theia

API Documentation

July 24, 2017

Contents

1 Package theia

This is theia, a Python package for Gaussian ray tracing in 3D optical setups.

Version: 0.1.2

Author: Raphaël Duque

1.1 Modules

- **helpers**: This is the helpers sub-package of theia.
  (Section 2, p. 4)
  - **core**: Defines some additional spice for theia.
    (Section 3, p. 5)
  - **geometry**: Geometry module for theia.
    (Section 4, p. 6)
  - **interaction**: Module to define interaction functions for theia.
    (Section 5, p. 9)
  - **settings**: Module to initiate all global variables for theia.
    (Section 6, p. 10)
  - **tools**: Defines some generic functions for theia.
    (Section 7, p. 11)
  - **units**: Various units for theia.
    (Section 8, p. 14)
- **main**: Main module of theia, defines the main function.
  (Section 9, p. 15)
- **optics**: This is the optics sub-package of theia.
  (Section 10, p. 16)
  - **beam**: Defines the GaussianBeam class for theia.
    (Section 11, p. 17)
  - **beamdump**: Defines the BeamDump class for theia.
    (Section 12, p. 21)
  - **component**: Defines the SetupComponent class for theia.
    (Section 13, p. 24)
  - **ghost**: Defines the Ghost class for theia.
    (Section 14, p. 27)
  - **lens**: Defines the Lens class for theia.
    (Section 15, p. 30)
  - **mirror**: Defines the Mirror class for theia.
    (Section 16, p. 34)
  - **optic**: Defines the Optic class for theia.
    (Section 17, p. 39)
  - **thicklens**: Defines the ThickLens class for theia.
    (Section 18, p. 42)
  - **thinlens**: Defines the ThinLens class for theia.
    (Section 19, p. 45)
- **rendering**: This is the rendering sub-package of theia.

(Section 20, p. 48)

- **features**: Features module or theia, to represent objects as FreeCAD Python features.
  (Section 21, p. 49)
- **shapes**: Shapes module for theia, provides shape-calculating for 3D rendering.
  (Section 22, p. 54)
- **writer**: Writer module for theia, to write CAD content to files.
  (Section 23, p. 55)

- **running**: This is the running sub-package of theia.
  (Section 24, p. 56)
  - **parser**: Module for the parsing on input data from .tia file.
    (Section 25, p. 57)
  - **simulation**: Defines the Simulation class for theia.
    (Section 26, p. 59)
- **tree**: This is the tree sub-package of theia.
  (Section 27, p. 62)
  - **beamtree**: Defines the BeamTree class for theia.
    (Section 28, p. 63)

2 Package theia.helpers

This is the helpers sub-package of theia.

It provides it provides all sorts of generic functions for theia.

Version: 0.1.2

Author: Raphaël Duque

Copyright: Copyright 2017, Raphaël Duque

License: GNU GPLv3+

2.1 Modules

- **core**: Defines some additional spice for theia.
  (Section 3, p. 5)
- **geometry**: Geometry module for theia.
  (Section 4, p. 6)
- **interaction**: Module to define interaction functions for theia.
  (Section 5, p. 9)
- **settings**: Module to initiate all global variables for theia.
  (Section 6, p. 10)
- **tools**: Defines some generic functions for theia.
  (Section 7, p. 11)
- **units**: Various units for theia.
  (Section 8, p. 14)

3 Module theia.helpers.core

Defines some additional spice for theia.

3.1 Functions

gbeamInit(*menu*)

Pick in the menu.

hang()

The whole hangman game, from welcome to exit.

magazzu()

pong()

pendu()

3.2 Variables

Name	Description
__package__	**Value:** `'theia.helpers'`

4 Module theia.helpers.geometry

Geometry module for theia.

4.1 Functions

refrAngle(*theta*, *n1*, *n2*)

Returns the refraction angle at n1/n2 interface for incoming theta.

May raise a TotalReflectionError.

linePlaneInter(*pos*, *dirV*, *planeC*, *normV*, *diameter*)

```
Computes the intersection between a line and a plane.

pos: position of the begining of the line. [3D vector]
dirV: directing vector of the line. [3D vector]
planeC: position of the center of the plane. [3D vector]
normV: vector normal to the plane. [3D vector]
diameter: diameter of the plane.

Returns a dictionary with keys:
    'isHit': whether of not the plane is hit. [boolean]
    'distance': geometrical distance from line origin to intersection point.
        [float]
    'intersection point': position of intersection point. [3D vector]
```

lineSurfInter(*pos*, *dirV*, *chordC*, *chordNorm*, *kurv*, *diameter*)

```
Computes the intersection between a line and a spherical surface.

The spherical surface is supposed to have a cylindrical symmetry around
    the vector normal to the 'chord', ie the plane which undertends
    the surface.

Note: the normal vector always looks to the center of the sphere and the
    surface is supposed to occupy less than a semi-sphere

pos: position of the begingin of the line. [3D vector]
dirV: direction of the line. [3D vector]
chordC: position of the center of the 'chord'. [3D vector]
chordNorm: normal vector the the chord in its center. [3D vector]
kurv: curvature (1/ROC) of the surface. [float]
diameter: diameter of the surface. [float]

Returns a dictionary with keys:
    'is Hit': whether the surface is hit or not. [boolean]
    'distance': distance to the intersection point from pos. [float]
    'intersection point': position of intersection point. [3D vector]
```

lineCylInter(*pos*, *dirV*, *faceC*, *normV*, *thickness*, *diameter*)

```
Computes the intersection of a line and a cylinder in 3D space.

The cylinder is specified by a disk of center faceC, an outgoing normal
normV, a thickness (thus behind the normal) and a diameter.

pos: origin of the line. [3D vector]
dirV: directing vector of the line. [3D vector]
faceC: center of the face of the cylinder where lies the normal vector.
    [3D vector]
normV: normal vector to this face (outgoing). [3D vector]
thickness: thickness of the cylinder (counted from faceC and behind normV)
    [float]
diameter: of the cylinder. [float]

Returns a dictionary with keys:
    'isHit': whether of not. [boolean]
    'distance': geometrical distance of the intersection point from pos.
        [float]
    'intersection point': point of intersection. [3D vector]
```

newDir(*inc*, *nor*, *n1*, *n2*)

```
Computes the refl and refr directions produced by inc at interface n1/n2.

inc: director vector of incoming beam. [3D vector]
nor: normal to the interface at the intersection point. [3D vector]
n1: refractive index of the first medium. [float]
n2: idem.

Returns a dictionary with keys:
    'r': normalized direction of reflected beam. [3D vector]
    't': normalized direction of refracted beam. [3D vector]
    'TR': was there total reflection?. [boolean]

Note: if total reflection then refr is None.
```

rotMatrix(*a*, *b*)

Provides the rotation matrix which maps a (unit) to b (unit).

a,b: unit 3D vectors. [3D np.arrays]

Returns an np.array such that np.dot(M,a) == b.

basis(*a*)

Returns two vectors u and v such that (a, u, v) is a direct ON basis.

rectToSph(*array*)

Returns the spherical coordinates of the unitary vector given by array.

array: 3D vector (unitary). [float]

Returns the theta and phi angles in radians with theta in [0, pi] and phi in [-pi, pi]

4.2 Variables

Name	Description
__package__	**Value:** `'theia.helpers'`

5 Module theia.helpers.interaction

Module to define interaction functions for theia.

5.1 Variables

Name	Description
usage	**Value:** `'Usage: theia [options] FNAME\n\nArguments:\n FNAME\t\t ...`
lhelp	**Value:** `'specify the FreeCAD library location. If none is specifi...`
welcome	**Value:** `...`
errorRecursion	**Value:** `'\n\nIt looks like you reached the maximum recursion dept...`
errorAtSpecifiedLocation	**Value:** `'theia: Error: The FreeCAD library was not found at the s...`
errorWhereIs	**Value:** `'theia: Error: Unix command \'whereis freecad\' did not y...`
errorUnknown	**Value:** `'theia: Error: %s was used as the source directory for th...`
__package__	**Value:** `None`

6 Module theia.helpers.settings

Module to initiate all global variables for theia.

6.1 Functions

init(*dic*)

Initiate globals with dictionary.

dic: dictionary holding values for globals. [dictionary]

6.2 Variables

Name	Description
__package__	**Value:** `None`

7 Module theia.helpers.tools

Defines some generic functions for theia.

7.1 Functions

timer(*func*)

Decorator function to log execution time of other functions.

formatter(*stringList*)

Returns a formatted version of the text formed by the list of lines.

7.2 Variables

Name	Description
__package__	**Value:** `'theia.helpers'`

7.3 Class TotalReflectionError

object

exceptions.BaseException

exceptions.Exception

theia.helpers.tools.TotalReflectionError

TotalReflectionError class.

Is raised when an interaction results in total reflection.

=== Attributes === Message: exception message. [string]

7.3.1 Methods

__init__(*self*, *message*)

TotalReflectionError exception initializer.

Overrides: object.__init__

__str__(*self*)

Printing error function.

Overrides: object.__str__

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

7.3.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

7.4 Class InputError

object
exceptions.BaseException
exceptions.Exception
theia.helpers.tools.InputError

InputError class.

Is raised when the input .tia file parsing to input data failed.

=== Attributes === Message: exception message. [string]

7.4.1 Methods

__init__(*self*, *message*)

InputError exception initializer.

Overrides: object.__init__

__str__(*self*)

Printing error function

Overrides: object.__str__

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

7.4.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object __class__	

8 Module theia.helpers.units

Various units for theia.

8.1 Variables

Name	Description
km	**Value:** `1000.0`
m	**Value:** `1.0`
cm	**Value:** `0.01`
mm	**Value:** `0.001`
um	**Value:** `1e-06`
nm	**Value:** `1e-09`
kW	**Value:** `1000.0`
W	**Value:** `1.0`
mW	**Value:** `0.001`
uW	**Value:** `1e-06`
nW	**Value:** `1e-09`
THz	**Value:** `1e+12`
GHz	**Value:** `1000000000.0`
MHz	**Value:** `1000000.0`
kHz	**Value:** `1000.0`
Hz	**Value:** `1.0`
mHz	**Value:** `0.001`
uHz	**Value:** `1e-06`
ppm	**Value:** `1e-06`
rad	**Value:** `1.0`
deg	**Value:** `0.0174532925199`
pi	**Value:** `3.14159265359`
__package__	**Value:** `None`

9 Module theia.main

Main module of theia, defines the main function.

9.1 Functions

main(*options*, *args*)

Main function of theia.

9.2 Variables

Name	Description
__package__	**Value:** `'theia'`

10 Package theia.optics

This is the optics sub-package of theia.

It provides the necessary classes and functions in order to calculate the gaussian beams of the setup.

Version: 0.1.2

Author: Raphaël Duque

10.1 Modules

- **beam**: Defines the GaussianBeam class for theia.
  (Section 11, p. 17)
- **beamdump**: Defines the BeamDump class for theia.
  (Section 12, p. 21)
- **component**: Defines the SetupComponent class for theia.
  (Section 13, p. 24)
- **ghost**: Defines the Ghost class for theia.
  (Section 14, p. 27)
- **lens**: Defines the Lens class for theia.
  (Section 15, p. 30)
- **mirror**: Defines the Mirror class for theia.
  (Section 16, p. 34)
- **optic**: Defines the Optic class for theia.
  (Section 17, p. 39)
- **thicklens**: Defines the ThickLens class for theia.
  (Section 18, p. 42)
- **thinlens**: Defines the ThinLens class for theia.
  (Section 19, p. 45)

11 Module theia.optics.beam

Defines the GaussianBeam class for theia.

11.1 Functions

userGaussianBeam(*Wx*=`0.001`, *Wy*=`0.001`, *WDistx*=`0.0`, *WDisty*=`0.0`, *Wl*=`1.064e-06`, *P*=`1.0`, *X*=`0.0`, *Y*=`0.0`, *Z*=`0.0`, *Theta*=`1.57079632679`, *Phi*=`0.0`, *Alpha*=`0.0`, *Ref*=`None`)

Constructor used for user inputed beams, separated from the class initializer because the internal state of a beam is very different from the input of this user-defined beam.

Input parameters are processed to make arguments for the class contructor and then the corresponding beam is returned.

11.2 Variables

Name	Description
__package__	**Value:** `'theia.optics'`

11.3 Class GaussianBeam

object

theia.optics.beam.GaussianBeam

```
GaussianBeam class.

This class represents general astigmatic Gaussian beams in 3D space.
These are the objects that are intended to interact with the optical
components during the ray tracing and that are rendered in 3D thanks to
FreeCAD.

*=== Attributes ===*
BeamCount: class attribute, counts beams. [integer]
Name: class attribute. [string]
QTens: general astigmatic complex curvature tensor at the origin.
    [np. array of complex]
```

```
N: Refraction index of the medium in which the beam is placed. [float]
Wl: Wave-length in vacuum of the beam (frequency never changes). [float]
P: Power of the beam. [float]
Pos: Position in 3D space of the origin of the beam. [3D vector]
Dir: Normalized direction in 3D space of the beam axis. [3D vector]
U: A tuple of unitary vectors which along with Dir form a direct orthonormal
    basis in which the Q tensor is expressed. [tuple of 3D vectors]
Ref: Reference to the beam. [string]
OptDist: Optical length. [float]
Length: Geometrical length of the beam. [float]
StrayOrder: Number representing the *strayness* of the beam. If the beams
    results from a transmission on a HR surface or a reflection on a AR
    surface, then its StrayOrder is the StrayOrder of the parent beam + 1.
    [integer]
Optic: Ref of optic where the beam departs from (None if laser). [string]
Face: face of the optic where the beam departs from. [string]
```

11.3.1 Methods

__init__(*self*, *Q*, *N*, *Wl*, *P*, *Pos*, *Dir*, *Ux*, *Uy*, *Ref*, *OptDist*, *Length*, *StrayOrder*, *Optic*, *Face*)

Beam initializer.

This is the initializer used internally for beam creation, for user inputed beams, see class method userGaussianBeam.

Returns a Gaussian beam with attributes as the parameters.

Overrides: object.__init__

__str__(*self*)

String representation of the beam, when calling print(beam).

Overrides: object.__str__

lines(*self*)

Returns the list of lines necessary to print the object.

Q(*self*, *d*=`0.0`)

Return the Q tensor at a distance d of origin.

QParam(*self*, *d*=`0.0`)

Compute the complex parameters q1 and q2 and theta of beam.

Returns a dictionary with keys: '1': q1 [complex] '2': q2 [complex] 'theta': theta [float]

ROC(*self*, *dist*=`0.0`)

Return the tuple of ROC of the beam.

waistPos(*self*)

Return the tuple of positions of the waists of the beam along Dir.

rayleigh(*self*)

Return the tuple of Rayleigh ranges of the beam.

width(*self*, *d*=`0.0`)

Return the tuple of beam widths.

waistSize(*self*)

Return a tuple with the waist sizes in x and y.

gouy(*self*, *d*=`0.0`)

Return the tuple of Gouy phases.

translate(*self*, *X*=`0.0`, *Y*=`0.0`, *Z*=`0.0`)

Move the beam to (current position + (X, Y, Z)).

X, Y, Z: components of the translation vector.

No return value.

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

11.3.2 Properties

Name	Description
Inherited from object __class__	

11.3.3 Class Variables

Name	Description
BeamCount	**Value:** `0`
Name	**Value:** `'Beam'`

12 Module theia.optics.beamdump

Defines the BeamDump class for theia.

12.1 Variables

Name	Description
__package__	**Value:** `'theia.optics'`

12.2 Class BeamDump

object

theia.optics.component.SetupComponent

theia.optics.beamdump.BeamDump

```
BeamDump class.

This class represents components on which rays stop. They have cylindrical
symmetry and stop beams on all their faces. They can represent baffles
for example.

*=== Attributes ===*
SetupCount (inherited): class attribute, counts all setup components.
    [integer]
Name: class attribute. [string]
HRCenter (inherited): center of the principal face of the BeamDump in space.
    [3D vector]
HRnorm (inherited): normal unitary vector the this principal face,
    supposed to point outside the media. [3D vector]
Thick (inherited): thickness of the dump, counted in opposite direction to
    HRNorm. [float]
Dia (inherited): diameter of the component. [float]
Ref (inherited): reference string (for keeping track with the lab). [string]
```

12.2.1 Methods

__init__(*self*, *X*=0.0, *Y*=0.0, *Z*=0.0, *Theta*=1.57079632679, *Phi*=0.0, *Ref*=None, *Thickness*=0.02, *Diameter*=0.05)

BeamDump initializer.

Parameters are the attributes.

Returns a BeamDump.

Overrides: object.__init__

lines(*self*)

Return the list of lines needed to print the object.

Overrides: theia.optics.component.SetupComponent.lines

isHit(*self*, *beam*)

```
Determine if a beam hits the BeamDump.

This uses the line***Inter functions from the geometry module to find
characteristics of impact of beams on beamdumps.

beam: incoming beam. [GaussianBeam]

Returns a dictionary with keys:
    'isHit': whether the beam hits the dump. [boolean]
    'intersection point': point in space where it is first hit.
        [3D vector]
    'face': to indicate which face is first hit, can be 'HR', 'AR' or
        'side'. [string]
    'distance': geometrical distance from beam origin to impact. [float]
```

Overrides: theia.optics.component.SetupComponent.isHit

hit(*self*, *beam*, *order*, *threshold*)

```
Compute the refracted and reflected beams after interaction.

BeamDumps always stop beams.

beam: incident beam. [GaussianBeam]
order: maximum strayness of daughter beams, which are not returned if
    their strayness is over this order. [integer]
threshold: idem for the power of the daughter beams. [float]

Returns a dictionary of beams with keys:
    't': None
    'r': None
```

Overrides: theia.optics.component.SetupComponent.hit

Inherited from theia.optics.component.SetupComponent(Section 13.2)

__str__(), translate()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

12.2.2 Properties

Name	Description
Inherited from object __class__	

12.2.3 Class Variables

Name	Description
Name	**Value:** `'BeamDump'`
__abstractmethods__	**Value:** `frozenset([])`
Inherited from theia.optics.component.SetupComponent (Section 13.2) SetupCount	

13 Module theia.optics.component

Defines the SetupComponent class for theia.

13.1 Variables

Name	**Description**
__package__	**Value:** `'theia.optics'`

13.2 Class SetupComponent

object ─┐
theia.optics.component.SetupComponent

Known Subclasses: theia.optics.beamdump.BeamDump, theia.optics.optic.Optic, theia.optics.ghost.Gho

```
SetupComponent class.

This is an Abstract Base Class for all the components (optical or not) of
the setup. Its methods may be implemented in daughter classes.

*=== Attributes ===*
SetupCount: class attribute, counts setup components. [integer]
HRCenter: center of the principal face of the component in space.
    [3D vector]
HRNorm: normal unitary vector the this principal face, supposed to point
    outside the media. [3D vector]
Thick: thickness of the component, counted in opposite direction to
    HRNorm. [float]
Dia: diameter of the component. [float]
Name: name of the component. [string]
Ref: reference string (for keeping track with the lab). [string]
```

13.2.1 Methods

__init__(*self*, *HRCenter*, *HRNorm*, *Ref*, *Thickness*, *Diameter*)

SetupComponent initializer.

Parameters are the attributes of the object to construct.

Returns a setupComponent.

Overrides: object.__init__

__str__(*self*)

String representation of the component, when calling print(object).

Overrides: object.__str__

hit(*self*, *beam*, *order*, *threshold*)

Compute the refracted and reflected beams after interaction.

Abstract (pure virtual) method.

isHit(*self*, *beam*)

Method to determine if component is hit by a beam.

Abstract (pure virtual) method.

lines(*self*)

Method to return the list of strings to __str__.

Abstract (pure virtual) method.

translate(*self*, *X*=`0.0`, *Y*=`0.0`, *Z*=`0.0`)

Move the component to (current position + (X, Y, Z)).

This version only takes care of the HRCenter, version of sub classes take care of ARCenter if relevant.

X, Y, Z: components of the translation vector.

No return value.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

13.2.2 Properties

Name	Description
Inherited from object ___class___	

13.2.3 Class Variables

Name	Description
Name	**Value:** `'SetupComponent'`
SetupCount	**Value:** `0`
___abstractmethods___	**Value:** `frozenset(['hit', 'isHit', 'lines'])`

14 Module theia.optics.ghost

Defines the Ghost class for theia.

14.1 Variables

Name	Description
__package__	**Value:** `'theia.optics'`

14.2 Class Ghost

object
theia.optics.component.SetupComponent
theia.optics.ghost.Ghost

```
Ghost class.

This class represents surfaces which don't interact with the beams. They
just transmit the same beam, and may be useful to monitor the beams on their
way, without having to calculate the Q yourself if you're looking for the
Q at another place than the origin of the beam.

Ghost surfaces basically have a null thickness and transmit the beams.

*==== Attributes ===*
SetupCount (inherited): class attribute, counts all setup components.
    [integer]
Name: class attribute. [string]
HRCenter (inherited): center of the principal face of the Ghost in space.
    [3D vector]
HRnorm (inherited): normal unitary vector the this principal face,
    supposed to point outside the media. [3D vector]
Thick (inherited): thickness of the dump, counted in opposite direction to
    HRNorm. [float]
Dia (inherited): diameter of the component. [float]
Ref (inherited): reference string (for keeping track with the lab). [string]
```

14.2.1 Methods

__init__(*self*, *X*=`0.0`, *Y*=`0.0`, *Z*=`0.0`, *Theta*=`1.57079632679`, *Phi*=`0.0`, *Ref*=`None`, *Diameter*=`0.05`)

Ghost initializer.

Parameters are the attributes.

Returns a Ghost.

Overrides: object.__init__

lines(*self*)

Return the list of lines needed to print the object.

Overrides: theia.optics.component.SetupComponent.lines

isHit(*self*, *beam*)

```
Determine if a beam hits the Ghost surface.

This uses the linePlaneInter function from the geometry module to find
characteristics of impact of beams on ghost surfaces.

beam: incoming beam. [GaussianBeam]

Returns a dictionary with keys:
    'isHit': whether the beam hits the dump. [boolean]
    'intersection point': point in space where it is first hit.
        [3D vector]
    'face': to indicate which face is first hit, can be 'HR', 'AR' or
        'side'. [string]
    'distance': geometrical distance from beam origin to impact. [float]
```

Overrides: theia.optics.component.SetupComponent.isHit

hit(*self*, *beam*, *order*, *threshold*)

```
Return the beam simply transmitted by the ghost surface.

beam: incident beam. [GaussianBeam]
order: maximum strayness of daughter beams, which are not returned if
    their strayness is over this order. [integer]
threshold: idem for the power of the daughter beams. [float]

Returns a dictionary of beams with keys:
    't': Gaussian beam which is the continuity of the incident beam.
```

Overrides: theia.optics.component.SetupComponent.hit

Inherited from theia.optics.component.SetupComponent(Section 13.2)

__str__(), translate()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

14.2.2 Properties

Name	Description
Inherited from object __class__	

14.2.3 Class Variables

Name	Description
Name	**Value:** `'Ghost'`
__abstractmethods__	**Value:** `frozenset([])`
Inherited from theia.optics.component.SetupComponent (Section 13.2) SetupCount	

15 Module theia.optics.lens

Defines the Lens class for theia.

15.1 Variables

Name	Description
__package__	**Value:** `'theia.optics'`

15.2 Class Lens

object

theia.optics.component.SetupComponent

theia.optics.optic.Optic

theia.optics.lens.Lens

Known Subclasses: theia.optics.thicklens.ThickLens, theia.optics.thinlens.ThinLens

```
Lens class.

This class is a base class for lenses. It implements the hit and hitActive
methods for all lenses.

*=== Attributes ===*
SetupCount (inherited): class attribute, counts all setup components.
    [integer]
OptCount (inherited): class attribute, counts optical components. [integer]
HRCenter (inherited): center of the 'chord' of the HR surface. [3D vector]
HRNorm (inherited): unitary normal to the 'chord' of the HR (always pointing
    towards the outside of the component). [3D vector]
Thick (inherited): thickness of the optic, counted in opposite direction to
    HRNorm. [float]
Dia (inherited): diameter of the component. [float]
Name (inherited): name of the component. [string]
Ref (inherited): reference string (for keeping track with the lab). [string]
ARCenter (inherited): center of the 'chord' of the AR surface. [3D vector]
ARNorm (inherited): unitary normal to the 'chord' of the AR (always pointing
    towards the outside of the component). [3D vector]
N (inherited): refraction index of the material. [float]
HRK, ARK (inherited): curvature of the HR, AR surfaces. [float]
HRr, HRt, ARr, ARt (inherited): power reflectance and transmission
    coefficients of the HR and AR surfaces. [float]
KeepI (inherited): whether of not to keep data of rays for interference
    calculations on the HR. [boolean]

**Note**: the curvature of any surface is positive for a concave surface
(coating inside the sphere).
Thus kurv*HRNorm/|kurv| always points to the center
of the sphere of the surface, as is the convention for the lineSurfInter of
geometry module. Same for AR.

*******     HRK > 0 and ARK > 0     *******           HRK > 0 and ARK < 0
 *****                               ********         and |ARK| > |HRK|
 H***A                               H*********A
 *****                               ********
*******                             *******
```

15.2.1 Methods

isHit(*self*, *beam*)

```
Determine if a beam hits the Lens.

This is a generic function for all lenses, using their geometrical
attributes. This uses the line***Inter functions from the geometry
module to find characteristics of impact of beams on lenses.

beam: incoming beam. [GaussianBeam]

Returns a dictionary with keys:
    'isHit': whether the beam hits the optic. [boolean]
    'intersection point': point in space where it is first hit.
            [3D vector]
    'face': to indicate which face is first hit, can be 'HR', 'AR' or
        'Side'. [string]
    'distance': geometrical distance from beam origin to impact. [float]
```

Overrides: theia.optics.component.SetupComponent.isHit

hit(*self*, *beam*, *order*, *threshold*)

```
Compute the refracted and reflected beams after interaction.

This function is valid for all types of lenses.
The beams returned are those selected after the order and threshold
criterion.

beam: incident beam. [GaussianBeam]
order: maximum strayness of daughter beams, whixh are not returned if
    their strayness is over this order. [integer]
threshold: idem for the power of the daughter beams. [float]

Returns a dictionary of beams with keys:
    't': refracted beam. [GaussianBeam]
    'r': reflected beam. [GaussianBeam]
```

Overrides: theia.optics.component.SetupComponent.hit

hitActive(*self*, *beam*, *point*, *faceTag*, *order*, *threshold*)

```
Compute the daughter beams after interaction on HR or AR at point.

AR andHr are the 'active' surfaces of the lens.
This function is valid for all types of lenses.

beam: incident beam. [GaussianBeam]
point: point in space of interaction. [3D vector]
faceTag: either 'AR' or 'HR' depending on the face. [string]
order: maximum strayness of daughter beams, whixh are not returned if
    their strayness is over this order. [integer]
threshold: idem for the power of the daughter beams. [float]

Returns a dictionary of beams with keys:
    't': refracted beam. [GaussianBeam]
    'r': reflected beam. [GaussianBeam]
```

Inherited from theia.optics.optic.Optic(Section 17.2)

__init__(), apexes(), collision(), geoCheck(), hitSide(), translate()

Inherited from theia.optics.component.SetupComponent(Section 13.2)

__str__(), lines()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

15.2.2 Properties

Name	Description
Inherited from object __class__	

15.2.3 Class Variables

Name	Description
__abstractmethods__	**Value:** `frozenset(['lines'])`
Inherited from theia.optics.optic.Optic (Section 17.2) Name, OptCount	
Inherited from theia.optics.component.SetupComponent (Section 13.2) SetupCount	

16 Module theia.optics.mirror

Defines the Mirror class for theia.

16.1 Variables

Name	Description
__package__	**Value:** `'theia.optics'`

16.2 Class Mirror

object
theia.optics.component.SetupComponent
theia.optics.optic.Optic
theia.optics.mirror.Mirror

```
Mirror class.

This class represents semi reflective mirrors composed of two faces (HR, AR)
and with a wedge angle. These are the objects with which the beams will
interqct during the ray tracing. Please see the documentation for details
on the geometric construction of these mirrors.

*==== Attributes ====*
SetupCount (inherited): class attribute, counts all setup components.
    [integer]
OptCount (inherited): class attribute, counts optical components. [integer]
Name: class attribute. [string]
HRCenter (inherited): center of the 'chord' of the HR surface. [3D vector]
HRNorm (inherited): unitary normal to the 'chord' of the HR (always pointing
 towards the outside of the component). [3D vector]
Thick (inherited): thickness of the optic, counted in opposite direction to
    HRNorm. [float]
Dia (inherited): diameter of the component. [float]
Ref (inherited): reference string (for keeping track with the lab). [string]
ARCenter (inherited): center of the 'chord' of the AR surface. [3D vector]
ARNorm (inherited): unitary normal to the 'chord' of the AR (always pointing
    towards the outside of the component). [3D vector]
N (inherited): refraction index of the material. [float]
HRK, ARK (inherited): curvature of the HR, AR surfaces. [float]
HRr, HRt, ARr, ARt (inherited): power reflectance and transmission
coefficients of the HR and AR surfaces. [float]
KeepI (inherited): whether of not to keep data of rays for interference
calculations on the HR. [boolean]
Wedge: wedge angle of the mirror, please refer to the documentation for
    detaild on the geometry of mirrors and their implementation here.
    [float]
Alpha: rotation alngle used in the geometrical construction of the mirror
    (see doc, it is the amgle between the projection of Ex on the AR plane
    and the vector from ARCenter to the point where the cylinder and the AR
    face meet). [float]

**Note**: the curvature of any surface is positive for a concave surface
(coating inside the sphere).
Thus kurv*HRNorm/|kurv| always points to the center
of the sphere of the surface, as is the convention for the lineSurfInter of
geometry module. Same for AR.

*******     HRK > 0 and ARK > 0     *******           HRK > 0 and ARK < 0
 *****                               ********         and |ARK| > |HRK|
 H***A                               H*********A
 *****                               ********
*******                             *******
```

16.2.1 Methods

__init__(*self*, *Wedge*=0.0, *Alpha*=0.0, *X*=0.0, *Y*=0.0, *Z*=0.0, *Theta*=1.57079632679, *Phi*=0.0, *Diameter*=0.1, *HRr*=0.99, *HRt*=0.01, *ARr*=0.1, *ARt*=0.9, *HRK*=0.01, *ARK*=0, *Thickness*=0.02, *N*=1.4585, *KeepI*=False, *Ref*=None)

Mirror initializer.

Parameters are the attributes and the angles theta and phi are spherical coordinates of HRNorm.

Returns a mirror.

Overrides: object.__init__

lines(*self*)

Returns the list of lines necessary to print the object.

Overrides: theia.optics.component.SetupComponent.lines

isHit(*self*, *beam*)

```
Determine if a beam hits the Optic.

This is a function for mirrors, using their geometrical
attributes. This uses the line***Inter functions from the geometry
module to find characteristics of impact of beams on mirrors.

beam: incoming beam. [GaussianBeam]

Returns a dictionary with keys:
    'isHit': whether the beam hits the optic. [boolean]
    'intersection point': point in space where it is first hit.
            [3D vector]
    'face': to indicate which face is first hit, can be 'HR', 'AR' or
        'Side'. [string]
    'distance': geometrical distance from beam origin to impact. [float]
```

Overrides: theia.optics.component.SetupComponent.isHit

hit(*self*, *beam*, *order*, *threshold*)

```
Compute the refracted and reflected beams after interaction.

The beams returned are those selected after the order and threshold
criterion.

beam: incident beam. [GaussianBeam]
order: maximum strayness of daughter beams, whixh are not returned if
    their strayness is over this order. [integer]
threshold: idem for the power of the daughter beams. [float]

Returns a dictionary of beams with keys:
    't': refracted beam. [GaussianBeam]
    'r': reflected beam. [GaussianBeam]
```

Overrides: theia.optics.component.SetupComponent.hit

hitHR(*self*, *beam*, *point*, *order*, *threshold*)

```
Compute the daughter beams after interaction on HR at point.

beam: incident beam. [GaussianBeam]
point: point in space of interaction. [3D vector]
order: maximum strayness of daughter beams, whixh are not returned if
    their strayness is over this order. [integer]
threshold: idem for the power of the daughter beams. [float]

Returns a dictionary of beams with keys:
    't': refracted beam. [GaussianBeam]
    'r': reflected beam. [GaussianBeam]
```

hitAR(*self*, *beam*, *point*, *order*, *threshold*)

```
Compute the daughter beams after interaction on AR at point.

beam: incident beam. [GaussianBeam]
point: point in space of interaction. [3D vector]
order: maximum strayness of daughter beams, which are not returned if
    their strayness is over this order. [integer]
threshold: idem for the power of the daughter beams. [float]

Returns a dictionary of beams with keys:
    't': refracted beam. [GaussianBeam]
    'r': reflected beam. [GaussianBeam]
```

Inherited from theia.optics.optic.Optic(Section 17.2)

apexes(), collision(), geoCheck(), hitSide(), translate()

Inherited from theia.optics.component.SetupComponent(Section 13.2)

__str__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

16.2.2 Properties

Name	Description
Inherited from object ___class___	

16.2.3 Class Variables

Name	Description
Name	**Value:** `'Mirror'`
___abstractmethods___	**Value:** `frozenset([])`
Inherited from theia.optics.optic.Optic (Section 17.2) OptCount	
Inherited from theia.optics.component.SetupComponent (Section 13.2) SetupCount	

17 Module theia.optics.optic

Defines the Optic class for theia.

17.1 Variables

Name	Description
__package__	**Value:** 'theia.optics'

17.2 Class Optic

object

theia.optics.component.SetupComponent

theia.optics.optic.Optic

Known Subclasses: theia.optics.lens.Lens, theia.optics.mirror.Mirror

```
Optic class.

This class is a base class for optics which may interact with Gaussian
beams and return transmitted and reflected beams (mirrors, lenses, etc.)


*=== Attributes ===*
SetupCount (inherited): class attribute, counts all setup components.
    [integer]
OptCount: class attribute, counts optical components. [integer]
Name: class attribute. [string]
HRCenter (inherited): center of the 'chord' of the HR surface. [3D vector]
HRNorm (inherited): unitary normal to the 'chord' of the HR (always pointing
 towards the outside of the component). [3D vector]
Thick (inherited): thickness of the optic, counted in opposite direction to
    HRNorm. [float]
Dia (inherited): diameter of the component. [float]
Ref (inherited): reference string (for keeping track with the lab). [string]
ARCenter: center of the 'chord' of the AR surface. [3D vector]
ARNorm: unitary normal to the 'chord' of the AR (always pointing
 towards the outside of the component). [3D vector]
```

```
N: refraction index of the material. [float]
HRK, ARK: curvature of the HR, AR surfaces. [float]
HRr, HRt, ARr, ARt: power reflectance and transmission coefficients of
    the HR and AR surfaces. [float]
KeepI: whether of not to keep data of rays for interference calculations
        on the HR. [boolean]

**Note**: the curvature of any surface is positive for a concave surface
(coating inside the sphere).
Thus kurv*HRNorm/|kurv| always points to the center
of the sphere of the surface, as is the convention for the lineSurfInter of
geometry module. Same for AR.

*******     HRK > 0 and ARK > 0     *******          HRK > 0 and ARK < 0
 *****                               ********        and |ARK| > |HRK|
 H***A                               H*********A
 *****                               ********
*******                             *******
```

17.2.1 Methods

__init__(*self*, *ARCenter*, *ARNorm*, *N*, *HRK*, *ARK*, *ARr*, *ARt*, *HRr*, *HRt*, *KeepI*, *HRCenter*, *HRNorm*, *Thickness*, *Diameter*, *Ref*)

Optic base initializer.

Parameters are the attributes of the object to construct.

Returns an Optic.

Overrides: object.__init__

apexes(*self*)

Returns the positions of the apexes of HR and AR as a tuple.

collision(*self*)

Determine whether the HR and AR surfaces intersect.

Returns True if there is an intersection, False if not.

geoCheck(*self*, *word*)

Makes geometrical checks on surfaces and warns when necessary.

hitSide(*self*, *beam*)

Compute the daughter beams after interaction on Side at point.

Generic function: all sides stop beams.

beam: incident beam. [GaussianBeam]

Returns {'t': None, 'r': None}

translate(*self*, *X*=`0.0`, *Y*=`0.0`, *Z*=`0.0`)

Move the optic to (current position + (X, Y, Z)).

This version takes care of HRcenter and ARCenter and overwrites the SetupComponent version.

X, Y, Z: components of the translation vector.

No return value.

Overrides: theia.optics.component.SetupComponent.translate

Inherited from theia.optics.component.SetupComponent(Section 13.2)

__str__(), hit(), isHit(), lines()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

17.2.2 Properties

Name	Description
Inherited from object __class__	

17.2.3 Class Variables

Name	Description
Name	**Value:** `'Optic'`
OptCount	**Value:** `0`
Inherited from theia.optics.component.SetupComponent (Section 13.2) SetupCount, __abstractmethods__	

18 Module theia.optics.thicklens

Defines the ThickLens class for theia.

18.1 Variables

Name	Description
__package__	**Value:** `'theia.optics'`

18.2 Class ThickLens

object
theia.optics.component.SetupComponent
theia.optics.optic.Optic
theia.optics.lens.Lens
theia.optics.thicklens.ThickLens

```
ThickLens class.

This class represents thick lenses, specified by curvatures and thickness
instead of focal length.

*==== Attributes ===*
SetupCount (inherited): class attribute, counts all setup components.
    [integer]
OptCount (inherited): class attribute, counts optical components. [integer]
Name: class attribute. [string]
HRCenter (inherited): center of the 'chord' of the HR surface. [3D vector]
HRNorm (inherited): unitary normal to the 'chord' of the HR (always pointing
    towards the outside of the component). [3D vector]
Thick (inherited): thickness of the optic, counted in opposite direction to
    HRNorm. [float]
Dia (inherited): diameter of the component. [float]
Ref (inherited): reference string (for keeping track with the lab). [string]
ARCenter (inherited): center of the 'chord' of the AR surface. [3D vector]
ARNorm (inherited): unitary normal to the 'chord' of the AR (always pointing
    towards the outside of the component). [3D vector]
N (inherited): refraction index of the material. [float]
HRK, ARK (inherited): curvature of the HR, AR surfaces. [float]
HRr, HRt, ARr, ARt (inherited): power reflectance and transmission
    coefficients of the HR and AR surfaces. [float]
KeepI (inherited): whether of not to keep data of rays for interference
    calculations on the HR. [boolean]

**Note**: the curvature of any surface is positive for a concave surface
(coating inside the sphere).
Thus kurv*HRNorm/|kurv| always points to the center
of the sphere of the surface, as is the convention for the lineSurfInter of
geometry module. Same for AR.

*******     HRK > 0 and ARK > 0     *******          HRK > 0 and ARK < 0
 *****                               ********        and |ARK| > |HRK|
 H***A                               H*********A
 *****                               ********
*******                             *******

**Note**: in the case of thicklenses, the thickness provided to and by the
initializer is the thickness *on the optical axis*, and not the thickness
on the side of the component (like mirrors).

**Note**: in the case of thicklenses, the center provided to the initializer
is the *apex* of the principal face, and not the chord of the HR surface.
```

18.2.1 Methods

__init__(*self*, *K1*=0.01, *K2*=0.01, *X*=0.0, *Y*=0.0, *Z*=0.0, *Theta*=1.57079632679, *Phi*=0.0, *Thickness*=0.02, *N*=1.4585, *KeepI*=False, *Diameter*=0.05, *R*=0.1, *T*=0.9, *Ref*=None)

ThickLens initializer.

Parameters are the attributes.

Returns a ThickLens.

Overrides: object.__init__

lines(*self*)

Returns the list of lines necessary to print the object.

Overrides: theia.optics.component.SetupComponent.lines

Inherited from theia.optics.lens.Lens(Section 15.2)

hit(), hitActive(), isHit()

Inherited from theia.optics.optic.Optic(Section 17.2)

apexes(), collision(), geoCheck(), hitSide(), translate()

Inherited from theia.optics.component.SetupComponent(Section 13.2)

__str__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

18.2.2 Properties

Name	Description
Inherited from object __class__	

18.2.3 Class Variables

Name	Description
Name	**Value:** `'ThickLens'`
__abstractmethods__	**Value:** `frozenset([])`
Inherited from theia.optics.optic.Optic (Section 17.2) OptCount	
Inherited from theia.optics.component.SetupComponent (Section 13.2) SetupCount	

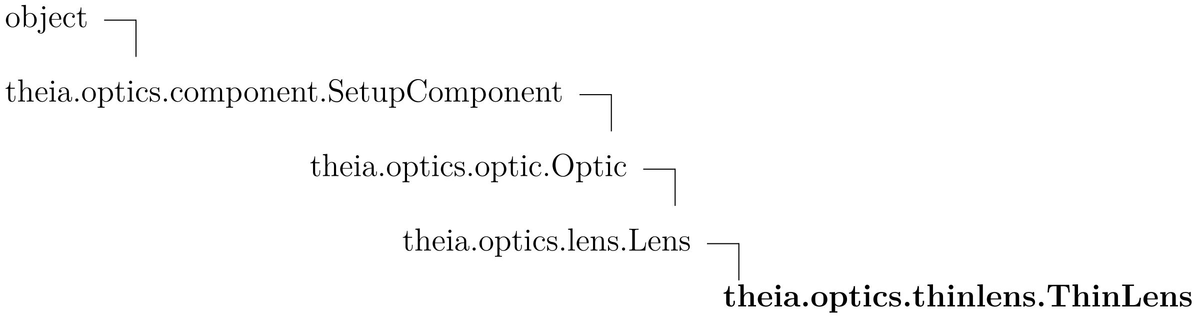
19 Module theia.optics.thinlens

Defines the ThinLens class for theia.

19.1 Variables

Name	Description
__package__	**Value:** `'theia.optics'`

19.2 Class ThinLens


object
theia.optics.component.SetupComponent
theia.optics.optic.Optic
theia.optics.lens.Lens
theia.optics.thinlens.ThinLens

```
ThinLens class.

This class represents thin lenses, which are specified only by their focal
lengths, diameter, position and orientation. Only the initializer and the
printing distinguishes thin lenses (in implementation) from other lenses.

*=== Attributes ===*
SetupCount (inherited): class attribute, counts all setup components.
    [integer]
OptCount (inherited): class attribute, counts optical components. [integer]
Name: class attribute. [string]
HRCenter (inherited): center of the 'chord' of the HR surface. [3D vector]
HRNorm (inherited): unitary normal to the 'chord' of the HR (always pointing
    towards the outside of the component). [3D vector]
Thick (inherited): thickness of the optic, counted in opposite direction to
    HRNorm. [float]
Dia (inherited): diameter of the component. [float]
Ref (inherited): reference string (for keeping track with the lab). [string]
ARCenter (inherited): center of the 'chord' of the AR surface. [3D vector]
```

```
ARNorm (inherited): unitary normal to the 'chord' of the AR (always pointing
    towards the outside of the component). [3D vector]
N (inherited): refraction index of the material. [float]
HRK, ARK (inherited): curvature of the HR, AR surfaces. [float]
HRr, HRt, ARr, ARt (inherited): power reflectance and transmission
    coefficients of the HR and AR surfaces. [float]
KeepI (inherited): whether of not to keep data of rays for interference
    calculations on the HR. [boolean]
Focal: Focal length of the lens. [float]

**Note**: the curvature of any surface is positive for a concave surface
(coating inside the sphere).
Thus kurv*HRNorm/|kurv| always points to the center
of the sphere of the surface, as is the convention for the lineSurfInter of
geometry module. Same for AR.

*******     HRK > 0 and ARK > 0     *******          HRK > 0 and ARK < 0
 *****                               ********        and |ARK| > |HRK|
 H***A                               H*********A
 *****                               ********
*******                             *******
```

19.2.1 Methods

__init__(*self*, *Focal*=0.1, *KeepI*=False, *Theta*=1.57079632679, *Phi*=0.0, *Diameter*=0.05, *R*=0.1, *T*=0.9, *X*=0.0, *Y*=0.0, *Z*=0.0, *Ref*=None)

ThinLens initializer.

Parameters are the attributes.

Returns a ThinLens.

Overrides: object.__init__

lines(*self*)

Returns the list of lines necessary to print the object.

Overrides: theia.optics.component.SetupComponent.lines

Inherited from theia.optics.lens.Lens(Section 15.2)

hit(), hitActive(), isHit()

Inherited from theia.optics.optic.Optic(Section 17.2)

apexes(), collision(), geoCheck(), hitSide(), translate()

Inherited from theia.optics.component.SetupComponent(Section 13.2)

__str__()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

19.2.2 Properties

Name	Description
Inherited from object __class__	

19.2.3 Class Variables

Name	Description
Name	**Value:** `'ThinLens'`
__abstractmethods__	**Value:** `frozenset([])`
Inherited from theia.optics.optic.Optic (Section 17.2) OptCount	
Inherited from theia.optics.component.SetupComponent (Section 13.2) SetupCount	

20 Package theia.rendering

This is the rendering sub-package of theia.

It allows to write the physical objects to FreeCAD format files for 3D rendering.

Version: 0.1.2

Author: Raphaël Duque

Copyright: Copyright 2017, Raphaël Duque

License: GNU GPLv3+

20.1 Modules

- **features**: Features module or theia, to represent objects as FreeCAD Python features.
  (Section 21, p. 49)
- **shapes**: Shapes module for theia, provides shape-calculating for 3D rendering.
  (Section 22, p. 54)
- **writer**: Writer module for theia, to write CAD content to files.
  (Section 23, p. 55)

21 Module theia.rendering.features

Features module or theia, to represent objects as FreeCAD Python features.

21.1 Class FCObject

object ─┐

theia.rendering.features.FCObject

```
Mother class for all FeaturePython objects.

This is to define the mandatory execute method, which is ran for every
    object of the document on the recompute function.

fact: Factor to compensate for unit difference with FreeCAD. [float]
```

21.1.1 Methods

__init__(*self*, *obj*)

Custom properties of the object.

Overrides: object.__init__

execute(*self*, *fp*)

We're not doing anything on recompute yet.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

21.1.2 Properties

Name	Description
Inherited from object __class__	

21.1.3 Class Variables

Name	Description
fact	**Value:** `0.001`

21.2 Class FCMirror

object

theia.rendering.features.FCObject

theia.rendering.features.FCMirror

21.2.1 Methods

__init__(*self*, *obj*, *mirror*)

Custom properties of the object.

Overrides: object.__init__ extit(inherited documentation)

Inherited from theia.rendering.features.FCObject(Section 21.1)

execute()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

21.2.2 Properties

Name	Description
Inherited from object __class__	

21.2.3 Class Variables

Name	Description
Inherited from theia.rendering.features.FCObject (Section 21.1) fact	

21.3 Class FCLens

```
object ─┐
        │
theia.rendering.features.FCObject ─┐
                                   │
                                   theia.rendering.features.FCLens
```

21.3.1 Methods

__init__(*self*, *obj*, *lens*)

Custom properties of the object.

Overrides: object.__init__ extit(inherited documentation)

Inherited from theia.rendering.features.FCObject(Section 21.1)

execute()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

21.3.2 Properties

Name	Description
Inherited from object	
__class__	

21.3.3 Class Variables

Name	Description
Inherited from theia.rendering.features.FCObject (Section 21.1)	
fact	

21.4 Class FCBeamDump

object
theia.rendering.features.FCObject
theia.rendering.features.FCBeamDump

21.4.1 Methods

__init__(*self*, *obj*, *beamDump*)

Custom properties of the object.

Overrides: object.__init__ extit(inherited documentation)

Inherited from theia.rendering.features.FCObject(Section 21.1)

execute()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

21.4.2 Properties

Name	Description
Inherited from object	
__class__	

21.4.3 Class Variables

Name	Description
Inherited from theia.rendering.features.FCObject (Section 21.1)	
fact	

21.5 Class FCBeam

object

theia.rendering.features.FCObject

theia.rendering.features.FCBeam

21.5.1 Methods

___init___(*self*, *obj*, *beam*)

Custom properties of the object.

Overrides: object.___init___ extit(inherited documentation)

Inherited from theia.rendering.features.FCObject(Section 21.1)

execute()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

21.5.2 Properties

Name	Description
Inherited from object	
___class___	

21.5.3 Class Variables

Name	Description
Inherited from theia.rendering.features.FCObject (Section 21.1)	
fact	

22 Module theia.rendering.shapes

Shapes module for theia, provides shape-calculating for 3D rendering.

22.1 Functions

mirrorShape(*mirror*)

Computes the 3D representation of the beam, a shape for a CAD file obj.

beam: beam to represent. [GaussianBeam]

Returns a shape for a CAD file object.

lensShape(*lens*)

Computes the 3D representation of the lens, a shape for a CAD file obj.

lens: lens to represent. [GaussianBeam]

Returns a shape for a CAD file object.

beamDumpShape(*beamDump*)

Computes the 3D representation of the beam, a shape for a CAD file obj.

beam: beam to represent. [GaussianBeam]

Returns a shape for a CAD file object.

ghostShape(*ghost*)

Computes the 3D representation of the beam, a shape for a CAD file obj.

beam: beam to represent. [GaussianBeam]

Returns a shape for a CAD file object.

beamShape(*beam*)

Computes the 3D representation of the beam, a shape for a CAD file obj.

beam: beam to represent. [GaussianBeam]

Returns a shape for a CAD file object.

23 Module theia.rendering.writer

Writer module for theia, to write CAD content to files.

23.1 Functions

writeToCAD(*component*, *doc*)

```
Write the relevant shape and feature content of components in CAD file.

This function is for everython except for beams.
To the doc .fcstd file are added two objects, one of type
    App::FeaturePython which will hold the internal data of the component
    for reviewing in the side panel of FreeCAD, and one of type
    Part::Feature for visualization. The classes for the App::FeaturePython
    objects are i nthe features modules, and those for the shapes are in
    the shapes module.
The important functions are the PythonFeatures
    constructors found in features, and the shape functions found in shapes.

component: component to represent. [Mirror, Lens, BeamDump, Ghost, Beam]
doc: CAD file to write to. [CAD file]

No return value.
```

writeTree(*tree*, *doc*)

```
Recursively write the shape and feature content of the beams of a tree.

If the tree’s root is not None, write the shape and feature for tree.Root
    and start over for the daughter trees.

tree: beamtree to write the info. [BeamTree]
doc: CAD file to write to. [CAD file]

No return value.
```

24 Package theia.running

This is the running sub-package of theia.

It provides the necessary classes and functions to allow the input, output and encapsulation of simulation data.

Version: 0.1.2

Author: Raphaël Duque

Copyright: Copyright 2017, Raphaël Duque

License: GNU GPLv3+

24.1 Modules

- **parser**: Module for the parsing on input data from .tia file.
  (Section 25, p. 57)
- **simulation**: Defines the Simulation class for theia.
  (Section 26, p. 59)

25 Module theia.running.parser

Module for the parsing on input data from .tia file.

25.1 Functions

dicOf(*st*, *line*, *fileName*, *lineNumber*)

```
Extract the initializer dictionary from a line.

st: object tag, 'bm', 'th', ... [string]
line: line of data in .tia format (supposed no spaces nor tabs nor comments)
and without the obect tag. [string]
fileName: name of file (used to write errors). [string]
lineNumber: number fo this line in the file (used to write errors). [int]

    May raise an InputError
    Returns a dictionary ready for construction.
```

readIn(*name*)

```
Finds the input data in a file.

Returns a list of tuples where tuple[0] identifies the object of which data
has been found and tuple[1] the data itself. tuple[1] may be a simple value
or a dictionary for constructors, etc.

Example return value: [ ('bd', {'X': 0., 'Y': 0., 'Z': 1.}),    #constructor
                        ('LName', 'foo')]   #string data.

name: file to read. [string]

May raise an InputError.

Returns a list of tuples.
```

25.2 Variables

Name	Description
GHz	**Value:** 1000000000.0
Hz	**Value:** 1.0

continued on next page

Name	Description
MHz	**Value:** `1000000.0`
THz	**Value:** `1e+12`
W	**Value:** `1.0`
__package__	**Value:** `’theia.running’`
arccos	**Value:** `<ufunc ’arccos’>`
arcsin	**Value:** `<ufunc ’arcsin’>`
arctan	**Value:** `<ufunc ’arctan’>`
cm	**Value:** `0.01`
cos	**Value:** `<ufunc ’cos’>`
deg	**Value:** `0.0174532925199`
exp	**Value:** `<ufunc ’exp’>`
kHz	**Value:** `1000.0`
kW	**Value:** `1000.0`
km	**Value:** `1000.0`
m	**Value:** `1.0`
mHz	**Value:** `0.001`
mW	**Value:** `0.001`
mm	**Value:** `0.001`
nW	**Value:** `1e-09`
nm	**Value:** `1e-09`
pi	**Value:** `3.14159265359`
ppm	**Value:** `1e-06`
rad	**Value:** `1.0`
sin	**Value:** `<ufunc ’sin’>`
sqrt	**Value:** `<ufunc ’sqrt’>`
tan	**Value:** `<ufunc ’tan’>`
uHz	**Value:** `1e-06`
uW	**Value:** `1e-06`
um	**Value:** `1e-06`

26 Module theia.running.simulation

Defines the Simulation class for theia.

26.1 Variables

Name	Description
__package__	**Value:** `'theia.running'`

26.2 Class Simulation

object
theia.running.simulation.Simulation

```
Simulation class.

This class is a wrapper for all the metadata (names of setup and of files,
etc.) as well as for the high level functions of a simulation.

*=== Attributes ===*
LName: name of the simulation [string]
FName: name of the file for outputs (without extension) [string]
OptList: list of optical components of the setup [list of optics]
InBeams: list of input beams [list of beams]
BeamTreeList: list of binary trees of beams [list of BeamTree]
Order: order of the simulation, beams transmitted by HRs or reflected by ARs
    have their orders augmented by 1, and simulation calculates only until
    this Order attribute. [int]
Threshold: Power under which beams are no longer traced. [float]
Date: string of the date-time when the simulation was created (not run).
    [string]
```

26.2.1 Methods

__init__(*self*, *FName*='simulationinput')

Simulation initializer.

FName: output files name without extension. [string]

Overrides: object.__init__

__str__(*self*)

String representation of the simulation, for print(simulation).

Overrides: object.__str__

numberOfOptics(*self*)

Calculate the number of optics of OptList.

Returns the number of optics (not components, optics).

load(*self*)

Initialize simulation attributes by input from .tia file.

See documantation for the format of the input file.

No return value.

run(*self*)

```
Run simulation with input as read by load.

threshold: power of beam below which the simulation stops tracing child
            beams. [float]
order: maximum order to keep daughter beams. [integer]

No return value.
```

writeOut(*self*)

Write the results from the simulation in the .out file.

writeCAD(*self*)

Write the CAD .fcstd file by calling rendering functions.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

26.2.2 Properties

Name	Description
Inherited from object	
__class__	

27 Package theia.tree

This is the tree sub-package of theia.

It provides the necessary classes and functions to allow the reverse ray tracing and stray light hunting features of theia.

Version: 0.1.2

Author: Raphaël Duque

Copyright: Copyright 2017, Raphaël Duque

License: GNU GPLv3+

27.1 Modules

- **beamtree**: Defines the BeamTree class for theia.
  (Section 28, p. 63)

28 Module theia.tree.beamtree

Defines the BeamTree class for theia.

28.1 Functions

treeOfBeam(*srcBeam*, *optList*, *order*, *threshold*)

Function to calculate the tree of daughter beams of srcBeam.

srcBeam: Input beam. [GaussianBeam] optList: List of optical components of the setup. [list of OpticalComponent] order: order of simulation. [integer] threshold: power threshold for daughter beams. [float]

Returns a BeamTree.

28.2 Variables

Name	Description
__package__	**Value:** `'theia.tree'`

28.3 Class BeamTree

object ─┐

theia.tree.beamtree.BeamTree

BeamTree class.

A BeamTree is a binary tree which allows to keep track of the beams as they are traced throughout the optical setup. The Root of the tree is a Gaussian beam and the other attributes are the daughter trees and all the data of the interaction producing these with the Root beam

=== Attributes === Name: class attribute, name of object. [string] Root: beam of this node of the tree. [GaussianBeam] T: beam resulting from the transmission of the Root beam. [BeamTree] R: beam resulting from the reflection of the Root beam. [BeamTree]

28.3.1 Methods

__init__(*self*, *Root*=`None`, *T*=`None`, *R*=`None`)

BeamTree initializer.

Overrides: object.__init__

__str__(*self*)

String representation of a BeamTree, for print(tree).

Overrides: object.__str__

lines(*self*)

Returns the list of lines necessary to print the object.

beamList(*self*)

Returns the string representation the tree of beams.

beamLines(*self*)

Returns the list of lines necessary to print the list of beams.

numberOfBeams(*self*)

Return the total number of beams.

outputLines(*self*)

Return the list of lines to write the output of simulation.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __subclasshook__()

28.3.2 Properties

Name	Description
Inherited from object __class__	

28.3.3 Class Variables

Name	**Description**
Name	**Value:** `'BeamTree'`

Index