Galaxia

1.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

Galaxia

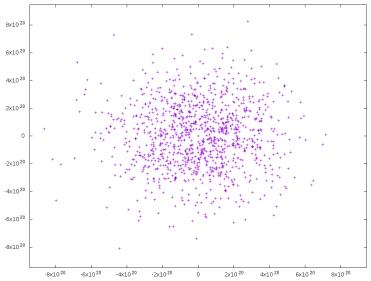

Figure 1.1 Simulating a galaxy with Galaxia

We developed this program to calculate the movement of stars in galaxy collisions.
By changing the intial conditions it can be used to tackle about any N-Body problem in classical astrophysics.

1.1 Prerequisites

You will need at least gnuplot, ffmpeg for usage. g++ and cmake are required for compiling. Boost is required for compiling the code and Doxygen is used for documentation. If you are using a Debian based System like Ubuntu, you can install most of these by typing `sudo apt-get install gnuplot ffmpeg doxygen g++ cmake`. After you have done this, go to `http://www.boost.org/users/download/` and get the source for the latest version of boost. In the Galaxia folder, create a folder named boost, then unpack the boost source into it. Open a terminal, `cd` to boost and type `./bootstrap.sh`. After this it done it will tell you to execute `./b2`. While root is building its binaries you can go grab a coffee, it takes quite a while.

1.2 Compiling

1. `cd` into the Galaxia dir.
2. Use `cmake .` to build the makefile.
3. Use `make` for compilation.
4. You can now run Galaxia with `./bin/Galaxia`
5. Use `doxygen Doxyfile`. This will generate documentation in Galaxia/doc .

1.3 Acknowledgements

We use Jeremy Conlin's Gnuplot interface for C++. You can find it at https://code.google.com/archive/p/gnuplot-cpp/.

Author

Maximilian Caspar
Johannes Esser

Date

2016

Chapter 2

Todo List

Member export_plot (const std::vector< Star > &cluster, const double scale)

Make saving files more efficient, ditch the timer.

File plot.h

Add file IO

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 gFile Class Reference

`#include <fileio.h>`

Public Member Functions

- gFile (string in_name, bool write_file)

  Sets the file name and determines swether to overwrite already existing files.
- vector< Star > load ()

  Returns the array of stars loaded from the .galaxia file.
- void save (vector< Star > &out_stars)

  Writes a vector of stars into a .galaxia file.

5.1.1 Detailed Description

A class used for input/output capability. It saves and loads .galaxia files containing boost archives.

Definition at line 27 of file fileio.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 gFile::gFile (string *in_name,* bool *write_file*) `[inline]`

Sets the file name and determines swether to overwrite already existing files.

The constructor for gFile.

Parameters

in_name	The file name (relative to main path)

Definition at line 35 of file fileio.h.

5.1.3 Member Function Documentation

5.1.3.1 void gFile::save (vector< Star > & *out_stars*) [inline]

Writes a vector of stars into a .galaxia file.

Parameters

out_stars	The vector of stars to be saved.

Definition at line 73 of file fileio.h.

The documentation for this class was generated from the following file:

- include/fileio.h

5.2 Star Class Reference

`#include <stars.h>`

Public Member Functions

- Star (double sx, double sy, double vx, double vy, double m)
  Sets the initial position, speed and mass. The initial force acting on a star is always 0.
- Star ()
  Defaul constructor for Star, does absolutely nothing and is only here for boost.
- void move (double dt)
  Moves the star according to its speed, accelerates the star according to its force and resets the force to 0.
- template<class Archive >
  void **serialize** (Archive &ar, const unsigned int version)

Public Attributes

- double xpos
  Current x position.
- double ypos
  Current y position.
- double xvel
  Current y velocity.
- double yvel
  Current y velocity.
- double xfor
  Current x force.
- double yfor
  Current y force.
- double mass
  Mass of star.

Friends

- class **boost::serialization::access**

5.2.1 Detailed Description

A simple class that represents an object of mass with a position, a velecity and a force acting on it.

Definition at line 21 of file stars.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Star::Star (double *sx*, double *sy*, double *vx*, double *vy*, double *m*) [inline]

Sets the initial position, speed and mass. The initial force acting on a star is always 0.

The constructor for Star.

Parameters

sx	The initial x position of the star.
sy	The initial y position of the star.
vx	The initial x velocity of the star.
vy	The initial y velocity of the star.
m	The mass of the star.

Definition at line 34 of file stars.h.

5.2.3 Member Function Documentation

5.2.3.1 void Star::move (double *dt*) [inline]

Moves the star according to its speed, accelerates the star according to its force and resets the force to 0.

Parameters

dt	This sets the time interval in which the movement takes place.

Definition at line 53 of file stars.h.

The documentation for this class was generated from the following file:

- include/stars.h

Chapter 6

File Documentation

6.1 include/fileio.h File Reference

Defines a class handling conversion from stars into files and back. Requires boost.

```
#include "stars.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/serialization/vector.hpp>
```

Classes

- class gFile

6.1.1 Detailed Description

Defines a class handling conversion from stars into files and back. Requires boost.

6.2 include/gravity.h File Reference

Provides the functions needed for particles to be generated and interact.

```
#include "stars.h"
#include <cmath>
#include <random>
#include <chrono>
#include <omp.h>
#include <iostream>
```

Functions

- double get_angle (const double &x, const double &y)
- double orbit_velocity (const double &radius)
- double orbit_velocity (const double &radius, const unsigned int &mco)
- void make_galaxy (std::vector< Star > &Elements, const double &radius, const unsigned int &nos, const double &sx, const double &sy, const double &vx, const double &vy, const unsigned int &mco)
- void accelerate (std::vector< Star > &cluster, double dt)

  Calculates the forces between all stars and moves them for a given timestep.

Variables

- const double G = 6.674E-11

  The gravitational constant.
- const double MASS_OF_SUN = 1.98892E30

  The mass of our sun serves as a template when creating galaxies.
- const double LIGTHYEAR = 9.46073E15

  The lightyear is needed for calculating orbital speed distribution.

6.2.1 Detailed Description

Provides the functions needed for particles to be generated and interact.

6.2.2 Function Documentation

6.2.2.1 void accelerate (std::vector< Star > & *cluster,* double *dt*)

Calculates the forces between all stars and moves them for a given timestep.

Parameters

cluster	The vector containing all stars, called by reference
dt	Time intervall used for calculating the movement.

Calculation of force and movement are parallelised in order to spped up the calculation for large amounts of stars. The movement of the star is calculated from within the Star class.

Definition at line 118 of file gravity.h.

6.2.2.2 double get_angle (const double & *x,* const double & *y*)

Return the Angle of a 2D-Vector given by (x,y)

The standart C++ function 'atan2()' returns an angle for two given coordinates, but we need the angle to be between 0 and 2Π.

Definition at line 26 of file gravity.h.

6.2.2.3 void make_galaxy (std::vector< Star > & *Elements*, const double & *radius*, const unsigned int & *nos*, const double & *sx*, const double & *sy*, const double & *vx*, const double & *vy*, const unsigned int & *mco*)

Initialize a galaxy distribution and write it in an array.

Parameters

Elements	The vector of stars the galaxy is going to be written into, called by reference.
radius	The mean radius of the galaxy.
nos	The number of stars that is going to be generated.
sx	The x position of the galaxy.
sy	The y position of the galaxy.
vx	The x velocity of the galaxy.
vy	The y velocity of the galaxy.
mco	The mass ratio from star to galactic center.

The galaxies are generated as random distributions of stars around a specific point in space. The orbital speed of the stars is set by an external function. A supermassive galactic center is set.

Definition at line 81 of file gravity.h.

6.2.2.4 double orbit_velocity (const double & *radius*)

Return the orbit velocity of a star based on the distance to the galaxy center.

Parameters

radius	The distance from galaxy center in meters The orbital velocity can be approximated as a funtion dependend only on the distance to the galaxy center according to http://spacemath.gsfc.nasa.gov/Calculus/6Page106.pdf . The output had to be scaled for usable results.

< Normalize radius to 10000 lightyears.

Definition at line 46 of file gravity.h.

6.2.2.5 double orbit_velocity (const double & *radius*, const unsigned int & *mco*)

Return the orbit velocity of a star based on the distance to the galaxy center and the mass ratio for the galactic center.

Parameters

radius	The distance from galaxy center in meters The orbital velocity can be approximated by assuming a homogeneous mass distribution. However, in discrete simulations it's more sensible to work with the direct approach $v_o = \sqrt{\frac{G \cdot M}{r}}$

Definition at line 60 of file gravity.h.

6.3 include/plot.h File Reference

Most of the conversion and talking to the Gnuplot interface happens in here.

```
#include "stars.h"
#include "gnuplot_i.hpp"
#include <stdio.h>
#include <ctime>
#include <assert.h>
```

Functions

- unsigned int number_of_digits (unsigned int i)

  Calculate the number of digits of an unsigned int based on its log10.
- string **filename** (unsigned int n)
- void make_video (bool delete_plotfiles)

  Turn all of the plot files in /out into a video file.
- void wait_for_key ()

  A simple way of keeping a program open, for example to keep a plot window from closing. A simple, system independent way of delaying a program until the user inputs a key. Taken from the example program that comes with the Gnuplot interface.
- void export_xy (const std::vector< Star > &cluster, std::vector< double > &x, std::vector< double > &y)

  Converts the vector of all stars into x,y values needed by the Gnuplot interface.
- void init_xy (std::vector< double > &x, std::vector< double > &y, unsigned int n)

  Initializes the vectors used for calling Gnuplot. Needed only once.
- void export_plot (const std::vector< Star > &cluster, const double scale)

  A makro for exporting an image of the current stars. Assumes (0,0) to be the middle of the plot.

Variables

- unsigned int nop = 0

  The global file counter.

6.3.1 Detailed Description

Most of the conversion and talking to the Gnuplot interface happens in here.

Todo Add file IO

6.3.2 Function Documentation

6.3.2.1 void export_plot (const std::vector< Star > & *cluster*, const double *scale*)

A makro for exporting an image of the current stars. Assumes (0,0) to be the middle of the plot.

Parameters

cluster	The vector containing all stars.
scale	Determines the scale of the plot.

Todo Make saving files more efficient, ditch the timer.

Definition at line 167 of file plot.h.

6.3.2.2 void export_xy (const std::vector< Star > & *cluster*, std::vector< double > & *x*, std::vector< double > & *y*)

Converts the vector of all stars into x,y values needed by the Gnuplot interface.

Parameters

cluster	The list of all stars in the simulation, called by reference.
x	Output list of all x values, called by reference.
y	Output list of all y values, called by reference.

Definition at line 132 of file plot.h.

6.3.2.3 void init_xy (std::vector< double > & *x*, std::vector< double > & *y*, unsigned int *n*)

Initializes the vectors used for calling Gnuplot. Needed only once.

Parameters

x	List of all x values, called by reference.
y	List of all y values, called by reference.

Definition at line 149 of file plot.h.

6.3.2.4 void make_video (bool *delete_plotfiles*)

Turn all of the plot files in /out into a video file.

Parameters

delete_plotfiles	Specify wether to delete the plots or to keep them This function removes temporary files and calls ffmpeg to turn all available plot files into an .mp4 file. By passing delete_plotfiles to TRUE all .png files are removed in the process.

Definition at line 63 of file plot.h.

6.3.2.5 unsigned int number_of_digits (unsigned int *i*)

Calculate the number of digits of an unsigned int based on its log10.

Parameters

i	Input number

Returns

Number of digits

Definition at line 32 of file plot.h.

6.3.2.6 void wait_for_key ( )

A simple way of keeping a program open, for example to keep a plot window from closing. A simple, system independent way of delaying a program until the user inputs a key. Taken from the example program that comes with the Gnuplot interface.

Author

Jeremy Conlin

Definition at line 108 of file plot.h.

6.4 include/scenarios.h File Reference

Provides some pre-made scenarios to play around with.

```
#include "stars.h"
#include "plot.h"
#include "gravity.h"
#include "fileio.h"
```

Functions

- void scenario_andromeda (bool video)

  A simple collision that will occur in a few billion years. The collision as depicted in Galaxia is probalbly not massive enough because it lacks stars, but it looks rather nice.
- void scenario_milkyway (bool video)

  A rather nice overview over our home galaxy.
- void scenario_square (bool video)

  A rather boring uniform distribution.

6.4.1 Detailed Description

Provides some pre-made scenarios to play around with.

6.4.2 Function Documentation

6.4.2.1 void scenario_andromeda (bool *video*)

A simple collision that will occur in a few billion years. The collision as depicted in Galaxia is probalbly not massive enough because it lacks stars, but it looks rather nice.

Parameters

video	Decide wether to convert the output plots into a video

Definition at line 22 of file scenarios.h.

6.4.2.2 void scenario_milkyway (bool *video*)

A rather nice overview over our home galaxy.

Parameters

video	Decide wether to convert the output plots into a video

Definition at line 61 of file scenarios.h.

6.4.2.3 void scenario_square (bool *video*)

A rather boring uniform distribution.

Parameters

video	Decide wether to convert the output plots into a video

Definition at line 100 of file scenarios.h.

6.5 include/stars.h File Reference

Definition of a simple Class representing a unit of mass in the simulation.

```
#include <vector>
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/serialization/vector.hpp>
```

Classes

- class Star

6.5.1 Detailed Description

Definition of a simple Class representing a unit of mass in the simulation.

6.6 src/main.cpp File Reference

The main program containing menus and basic structure.

```
#include "scenarios.h"
#include <iomanip>
```

Functions

- void intro_art ()

  Print the program name to the console.
- void make_menu ()

  Print a basic user menu.
- int main (int argc, char const ∗argv[ ])

  Initialize components and handle user choices.

6.6.1 Detailed Description

The main program containing menus and basic structure.

6.6.2 Function Documentation

6.6.2.1 int main (int *argc,* char const ∗ *argv[ ]*)

Initialize components and handle user choices.

Parameters

argc	The number of arguments.
argv	Arguments passed to the program, currently unused.

Definition at line 76 of file main.cpp.

Index