

rl_basicray Reference Manual
1.0.10

Generated by Doxygen 1.5.1

Fri Jun 27 15:52:40 2008

Contents

1section.2	rl_basicray Directory Hierarchy	2
3	rl_basicray Class Index	2
4	rl_basicray Directory Documentation	3
5	rl_basicray Class Documentation	3

1 [rl_BasicRay](#) User's Guide

1.1 Copyright and License

1.1.1 Copyright and License

Copyright (C) 2006 Smithsonian Astrophysical Observatory

This file is part of the `rl_basicray` package.

`rl_basicray` is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

`tracefct` is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor Boston, MA 02110-1301, USA

Author:

`Terry Gaetz`

1.2 Purpose

1.2.1 Purpose

The `rl_BasicRay` library consists of a set of C++ classes for manipulating rays.

1.3 A Basic Ray Class

1.3 A Basic Ray Class

The `rl_BasicRay` class represents a stripped-down ray consisting of position, direction, energy, and an id number. It serves as a base class for a more general ray class `rl_Ray` (part of the `rl_raylib` package) which adds in support for ray polarization information.

The `rl_BasicRay` library includes a class encapsulating a basic ray consisting of position and direction information, ray energy, and a ray id number.

The ray is nominally defined in a standard coordinate system (STD) which serves as a default global coordinate system. The ray can be transformed into a local “body-centered” coordinate system (BCS) (e.g., attached to a piece of hardware such as a mirror).

In transforming from the STD system to the BCS system, the ray is first translated by the difference between the BCS origin and the STD origin, then transformed to the orientation of the BCS system relative to the STD system by rotation about the BCS origin.

To transform back to the STD system the operations are performed in reverse: first “derotate” the vector about the BCS origin to account for the different orientation, then “detranslate” the position by the difference between the BCS and STD origins.

Other operations on a basic ray include projection (moving the ray position by a given distance in the direction specified by the ray direction vector) and reflection of the ray direction vector about a surface normal provided by the user. In the reflection operation, it is assumed that the ray position is at the surface about which the reflection takes place.

2 `rl_basicray` Directory Hierarchy

2 `rl_basicray` Directory Hierarchy

2.1 `rl_basicray` Directories

2.1 `rl_basicray` Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

`rl_basicray` 3

3 `rl_basicray` Class Index

3 `rl_basicray` Class Index

3.1 `rl_basicray` Class List

3.1 rl_basicray Class List

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

rl_BasicRay	3
rl_RayMath	12

4 rl_basicray Directory Documentation

4 rl_basicray Directory Documentation

4.1 rl_basicray/ Directory Reference

4.1 rl_basicray/ Directory Reference



rl_basicray

Files

- file [rl_BasicRay.cc](#)
- file [rl_BasicRay.h](#)
- file [rl_RayMath.h](#)

5 rl_basicray Class Documentation

5 rl_basicray Class Documentation

5.1 rl_BasicRay Class Reference

5.1 rl_BasicRay Class Reference

```
#include <rl_BasicRay.h>
```

Public Member Functions

- virtual [~rl_BasicRay \(\)](#)
A virtual do-nothing destructor.
- [rl_BasicRay \(\)](#)

- [rl_BasicRay](#) (dvm3_Vector const &pos, dvm3_Vector const &dir, double energy, long int id)
- [rl_BasicRay](#) (rl_BasicRay const &other)
- void [set_id](#) (long int id)
- long int [id](#) () const
- double & [energy](#) ()
- double [energy](#) () const
- double & [position](#) (int i)
- double [position](#) (int i) const
- dvm3_Vector & [position](#) ()
- dvm3_Vector const & [position](#) () const
- double & [direction](#) (int i)
- double [direction](#) (int i) const
- dvm3_Vector & [direction](#) ()
- dvm3_Vector const & [direction](#) () const
- double [reflect_direction_vector](#) (dvm3_Vector const &normal)
- void [project](#) (double s)
- virtual void [translate_rotate](#) (dvm3_Vector const &trans, dvm3_RotMat const &rotmat)
- virtual void [derotate_detranslate](#) (dvm3_Vector const &trans, dvm3_RotMat const &rotmat)
- std::ostream & [print_on](#) (std::ostream &os, char const prefix[]="", char const postfix[]="") const

Protected Member Functions

- void [init](#) (dvm3_Vector const &pos, dvm3_Vector const &dir, double energy, long int id)

Protected Attributes

- dvm3_Vector [pos_](#)
ray position
- dvm3_Vector [dir_](#)
ray direction vector (direction cosines)
- double [energy_](#)
ray energy
- long int [id_](#)
ray id number

Friends

- `std::ostream & operator<<` (`std::ostream &os, rl_BasicRay const &`)

5.1.1 Detailed Description

A basic ray: encapsulates position and direction vectors, energy, and ray id.

Definition at line 47 of file `rl_BasicRay.h`.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 `rl_BasicRay::~~rl_BasicRay ()` [virtual]

A virtual do-nothing destructor.

Definition at line 41 of file `rl_BasicRay.cc`.

5.1.2.2 `rl_BasicRay::rl_BasicRay ()` [inline]

Default constructor; constructs a ray in an INVALID state. Use `init` to initialize the fields.

Definition at line 274 of file `rl_BasicRay.h`.

5.1.2.3 `rl_BasicRay::rl_BasicRay (dvm3_Vector const & pos, dvm3_Vector const & dir, double energy, long int id)` [inline]

Constructor. Construct an `rl_BasicRay` from ray initial position and direction vector, energy, and id number.

Parameters:

pos ray initial position.

dir ray initial direction unit vector.

energy ray energy, in keV.

id ray id.

Definition at line 279 of file `rl_BasicRay.h`.

5.1.2.4 `rl_BasicRay::rl_BasicRay (rl_BasicRay const & other)` [inline]

Copy constructor

Parameters:

other ray to be copied.

Definition at line 286 of file rl_BasicRay.h.

5.1.3 Member Function Documentation

5.1.3.1 void rl_BasicRay::init (dvm3_Vector const & *pos*, dvm3_Vector const & *dir*, double *energy*, long int *id*) [inline, protected]

Initialize [rl_BasicRay](#) with ray initial position and direction vector, energy, and id number

Parameters:

- pos* ray position vector
- dir* ray direction vector
- energy* ray energy
- id* ray id number

Definition at line 292 of file rl_BasicRay.h.

References [dir_](#), [energy_](#), [id_](#), and [pos_](#).

5.1.3.2 void rl_BasicRay::set_id (long int *id*) [inline]

Reset ray id

Parameters:

- id* new id value.

Definition at line 310 of file rl_BasicRay.h.

References [id_](#).

5.1.3.3 long int rl_BasicRay::id () const [inline]

Read-only access to current ray id

Returns:

- ray id number

Definition at line 306 of file rl_BasicRay.h.

References [id_](#).

5.1.3.4 `double & rl_BasicRay::energy ()` [inline]

Read/write access to current ray energy

Returns:

a reference to the ray energy

Definition at line 315 of file `rl_BasicRay.h`.

References `energy_`.

5.1.3.5 `double rl_BasicRay::energy () const` [inline]

Read-only access to current ray position

Returns:

the ray energy

Definition at line 320 of file `rl_BasicRay.h`.

References `energy_`.

5.1.3.6 `double & rl_BasicRay::position (int i)` [inline]

Read/write access to component *i* of current ray position

Parameters:

i index.

Returns:

a reference to the *i* component of the position vector.

Definition at line 325 of file `rl_BasicRay.h`.

References `pos_`.

5.1.3.7 `double rl_BasicRay::position (int i) const` [inline]

Read-only access to component *i* of current ray position

Parameters:

i index.

Returns:

the *i* component of the position vector.

Definition at line 330 of file rl_BasicRay.h.

References pos_.

5.1.3.8 `dvm3_Vector & rl_BasicRay::position ()` [inline]

Read/write access to current ray position

Returns:

a reference to the position vector dvm3_Vector.

Definition at line 335 of file rl_BasicRay.h.

References pos_.

5.1.3.9 `dvm3_Vector const & rl_BasicRay::position () const` [inline]

Read-only access to current ray position

Returns:

a const reference to the position vector dvm3_Vector

Definition at line 340 of file rl_BasicRay.h.

References pos_.

5.1.3.10 `double & rl_BasicRay::direction (int i)` [inline]

Read/write access to component *i* of current ray direction

Parameters:

i index.

Definition at line 345 of file rl_BasicRay.h.

References dir_.

5.1.3.11 `double rl_BasicRay::direction (int i) const` [inline]

Read-only access to component *i* of current ray direction

Parameters:

i index.

Returns:

component i of the position vector.

Definition at line 350 of file `rl_BasicRay.h`.

References `dir_`.

5.1.3.12 `dvm3_Vector & rl_BasicRay::direction ()` [inline]

Read/write access to current ray direction vector

Returns:

reference to current ray direction vector `dvm3_Vector`

Definition at line 355 of file `rl_BasicRay.h`.

References `dir_`.

5.1.3.13 `dvm3_Vector const & rl_BasicRay::direction () const` [inline]

Read-only access to current ray direction vector

Returns:

const reference to current ray direction vector `dvm3_Vector`

Definition at line 360 of file `rl_BasicRay.h`.

References `dir_`.

5.1.3.14 `double rl_BasicRay::reflect_direction_vector (dvm3_Vector const & normal)`

Reflect this ray's direction vector about a (surface) normal

Parameters:

normal - surface normal unit vector to be used in reflecting this [rl_BasicRay](#).

Returns:

sine of the graze angle between the ray and the surface

Definition at line 52 of file `rl_BasicRay.cc`.

References `dir_`.

5.1.3.15 `void rl_BasicRay::project (double s) [inline]`

Project a distance *s* along this ray

Parameters:

s distance to project the ray.

Definition at line 369 of file `rl_BasicRay.h`.

References `dir_`, and `pos_`.

5.1.3.16 `void rl_BasicRay::translate_rotate (dvm3_Vector const & trans, dvm3_RotMat const & rotmat) [inline, virtual]`

VIRTUAL: Translate to the BCS origin; rotate from STD to BCS coordinates

Parameters:

trans translation vector.

rotmat rotation matrix to be applied. *rotmat* specifies a rotation from `std` to `bcs`.

Definition at line 379 of file `rl_BasicRay.h`.

References `dir_`, `pos_`, `rl_RayMath::rotate()`, and `rl_RayMath::translate_rotate()`.

5.1.3.17 `void rl_BasicRay::derotate_detranslate (dvm3_Vector const & trans, dvm3_RotMat const & rotmat) [inline, virtual]`

VIRTUAL: Derotate back to STD coordinates; detranslate back to STD origin

Parameters:

trans translation vector.

rotmat rotation matrix to be applied. *rotmat* specifies a rotation from `std` to `bcs`; the inverse of *rotmat* is applied to the ray.

Definition at line 391 of file `rl_BasicRay.h`.

References `rl_RayMath::derotate()`, `rl_RayMath::derotate_detranslate()`, `dir_`, and `pos_`.

5.1.3.18 `std::ostream& rl_BasicRay::print_on (std::ostream & os, char const prefix[] = "", char const postfix[] = "") const`

Print the ray properties to an output stream.

Parameters:

os output stream.
prefix an optional prefix string.
postfix an optional postfix string.

Returns:

output stream

5.1.4 Friends And Related Function Documentation**5.1.4.1 `std::ostream& operator<<` (`std::ostream & os, rl_BasicRay const &)`
[friend]**

Print the ray properties to an output stream.

Parameters:

os output stream.

Returns:

output stream

5.1.5 Member Data Documentation**5.1.5.1 `dvm3_Vector rl_BasicRay::pos_` [protected]**

ray position

Definition at line 52 of file `rl_BasicRay.h`.

Referenced by `derotate_detranslate()`, `init()`, `position()`, `project()`, and `translate_rotate()`.

5.1.5.2 `dvm3_Vector rl_BasicRay::dir_` [protected]

ray direction vector (direction cosines)

Definition at line 54 of file `rl_BasicRay.h`.

Referenced by `derotate_detranslate()`, `direction()`, `init()`, `project()`, `reflect_direction_vector()`, and `translate_rotate()`.

5.1.5.3 `double rl_BasicRay::energy_` [protected]

ray energy

Definition at line 56 of file rl_BasicRay.h.

Referenced by energy(), and init().

5.1.5.4 long int rl_BasicRay::id_ [protected]

ray id number

Definition at line 58 of file rl_BasicRay.h.

Referenced by id(), init(), and set_id().

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

- rl_BasicRay.h
- rl_BasicRay.cc

5.2 rl_RayMath Class Reference

5.2 rl_RayMath Class Reference

```
#include <rl_RayMath.h>
```

Static Public Member Functions

- static void [rotate](#) (dvm3_RotMat const &rotmat, dvm3_Vector &vector)
- static void [rotate](#) (dvm3_RotMat const &rotmat, double vector[])
- static void [derotate](#) (dvm3_RotMat const &rotmat, dvm3_Vector &vector)
- static void [derotate](#) (dvm3_RotMat const &rotmat, double vector[])
- static void [translate_rotate](#) (dvm3_Vector const &trans, dvm3_RotMat const &rotmat, dvm3_Vector &vector)
- static void [translate_rotate](#) (double const trans[], dvm3_RotMat const &rotmat, double vector[])
- static void [derotate_detranslate](#) (dvm3_Vector const &trans, dvm3_RotMat const &rotmat, dvm3_Vector &vector)
- static void [derotate_detranslate](#) (double const trans[], dvm3_RotMat const &rotmat, double vector[])

5.2.1 Detailed Description

A class gathering together static methods for operations on ray position or direction vectors.

The class is a simple class to handle common numerical operations on 3-vectors of floating point T_fps representing ray position vectors and ray direction vectors.

[rl_RayMath](#) has only static member functions; there are no member data. Where possible, the static member functions are inlined.

The static member functions facilitate transformations between an initial coordinate system (termed "Standard" or STD) and another coordinate system (termed "Body Centered" or BCS). This is just convenient labeling which reflects a common usage of the transformations.

The BCS coordinate origin is (optionally) translated relative to the STD origin, and the orientation of the BCS coordinate axes relative to the STD axes is described by a proper rotation matrix.

The transformation from STD to BCS is accomplished by "rotate_translate" (or "rotate" if the coordinate origins are coincident).

The transformation from BCS back to STD is accomplished by "detranslate_derotate" (or "derotate" if the coordinate origins are coincident).

Direction vectors are always transformed using the "rotate"- "derotate" pair, NEVER the versions involving translation.

Positional vectors are transformed using "rotate_translate" and "detranslate_derotate"; if the STD and BCS coordinate systems origins are coincident, the transformation can be accomplished more economically using the "rotate" and "derotate" pair.

NOTE: although `dvm3_Matrix` works with either `dvm3_Vector`'s or plain old c-style one-dimensional arrays, any call to an [rl_RayMath](#) method must use only one vector type within a given method call.

Definition at line 91 of file `rl_RayMath.h`.

5.2.2 Member Function Documentation

5.2.2.1 `void rl_RayMath::rotate (dvm3_RotMat const & rotmat, dvm3_Vector & vector) [inline, static]`

Rotate vector from STD coordinates to BCS coordinates.

Parameters:

rotmat rotation vector (STD to BCS) to be applied.

vector vector to be rotated.

Definition at line 197 of file `rl_RayMath.h`.

Referenced by `rl_BasicRay::translate_rotate()`.

5.2.2.2 `void rl_RayMath::rotate (dvm3_RotMat const & rotmat, double vector[]) [inline, static]`

Rotate vector from STD coordinates to BCS coordinates.

Parameters:

rotmat rotation vector (STD to BCS) to be applied.
vector c-style vector to be rotated.

Definition at line 206 of file rl_RayMath.h.

5.2.2.3 void rl_RayMath::derotate (dvm3_RotMat const & *rotmat*, dvm3_Vector & *vector*) [inline, static]

De-rotate vector from BCS coordinates back to STD coordinates.

Parameters:

rotmat rotation vector (STD to BCS) to be used. The *INVERSE* of *rotmat* is applied to the vector.
vector vector to be rotated.

Definition at line 219 of file rl_RayMath.h.

Referenced by rl_BasicRay::derotate_detranslate().

5.2.2.4 void rl_RayMath::derotate (dvm3_RotMat const & *rotmat*, double *vector*[]) [inline, static]

De-rotate vector from BCS coordinates back to STD coordinates.

Parameters:

rotmat rotation vector (STD to BCS) to be used. The *INVERSE* of *rotmat* is applied to the vector.
vector c-style vector to be rotated.

Definition at line 228 of file rl_RayMath.h.

5.2.2.5 void rl_RayMath::translate_rotate (dvm3_Vector const & *trans*, dvm3_RotMat const & *rotmat*, dvm3_Vector & *vector*) [inline, static]

Translate to BCS origin, then rotate from STD to BCS coordinates.

Parameters:

trans translation vector.
rotmat rotation vector (STD to BCS) to be applied.
vector vector to be derotated.

Definition at line 241 of file rl_RayMath.h.

Referenced by rl_BasicRay::translate_rotate().

5.2.2.6 void `rl_RayMath::translate_rotate` (double const *trans*[], `dvm3_RotMat` const & *rotmat*, double *vector*[]) [*inline, static*]

Translate to BCS origin, then rotate from STD to BCS coordinates.

Parameters:

- trans* translation vector.
- rotmat* rotation vector (STD to BCS) to be applied.
- vector* c-style vector to be derotated.

Definition at line 252 of file `rl_RayMath.h`.

5.2.2.7 void `rl_RayMath::derotate_detranslate` (`dvm3_Vector` const & *trans*, `dvm3_RotMat` const & *rotmat*, `dvm3_Vector` & *vector*) [*inline, static*]

Rotate from BCS to STD coordinates, then translate back to original position. This is the inverse operation to `translate_rotate`.

Parameters:

- trans* translation vector.
- rotmat* rotation vector (STD to BCS) to be applied.
- vector* c-style vector to be derotated.

Definition at line 267 of file `rl_RayMath.h`.

Referenced by `rl_BasicRay::derotate_detranslate()`.

5.2.2.8 void `rl_RayMath::derotate_detranslate` (double const *trans*[], `dvm3_RotMat` const & *rotmat*, double *vector*[]) [*inline, static*]

Rotate from BCS to STD coordinates, then translate back to original position. This is the inverse operation to `translate_rotate`.

Parameters:

- trans* c-style translation vector.
- rotmat* rotation vector (STD to BCS) to be applied.
- vector* c-style vector to be derotated.

Definition at line 278 of file `rl_RayMath.h`.

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

- `rl_RayMath.h`

Index

- ~rl_BasicRay
 - rl_BasicRay, 5
- derotate
 - rl_RayMath, 13, 14
- derotate_detranslate
 - rl_BasicRay, 10
 - rl_RayMath, 15
- dir_
 - rl_BasicRay, 11
- direction
 - rl_BasicRay, 8, 9
- energy
 - rl_BasicRay, 6
- energy_
 - rl_BasicRay, 11
- id
 - rl_BasicRay, 6
- id_
 - rl_BasicRay, 11
- init
 - rl_BasicRay, 5
- operator<<
 - rl_BasicRay, 11
- pos_
 - rl_BasicRay, 11
- position
 - rl_BasicRay, 7, 8
- print_on
 - rl_BasicRay, 10
- project
 - rl_BasicRay, 9
- reflect_direction_vector
 - rl_BasicRay, 9
- rl_BasicRay, 3
 - rl_BasicRay, 5
- rl_BasicRay
 - ~rl_BasicRay, 5
 - derotate_detranslate, 10
 - dir_, 11
 - direction, 8, 9
 - energy, 6
 - energy_, 11
 - id, 6
 - id_, 11
 - init, 5
 - operator<<, 11
 - pos_, 11
 - position, 7, 8
 - print_on, 10
 - project, 9
 - reflect_direction_vector, 9
 - rl_BasicRay, 5
 - set_id, 6
 - translate_rotate, 9
- rl_basicray/ Directory Reference, 3
- rl_RayMath, 12
- rl_RayMath
 - derotate, 13, 14
 - derotate_detranslate, 15
 - rotate, 13
 - translate_rotate, 14
- rotate
 - rl_RayMath, 13
- set_id
 - rl_BasicRay, 6
- translate_rotate
 - rl_BasicRay, 9
 - rl_RayMath, 14