walter Dehnen's gyrfalcON[9] N-body simulation program coupled with a network plug-in

5. Multiple-platform

The application is globally written in the C++ language using intensively object programming paradigm. The library OpenGL is used to manage the 3D operations and QT4 API is used for the implementation of the graphical user interface. These three technologies have been chosen for their portability across architectures, thus making

[5]`http://www.opengl.org/wiki/Vertex\_Buffer\_Object`

[6]`http://www.opengl.org/wiki/Framebuffer\_Object`

[7]`http://www.opengl.org/documentation/glsl/`

[8]PHIGRAPE project: `http://www-astro.physik.tu-berlin.de/~harfst/index.php?pid=8`

[9]GyrfalcON article: `http://iopscience.iop.org/1538-4357/536/1/L39/pdf/005197.web.pdf`

glnemo2 fully multi-platform. Glnemo2 runs on the three major operating systems, on Linux, Mac OS X and Windows, with no modification of the code, and with the same behavior.

Glnemo2 has a dedicated website[10] provided by the CeSAM[11] where you can download the code, get some useful information about how to compile and install the code, and get some basic help and tutorials.

Glnemo2 is an open-source application under the term of the license CeCILL[12].

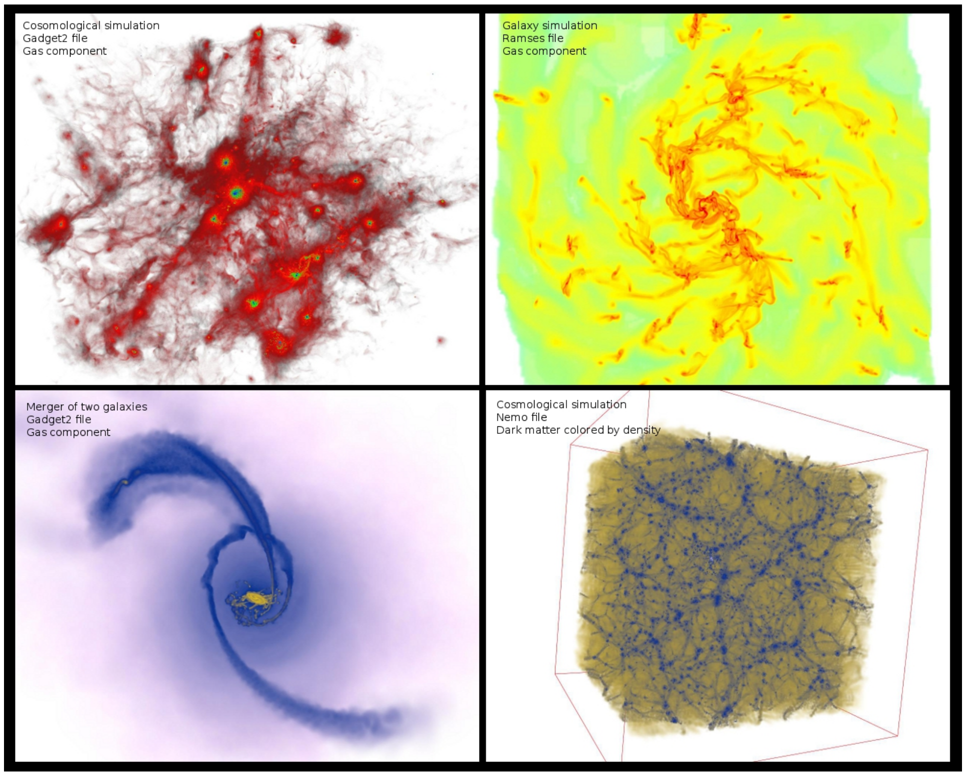


Figure 2. Glnemo2 rendering of different simulations colored by density

Acknowledgments. Thanks to Lia Athanassoula for her support and advice. Thanks to Sergey Rodionov, Emmanuel Nezri, Ingo Berentzen, Rubens Machado for their tests and bug reports. Special thanks to Peter Teuben for his skills and advice and for supporting me for so many years.

[10]Glnemo2 home page: `http://projets.oamp.fr/projects/glnemo2`

[11]Centre de données Astrophysiques de Marseille (`http://lam.oamp.fr/cesam`)

[12]CeCILL: `http://www.cecill.info/licences/Licence\_CeCILL\_V2-en.html`