

rl_raysuplib Reference Manual

1.0.8

Generated by Doxygen 1.5.1

Tue Jun 26 15:11:07 2007

Contents

1 rl_RayLib User's Guide	1
2 rl_raysuplib Directory Hierarchy	2
3 rl_raysuplib Class Index	2
4 rl_raysuplib Page Index	2
5 rl_raysuplib Directory Documentation	2
6 rl_raysuplib Class Documentation	3
7 rl_raysuplib Page Documentation	6

1 rl_RayLib User's Guide

1.1 Copyright and License

Copyright (C) 2006 Smithsonian Astrophysical Observatory

This file is part of the rl_RaySupLib package.

rl_RaySupLib 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

The `rl_RaySupLib` library consists of a set of C++ classes for manipulating rays, including the effects of multilayer reflectivity.

1.2.1 classes.

The `rl_RaySupLib` library includes a number of classes providing i/o support for `rl_RayLib`: reading initialization information from rdb database tables. These include:

- [rl_DielectricPOD_rdb](#) handles initializing `rl_DielectricPOD` from rdb tables. The rdb table contains a list of energies and the corresponding complex dielectric decrements.
- [rl_Multilayer_rdb](#) handles initializing an `rl_Multilayer` from an rdb table. The rdb table provides the information on each layer (name, thickness, roughness type) and the name of an rdb file specifying the dielectric decrement information.

2 `rl_raysuplib` Directory Hierarchy

2.1 `rl_raysuplib` Directories

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

[rl_raysuplib](#) 2

3 `rl_raysuplib` Class Index

3.1 `rl_raysuplib` Class List

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

[rl_DielectricPOD_rdb](#) 3

[rl_Multilayer_rdb](#) 4

4 `rl_raysuplib` Page Index

4.1 `rl_raysuplib` Related Pages

Here is a list of all related documentation pages:

Basic Interface 6

Purpose 6

5 rl_raysuplib Directory Documentation

5.1 rl_raysuplib/ Directory Reference

rl_raysuplib

Files

- file `rl_DielectricPOD_rdb.cc`
- file `rl_DielectricPOD_rdb.h`
- file `rl_Multilayer_rdb.cc`
- file `rl_Multilayer_rdb.h`
- file `rl_RaySupLib.h`

6 rl_raysuplib Class Documentation

6.1 rl_DielectricPOD_rdb Class Reference

```
#include <rl_DielectricPOD_rdb.h>
```

Public Member Functions

- [~rl_DielectricPOD_rdb](#) ()
- [rl_DielectricPOD_rdb](#) (char const rdb_file[]= "")
- void [init](#) (char const rdb_file[])

6.1.1 Detailed Description

A class encapsulating reading of `rl_DielectricPOD` initialization data from an rdb table. The rdb table is assumed to have at least two data rows and 3 columns:

- energy (keV)

- alpha (real part of dielectric decrement)
- gamma (imaginary part of the dielectric decrement)

The complex dielectric constant has real part (1-alpha) and imaginary part (-gamma).

Definition at line 58 of file rl_DielectricPOD_rdb.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 rl_DielectricPOD_rdb::~~rl_DielectricPOD_rdb ()

Destructor

Definition at line 69 of file rl_DielectricPOD_rdb.cc.

6.1.2.2 rl_DielectricPOD_rdb::rl_DielectricPOD_rdb (char const *rdb_file*[] = "")

Constructor.

Parameters:

rdb_file name of the /rdb file to be read (optional). If *rdb_file* is a nonempty string, read in energy, alpha, and gamma from the specified /rdb table. The array is sorted on the energy field. If *rdb_file* is an empty string, an empty uninitialized rl_DielectricPOD is created and the init method must be called to initialize the object.

Definition at line 73 of file rl_DielectricPOD_rdb.cc.

References `init()`.

6.1.3 Member Function Documentation

6.1.3.1 void rl_DielectricPOD_rdb::init (char const *rdb_file*[])

Initializer.

Parameters:

rdb_file name of the /rdb file to be read. *rdb_file* specifies the name of an /rdb table; energy, alpha, and gamma are read in from the specified /rdb table. The array is sorted on the energy field.

Definition at line 78 of file rl_DielectricPOD_rdb.cc.

Referenced by `rl_DielectricPOD_rdb()`.

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

- rl_DielectricPOD_rdb.h
- rl_DielectricPOD_rdb.cc

6.2 rl_Multilayer_rdb Class Reference

Public Member Functions

- [~rl_Multilayer_rdb](#) ()
- [rl_Multilayer_rdb](#) (char const rdb_file[], rl_Traits::EInterpMode interp_mode)
- void [init_from_rdb](#) (char const rdb_file[], rl_Traits::EInterpMode interp_mode)

6.2.1 Detailed Description

Definition at line 74 of file rl_Multilayer_rdb.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 rl_Multilayer_rdb::~~rl_Multilayer_rdb ()

Destructor.

Definition at line 60 of file rl_Multilayer_rdb.cc.

6.2.2.2 rl_Multilayer_rdb::rl_Multilayer_rdb (char const *rdb_file*[], rl_Traits::EInterpMode *interp_mode*)

Constructor.

Parameters:

- rdb_file* name of the /rdb file to be read.
- interp_mode* interpolation type for the optical constants.

If rdb_file is a nonempty string, read in energy, alpha, and gamma from the specified /rdb table. The array is sorted on the energy field. If rdb_file is an empty string, init_from_rdb must be called to initialize the object.

The interp_mode argument specifies the form of energy/optical constant interpolation (see rl_Traits in rl_raylib).

Definition at line 67 of file rl_Multilayer_rdb.cc.

References [init_from_rdb\(\)](#).

6.2.3 Member Function Documentation

6.2.3.1 void rl_Multilayer_rdb::init_from_rdb (char const *rdb_file*[], rl_Traits::EInterpMode *interp_mode*)

Initializer.

Parameters:

rdb_file name of the /rdb file to be read.

interp_mode interpolation type for the optical constants.

rdb_file specifies the name of an /rdb table; energy, alpha, and gamma are read in from the specified /rdb table. The array is sorted on the energy field.

The *interp_mode* argument specifies the form of energy/optical constant interpolation (see rl_Traits in rl_raylib).

Referenced by rl_Multilayer_rdb().

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

- rl_Multilayer_rdb.h
- rl_Multilayer_rdb.cc

7 rl_raysuplib Page Documentation

7.1 Basic Interface

7.1.1 A Basic Ray Class

The rl_RayLib 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.

7.2 Purpose

7.2.1 Purpose

The `rl_RayLib` library consists of a set of C++ classes for manipulating rays, including the effects of multilayer reflectivity.

7.2.1.1 classes The ray classes include:

- `rl_BasicRay`: a stripped-down ray consisting of position, direction, energy, and an id number.
- `rl_Ray`: adds ray polarization information to an `rl_BasicRay`. The rays can be translated/rotated from a standard coordinate system (STD) to a “body center system” (BCS), and de-rotated/de-translated from the BCS system back to the STD system. Given a surface normal, the ray direction can be reflected to a new direction. This replicates much of the transformation functionality of the OSAC library. the intention is to eventually implement the rest of the OSAC functionality. The main missing component at this point is the evaluation of the distorted surface interception; most of the rest of the functionality (including [multilayer] reflectivity is in `rl_RayLib`.

7.2.1.2 classes The reflectivity classes include a number of components:

- `rl_DielectricData` encapsulates an array of energy bins providing the dielectric decrement information and methods to evaluate (interpolate) the decrements at a requested energy. The array is built on a helper “Plain Ol’ Data” (POD) struct `rl_DielectricPOD` which provides the dielectric decrement at a specific energy.
- `rl_DielectricLayer` encapsulates the information about the interaction of a photon with a single dielectric layer, including the layer thickness, dielectric decrements given the photon energy, the component of the photon wave vector perpendicular to the layer, and various reflection and transmission coefficients, and surface “roughness” information. The layer can be a “substrate” (in which case the layer is considered as semi-infinite), vacuum, or a dielectric layer. These are mediated by helper “Plain Ol’ Data” (POD) structs and classes: `rl_ReflectionCoefPOD`, `rl_TransmissionCoefPOD`, `rl_ReflectionCoefPOD`.

- `rl_Multilayer` encapsulates a stack of `rl_DielectricLayer`'s. Given the energy, sine of the graze angle, the multilayer reflectivity can be evaluated.
- `rl_MultilayerSurface` adds surface normal information to an `rl_Multilayer`. Given an `rl_Ray`, `rl_MultilayerSurface` can evaluate the reflectivity for the surface. This is also where hooks for surface interception and scattering could be placed.

7.2.1.3 classes. The `rl_RaySupLib` library includes a number of classes providing i/o support: interfacing with the BPipe transport, and reading initialization information from rdb database tables. These include:

- `rl_BPipe` sets up and handles the BPipe transport
- `rl_DielectricPOD_rdb` handles initializing `rl_DielectricPOD` from rdb tables. The rdb table contains a list of energies and the corresponding complex dielectric decrements.
- `rl_Multilayer_rdb` handles initializing an `rl_Multilayer` from an rdb table. The rdb table provides the information on each layer (name, thickness, roughness type) and the name of an rdb file specifying the dielectric decrement information.

Index

- ~rl_DielectricPOD_rdb
 - rl_DielectricPOD_rdb, [3](#)
- ~rl_Multilayer_rdb
 - rl_Multilayer_rdb, [5](#)
- init
 - rl_DielectricPOD_rdb, [4](#)
- init_from_rdb
 - rl_Multilayer_rdb, [5](#)
- rl_DielectricPOD_rdb, [3](#)
 - rl_DielectricPOD_rdb, [3](#)
- rl_DielectricPOD_rdb
 - ~rl_DielectricPOD_rdb, [3](#)
 - init, [4](#)
 - rl_DielectricPOD_rdb, [3](#)
- rl_Multilayer_rdb, [4](#)
 - ~rl_Multilayer_rdb, [5](#)
 - init_from_rdb, [5](#)
 - rl_Multilayer_rdb, [5](#)
- rl_raysuplib/ Directory Reference, [2](#)