nbody2 Documentation

Release

Thomas Malthouse

May 19, 2017

CONTENTS:

CHAPTER ONE

VEC3

vec3

A 3-dimensional vector of doubles, powered by clang vector extensions. Individual elements are accessible with dot (a.x) or array (a[0]) notation.

vec3_0

A zero-vector for the vec3 type

vec3_I

A vec3 v where v.x=1, and v.y=0, v.z=0

vec3_J

A vec3 v where v.x=0, v.y=1, v.z=0

vec3_K

A vec3 v where v.x=0, v.y=0, v.z=1

double **vabs** (*vec3 v*)

This function gives the absolute value of a given vector v

bool **vec3_eq** (*vec3 v*, *vec3 w*)

This function checks two vectors for element equality. Note that because of floating point error, this function is mostly useful for checking if a vector is uninitialized (i.e. the 0 vector.)

vec3 **vec3_unit** (*vec3 v*)

Given a vector, this function returns the unit vector (i.e. the vector pointing in the same direction with magnitude 1).

CHAPTER

TWO

SYSTEM

system.c system.h

System

A System object represents a collection of celestial bodies and their state at a given time.

*Body** **System.bodies**

An array of Body objects

uint **System.count**

The number of bodies in System.bodies

uint64_t **time**

The current time of the system, in the number of seconds since some epoch (the beginning of the simulation, perhaps?)

TreeNode **tree**

The tree representing the system.

void **update_system** (*System* **sys*)

This function updates the given system by the smallest timestep possible.

CHAPTER THREE

BODY

body.c body.h

Body

A `Body` object represents a celestial body (or other abstract celestial object.) It contains state information, body type, and other pertinent data.

uint32_t **Body.id**

This is a unique 32-bit integer, used to check whether two bodies are the same. Duplicate IDs will lead to inaccurate calculations. By default, IDs are based on the body's position in the main Body array.

vec3 **Body.pos**

This holds the position information

vec3 **Body.vel**

This holds the velocity information

BodyType **Body.type**

This holds information about the class of the body (gas, dust, DM, etc.) Not used at the moment, it will eventually be used for hydrodynamic calculation

double **Body.mass**

This field ONLY exists if the macro UNIT_MASS is undefined. Its purpose should be self-explanatory.

vec3 **acc**

This field holds the acceleration at the last timestep. It shouldn't be useful for anything other than calculating the new timestep.

uint64_t **tstep**

This is the current timestep, in seconds. It will always be a power of 2.

uint64_t **update_timestep**(*Body *b*, uint64_t *cur_time*)

This function updates the given body's timestep, using the last step's acceleration as a guide. The algorithm used comes from Noah Muldavin's thesis (2013).

void **update_body**(*Body *b*, TreeNode **tree*)

This function updates the given body by one timestep. The rest of the system is contained in the `tree` parameter, which makes the calculations much faster than going through the bodies individually.

3.1 Private Functions and Types

The following entries are only visible from the body.c file. They are just documented here for reference.

NodeList

This type represents a simple dynamic array of TreeNode*s (similar to C++'s vector<T>.)

void **NodeList_append**(*NodeList* *l, TreeNode *n)

This function adds element `n` to the end of NodeList `l` (allocating more memory if necessary.)

uint64_t **t_ideal**(*Body* *b)

This function calculates the ideal timestep for a given body, using an algorithm from Muldavin 2013.

node_finder(*NodeList* *l, *vec3* *pos*, TreeNode *tree)

This function walks through the given tree, adding any node that needs to be accounted for to NodeList `l`.

body_acc(TreeNode **nodes, size_t *node_count*, *Body* *b)

This function calculates the acceleration on Body `b`, using the list of nodes found by `node_finder`.

vec3 **g_acc**(*vec3* *pos1*, *vec3* *pos2*, double *m2*)

This function calculates the acceleration at `pos1` caused by an object at `pos2` with mass `m2`, using the standard equation

r = pos2-pos1

f_{acc} = frac{G m_2}{(|r|)^2}hat{r}

vec3 **body_acc**(TreeNode **nodes, size_t *node_count*, *Body* *b)

This function calculates the total acceleration on body `b`, calling `g_acc` on each of the provided nodes and adding all the results together.

CHAPTER FOUR

TREENODE

tree.c tree.h

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX