bonsai_phys241

1.0

Generated by Doxygen 1.7.6.1

Fri Mar 21 2014 00:12:45

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 bonsai Namespace Reference

The bonsai module provides a set of wrapping functions to the Bonsai tree code (https://github.com/fizxmike/Bonsai)

Functions

- def run_tipsy

  Runs Bonsai with initial conditions defined by tipsy file.
- def run_mode

  Run Bonsai in mode "plummer", "sphere" or "infile".
- def run_plummer

  Run a Bonsai's built in plummer model.
- def run_sphere

  Run a Bonsai's built in plummer model.

Variables

- string BONSAI_BIN = "../Bonsai/runtime/bonsai2_slowdust"

  Path to Bonsai binary.

4.1.1 Detailed Description

The bonsai module provides a set of wrapping functions to the Bonsai tree code (https://github.com/fizxmike/Bonsai)

4.1.2 Function Documentation

4.1.2.1 def bonsai.run_mode (*mode, nPart_or_file, snap_prefix, T, dt, dSnap, eps, bonsai_bin, mpi_n, mpi_log_file*)

Run Bonsai in mode "plummer", "sphere" or "infile".

This is an internal function, use the other interfaces instead.

Parameters

in	*mode*	"plummer" or "sphere" or "infile"
in	*nPart_or_file*	number of particles per mpi process, or path to tipsy file for "infile" mode
in	*snap_prefix*	path prefix for snapshot (tipsy) files (simulation time will be appended)
in	*T*	total simulation time
in	*dt*	internal time step
in	*dSnap*	interval at which snapshot files are generated
in	*bonsai_bin*	path to bonsai exe
in	*mpi_n*	specifies the number of mpi processes (0 = mpi not used)
in	*mpi_log_file*	single log file for mpi output (when mpi_n $>$ 0)

Returns

None

See also

run_tipsy(), run_plummer(), run_sphere()

4.1.2.2 def bonsai.run_plummer (*nParticles, snap_prefix, T* = `2`, *dt* = `0.0625`, *dSnap* = `0.0625`, *eps* = `0.05`, *bonsai_bin* = `None`, *mpi_n* = `0`, *mpi_log_file* = `"mpiout.log"`)

Run a Bonsai's built in plummer model.

Parameters

in	*nParticles*	number of particles (per mpi process)
in	*snap_prefix*	path prefix for snapshot files (time will be appended)
in	*T*	total simulation time
in	*dt*	internal time step
in	*dSnap*	interval at which snapshot files are generated
in	*bonsai_bin*	path to bonsai exe
in	*mpi_n*	specifies the number of mpi processes (0 = mpi not used)
in	*mpi_log_file*	single log file for mpi output (when mpi_n $>$ 0)

Returns

None

4.1.2.3 def bonsai.run_sphere (*nParticles, snap_prefix, T* = 2, *dt* = 0.0625, *dSnap* = 0.0625, *eps* = 0.05, *bonsai_bin* = None, *mpi_n* = 0, *mpi_log_file* = "mpiout.log")

Run a Bonsai's built in plummer model.

Parameters

in	*nParticles*	number of particles (per mpi process)
in	*snap_prefix*	path prefix for snapshot files (time will be appended)
in	*T*	total simulation time
in	*dt*	internal time step
in	*dSnap*	interval at which snapshot files are generated
in	*bonsai_bin*	path to bonsai exe
in	*mpi_n*	specifies the number of mpi processes (0 = mpi not used)
in	*mpi_log_file*	single log file for mpi output (when mpi_n $>$ 0)

Returns

None

4.1.2.4 def bonsai.run_tipsy (*tipsy_file, snap_prefix, T, dt, dSnap, eps, bonsai_bin* = None, *mpi_n* = 0, *mpi_log_file* = "mpiout.log")

Runs Bonsai with initial conditions defined by tipsy file.

Parameters

in	*tipsy_file*	containing initial conditions
in	*snap_prefix*	path prefix for snapshot files (time will be appended)
in	*T*	total simulation time
in	*dt*	internal time step
in	*dSnap*	interval at which snapshot files are generated
in	*bonsai_bin*	path to bonsai exe
in	*mpi_n*	specifies the number of mpi processes (0 = mpi not used)
in	*mpi_log_file*	single log file for mpi output (when mpi_n $>$ 0)

Returns

None

4.1.3 Variable Documentation

4.1.3.1 string bonsai::BONSAI_BIN = "../Bonsai/runtime/bonsai2_slowdust"

Path to Bonsai binary.

Default path, assuming bonsai_phys241 and Bonsai share parent folders

4.2 tipsy Namespace Reference

The tipsy module provides utilites for managing input and output in .tipsy format for nbody simulations.

Classes

- class Stars

  The Stars object holds mass, position, velocity, and particle IDs extracted from a .tipsy file.

Functions

- def make_mp4

  Makes an MP4 video from a set of PNG files.
- def read_tipsy

  Reads a set of tipsy files and returns an array of Star objects or plots figures.
- def txt2tipsy

  Converts a typical nbody or Aarseth text file to .tipsy format.

4.2.1 Detailed Description

The tipsy module provides utilites for managing input and output in .tipsy format for nbody simulations.

4.2.2 Function Documentation

4.2.2.1 def tipsy.make_mp4 (*png_prefix*, *mp4_prefix*, *frame_rate* = 20, *bit_rate* = '8000k', *codec* = 'libx264')

Makes an MP4 video from a set of PNG files.

Generates "[mp4_prefix].mp4" video from a set of "[png_prefix]{number}.png" where {number} is a placeholder for consecutive integers.

Parameters

in	*png_prefix*	prefix of png files
in	*mp4_prefix*	name of .mp4 file
in	*frame_rate*	in frames per second (default: 20)
in	*bit_rate*	(string) in bits per second, higer rate = higer quality (default: '10000k')
in	*codec*	used to encode video, may require extra libraries on system (defaut: 'libx264')

Returns

None

4.2.2.2 def tipsy.read_tipsy (*tipsy_prefix, figures_prefix* = `None`, *lim* = `.8`, *pointsize* = `.1`, *nRed* = `None`, *nThreads* = `4`)

Reads a set of tipsy files and returns an array of Star objects or plots figures.

Reads a set of "[tipsy_prefix]{number}" files, where {number} is a placeholder for consecutive dicimal numbers, and returns an array of Star() objects.

Optionally, if figures_prefix is not None, will generate a set of figures "[figures_prefix]{index}.png", where {index} is found from sorting {number}, and returns nothing (saves RAM for large set of tipsy files).

Parameters

in	*tipsy_prefix*	prefix of tipsy files
in	*figures_prefix*	generates figures only, no array returned (default: None)
in	*lim*	limits the range of all axis in view (default: .8)
in	*pointsize*	size of points in figure
in	*nRed*	figure colors the first nRed particles red and the remaining blue (default: None)

Returns

an array of Star objects (only if figures_prefix is None)

4.2.2.3 def tipsy.txt2tipsy (*nbody_file, tipsy_file*)

Converts a typical nbody or Aarseth text file to .tipsy format.

Header may be in one of two formats:

- [particle_count] [time_stamp]
- [particle_count] [eta] [dt] [tmax] [eps2]

Parameters

in	*nbody_file*	path to the nbody text formated file
in	*tipsy_file*	path to the tipsy output file

Returns

None

Chapter 5

Class Documentation

5.1 tipsy.Stars Class Reference

The Stars object holds mass, position, velocity, and particle IDs extracted from a .tipsy file.

Public Member Functions

- def __init__

  Constructs Stars object from a .tipsy file.
- def boost

  Add a net velocity (3-vector) to all stars.
- def translate

  Rigidly displaces all stars.
- def rotate_euler_deg

  *Rotates all stars using Euler angles in degrees (*http://mathworld.-wolfram.com/EulerAngles.html*)*
- def rotate_euler

  *Rotates all stars using Euler angles in radians (*http://mathworld.wolfram.-com/EulerAngles.html*)*
- def append

  Appends stars from a tipsy file into this Stars object.
- def save_tipsy

  Saves the Stars object in tipsy format.
- def save_figure

  Generates a figure "[figure_name].png" for this Star object.

Static Public Attributes

- time = None

Simulation timestamp (carried over from .tipsy file)

- mass = None

  Array of star masses.

- pos = None

  Array of star positions.

- vel = None

  Array of star velocities.

- IDs = None

  Array of star id numbers.

- int nStars = 0

  Number of stars stored in this object.

5.1.1 Detailed Description

The Stars object holds mass, position, velocity, and particle IDs extracted from a .tipsy file.

Currently on stars are processed (gas and dark matter ignored). Also, the phi parameter is used as a particle ID (like Bonsai).

5.1.2 Constructor & Destructor Documentation

5.1.2.1 def tipsy.Stars.__init__ (*self, tipsyFilePath*)

Constructs Stars object from a .tipsy file.

Converts from binary .tipsy format to a python object

Parameters

in	*tipsyFilePath*	path to a single .tipsy file

Returns

an instance of the Stars object

5.1.3 Member Function Documentation

5.1.3.1 def tipsy.Stars.append (*self, tipsyFilePath*)

Appends stars from a tipsy file into this Stars object.

The particle IDs will continue to increment starting from Stars.nStars

Parameters

in	*tipsyFilePath*	path to tipsy file to be appended

Returns

None

5.1.3.2 def tipsy.Stars.boost (*self, velocity*)

Add a net velocity (3-vector) to all stars.

The velocity vector is added to the current volocity of each star

Parameters

in	*velocity*	a tuple of lenght 3

Returns

None

5.1.3.3 def tipsy.Stars.rotate_euler (*self, phi, theta, psi*)

Rotates all stars using Euler angles in radians (http://mathworld.wolfram.-com/EulerAngles.html)

Angles are made negative within this function for "active" rotation of the star position. Rotations are applied in the function argument order.

Parameters

in	*phi*	(radians) right hand rotation about +Z axis
in	*theta*	(radians) right hand roation about resultant +X axis
in	*psi*	(radians) right hand rotation about resultant +Z axis

Returns

None

5.1.3.4 def tipsy.Stars.rotate_euler_deg (*self, phi, theta, psi*)

Rotates all stars using Euler angles in degrees (http://mathworld.wolfram.-com/EulerAngles.html)

Angles are made negative within this function for "active" rotation of the star position. Rotations are applied in the function argument order.

Parameters

in	*phi*	(degrees) right hand rotation about +Z axis
in	*theta*	(degrees) right hand roation about resultant +X axis
in	*psi*	(degrees) right hand rotation about resultant +Z axis

Returns

None

5.1.3.5 def tipsy.Stars.save_figure (*self, figure_name, lim = .8, figsize = 10, pointsize = .1, nRed = None*)

Generates a figure "[figure_name].png" for this Star object.

Parameters

in	*figure_name*	path where figure is to be saved
in	*lim*	limits the range of all axis in view (default: .8)
in	*figsize*	size of final image (.png)
in	*pointsize*	size of stars
in	*nRed*	figure colors the first nRed particles red and the remaining blue (default: None)

Returns

path to file just saved (string)

5.1.3.6 def tipsy.Stars.save_tipsy (*self, tipsyFilePath*)

Saves the Stars object in tipsy format.

Prints saved file path to stdout

Parameters

in	*tipsyFilePath*	path to output file

Returns

None

5.1.3.7 def tipsy.Stars.translate (*self, displacement*)

Rigidly displaces all stars.

The displacement vector is added to each star location.

Parameters

in	*displacement*	-- a vector in the direction of the desired displacement

Returns

None

5.1.4 Member Data Documentation

5.1.4.1 tipsy.Stars::IDs = None [static]

Array of star id numbers.

5.1.4.2 tipsy.Stars::mass = None [static]

Array of star masses.

5.1.4.3 int tipsy.Stars::nStars = 0 [static]

Number of stars stored in this object.

5.1.4.4 tipsy.Stars::pos = None [static]

Array of star positions.

5.1.4.5 tipsy.Stars::time = None [static]

Simulation timestamp (carried over from .tipsy file)

5.1.4.6 tipsy.Stars::vel = None [static]

Array of star velocities.

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

- tipsy.py

Chapter 6

File Documentation

6.1 bonsai.py File Reference

Namespaces

- namespace bonsai

  The bonsai module provides a set of wrapping functions to the Bonsai tree code (https://github.com/fizxmike/Bonsai)

Functions

- def bonsai.run_tipsy

  Runs Bonsai with initial conditions defined by tipsy file.
- def bonsai.run_mode

  Run Bonsai in mode "plummer", "sphere" or "infile".
- def bonsai.run_plummer

  Run a Bonsai's built in plummer model.
- def bonsai.run_sphere

  Run a Bonsai's built in plummer model.

Variables

- string bonsai.BONSAI_BIN = "../Bonsai/runtime/bonsai2_slowdust"

  Path to Bonsai binary.

6.2 tipsy.py File Reference

Classes

- class tipsy.Stars

  The Stars object holds mass, position, velocity, and particle IDs extracted from a .tipsy file.

Namespaces

- namespace tipsy

  The tipsy module provides utilites for managing input and output in .tipsy format for nbody simulations.

Functions

- def tipsy.make_mp4

  Makes an MP4 video from a set of PNG files.
- def tipsy.read_tipsy

  Reads a set of tipsy files and returns an array of Star objects or plots figures.
- def tipsy.txt2tipsy

  Converts a typical nbody or Aarseth text file to .tipsy format.