NAME

trace-nest - ray trace a nest of shells, hey!

SYNOPSIS

trace-nest *options*

ARGUMENTS

trace-nest uses an IRAF-compatible parameter interface. See the section on *Setup* below. The available parameters are:

tag

A prefix to be used on all intermediate files created. There are lots of intermediate files; see the section on *Intermediate Files*.

src

The location of a **raygen** compatible source script. If it is the string `default`, the value of the `source_spec` keyword in the **trace-nest** configuration file is used.

srcpars

Extra parameters to be passed to the source script. See the documentation for the source script for information on which parameters are available. script accepts

These are passed directly to **raygen** via the `source_override` parameter.

shells

The shells that should be raytraced and merged. The value may be take one of the following values:

`all`

All of the shells defined in the mirror geometry database will be raytraced.

`active`

All of the shells defined in the mirror geometry database which are marked as `active` will be raytraced. If the geometry database does not provide the `active` attribute, all of the shells are raytraced.

list

A comma delimited list of individual shell ids or ranges (*min-max*). For example:

```
shells=1-10,23,45
```

output

The output stream to which to write the rays. It may be a filename, or the string `stdout`, in which case rays will be written to the standard output stream. If it is the string `default`, a file name will be created by appending the **output_fmt** to the **tag** (with an intervening period).

output_fmt

The output format of the rays. May be one of `fr`, `bp`, `bpipe`, `rdb`, or a `fits` variant. See *Output Formats* for more information.

output_coord

The output coordinate system of the rays. May be one of `osac`, `hrma`, `xrcf`.

output_fields

Which data fields to output for each ray. The value may be one of

`all`

A rather large amount of information.

<field names>

A comma delimited list of field names to output. Field names may be prefixed with `-`, indicating that they are to be *removed* from the list of output fields. If the only fields specified are those to be removed, the initial output list contains all of the fields in the data.

The field name `min` is an alias for specifying the following fields:

```
position direction weight energy time
```

The order of additive and subtractive fields is unimportant; all additive fields are inserted into the list before the subtractive fields are removed.

seed1

The first seed for the random number generator. It must be in the range [1,2147483562].

seed2

The second seed for the random number generator. It must be in the range [1,214748339]

block

The random number block to start at. It must be in the range [0,1048575].

block_inc

The spacing between random number blocks for each random process. 100 is a good number.

tstart

The start time of the observation in seconds. If less than zero and jitter is turned on, the start of the valid jitter time range is used.

limit

The numerical value of the limit at which to stop generating rays at the telescope entrance aperture. The number of rays which reach the focal plane are typically lower than this. The **limit_type** parameter specifies the units for this value.

limit may be either a floating point number, in which it is used for all shells, or the name of a file containing *limit* values for all shells. The file must be an **RDB** formatted file with columns *shell* and *limit*.

If `limit_type` is a unit of time, this is added to the start time (see `tstart`) to determine the stop time of the simulation. If jitter is on and this is set to `0`, then the stop time is set equal to the end of the valid jitter time range.

limit_type

The units of the limit at which to stop generating rays.

`ksec`

kiloseconds of observation time

`sec`

seconds of observation time

`r/mm2`

a ray density at the entrance aperture in rays / mm^2

`r/cm2`

a ray density at the entrance aperture in rays / cm^2

ray_dist

The ray distribution at the entrance aperture. It may be one of `random` or `ringspoke`. *Currently,* `ringspoke` *is ignored.*

nrings

The number of rings to use if the **ray_dist** parameter is `ringspoke`.

nspokes

The number of spokes to use if the **ray_dist** parameter is `ringspoke`.

focus

A boolean parameter indicating that the focus of the system is to be determined. See the *Focus* section for more details.

z

The position along the *Z* (optical) axis at which to leave the rays. This is in the OSAC coordinate system.

z may be either a floating point number, in which it is used for all shells, or the name of a file containing *Z* values for all shells. The file must be an **RDB** formatted file with columns *shell* and *z*. The *Z* value for the combined shells should have a *shell* value of `99999`.

tally

If non-zero, a tally of rays will be written to the standard error stream every `tally` rays. This is useful if you're wondering why it's taking so long to run the raytrace. This tallies the number of rays which make it out of the nest, after all of the post-optic apertures.

throttle

If non-zero, this specifies the number of rays which should be output after the shells are merged. Rays are first eliminated based upon their probability of reflection, and then only the requested number are passed through through (provided enough are available). The number may be varied in a Poisson fashion by setting **throttle_poisson**.

If negative, rays will be eliminated by their probability of reflection, but no total limit to their number will be set.

A side effect of this option is that all rays will have a weight of one.

throttle_poisson

If true, and if **throttle** is non-zero, then the number specified by **throttle** is treated as the mean of a Poisson distribution, and the number of rays actually output will be drawn from that distribution.

clean

Which intermediate files to delete. It may be `none`, in which case nothing is removed, `rays` in which case intermediate ray files are removed, or `all` in which everything is

config_dir removed.

The directory containing the **trace-nest** configuration file.

config_db

The name of the configuration file which provides the details of the HRMA configuration. See the *Configuration File* section below.

version

Print out the version information and exit.

help

Print out this message and exit.

debug

A comma separated list of debugging options. See the *Debugging* section for more information.

DESCRIPTION

trace-nest raytraces a nest of Wolter type I X-ray telescope shells with various apertures and baffles. It was designed around the AXAF HRMA, but may be used for other systems. It works by using **trace-shell** to raytrace each shell, finally merging them into a single file. It traces each shell sequentially, storing each shells' rays on disk; it ends up using twice as much disk space as you think. It'll clean up after itself, though (see the **clean** parameter).

trace-nest uses a variety of programs to accomplish the raytrace. To see the actual raytrace command pipeline, use the **debug** `pcomm` option.

Setup

trace-nest uses an IRAF compatible parameter interface. Because it calls many other programs, you will actually need to have parameter files for all of them handy.

To simplify things, there is a command (**trace-nest_setup**) which creates copies in the current directory of the all of the required parameter files.

Configuration File

The **trace-nest** configuration file (specified by the **config_dir** and **config_db** parameters) describes the telescope configuration. Before you create your own, look at */proj/axaf/simul/databases/ts_config/00Index.html* and see if there's one to suit your fancy. Also, note that **trace-nest** can only use configuration files with a `.cnf` suffix. For more information on raytrace configuration files, etc. see *ts_config*.

Intermediate Files

trace-nest produces a few intermediate files. Each file is given a prefix which consists of the value of the **tag** parameter followed the shell number. For the final set of merged rays, the shell number is left off. For example, if **tag** is `foo`, you'll get files of

```
foo_1.totwt-in foo_2.totwt-in
```

and

```
foo.totwt_in
```

If **trace-nest** is only tracing one shell, it doesn't include the shell number in the file name.

tag.bp

The rays for the particular shell, in **bpipe** format.

tag.gi

This is a rather arcanely formatted file required by **SAOdrat**. It's not of much general interest.

tag.totwt-in

This file contains the number and weight of the rays at the entrance aperture. It is produced by **tot_wt**.

tag.totwt-out

This file contains the number and weight of the rays which have made it through the entire configuration. It is produced by **tot_wt**.

tag.totwt-throttle

This file contains the number and weight of the rays which have made it through the entire configuration, after getting throttled. It is produced by **tot_wt**.

tag.focus.lis

This is created during a focus run by **saofocus**.

Output Formats

trace-nest outputs one of the following formats, specified by the **output_fmt** parameter:

`fr`

The `fr` format has no header. Each ray is in a `fullray` structure. See */proj/axaf/simul/include/fullray.h* for the formats of the ray structure.

`bp` or `bpipe`

The rays are in `bpipe` format. See the **bpipe** documentation for more information on this.

`rdb`

The rays are written as an RDB table.

a `fits` variant

Various FITS formatted outputs may be specified. In all cases the output must be to a file.

`fits` or `fits-std`

The rays are written in the uncommon and seldom used AXAF Photon FITS standard.

`fits-events`

The rays are written in the much more common "events" format. It differs from the AXAF FITS Photon Standard in that the binary table is named EVENTS, the `rt_` prefix is removed from the column names, and the energy column is named `energy` and is in units of eV. Most X-ray Astronomy software uses this convention.

Focus

If you wish to determine where the focal point for a given configuration is, set the **focus** parameter to `yes` . However, because of bad interactions between the focus algorithm and wildly scattered rays, micro-roughness induced ray scattering and ghost-ray tracking is turned off when focussing. Additionally,

the source is forced to be the default source specified by the configuration file, which should be a point source. The default source requires at least one parameter, namely `energy`. The focus procedure is carried out by **saofocus** which leaves its results in a file called `tag.focus.lis`, (where you've specified **tag**). This file is pretty arcane; generally to extract the focus from there, run the script *getfocus* on it:

```
getfocus tag.focus.lis
```

which will write out the focal position (in OSAC coordinates) to the UNIX standard output stream. Note that you'll get the focus of all of the individual shells as well as the nest (unless you set **clean** to `all`).

All of the shells' focal distances, including the focal distance for the combined shells, is written to the file `tag.focus.rdb`. The combined shells are assigned a shell number of `99999`.

Debugging

There are several **debug** options available:

`pcomm`

Print out the raytrace command before executing it. This gives you some idea of which programs are running and what their inputs are.

`noexec`

Generate the raytrace command and any required intermediate files, but do not execute it. Most useful with the **pcomm debug** option.

`reuse`

Reuse the raytrace output from a previous **identical** run to regenerate the summary information. `noexec` must *not* be specified simultaneously. The raytrace parameters should be identical except for the addition of this flag.

`noproject`

Do not project the rays to the value specified by the **z** parameter. This is a temporary kludge, and will probably not survive into the next version of **trace-nest**.

`noghosts`

Ghost rays will not be propagated through the system.

`saveblock`

The next unused random number block is written to *tag.block*.

SEE ALSO

trace-shell, **ts_config**

COPYRIGHT AND LICENSE

Author

Diab Jerius (djerius@cfa.harvard.edu)