StarLyX, Monte-Carlo simulation for complex scenes and optical instruments (DRAFT)

Laurent Brunel, PhotonLyX

December 29, 2023

Contents

1 Introduction

Starlyx address to people interested in modelling and measuring light interaction with complex materials. Despite the huge offer for rendering engines, none, to our knowledge, allows flexibility to go beyond synthetic image rendering and simulate optical instruments able to exploit scattering properties of materials. Starlyx allows in one tool to modelise instruments involving diffusive material and to render visual sensation given by volume light scattering inside complex objects. We propose this software, GPL3 licensed, that could be a common tool for people involved in radiative transfer, scattering media studies and visual appearance. Starlyx uses the star-engine library (Meso-Star, Toulouse, France) (star-engine[?]) with itself stands on Embree (Intel)[?]. It is written in C language.

The main points characterizing starlyx are: Monte-Carlo engine dedicated to volume-scattering, spectral simulation, no C coding needed at a start, CPU parallelism, scene and studies described by a single XML file and mesh files (obj), calculation time little dependent on geometry complexity, Monte-Carlo error estimation . Starlyx is initiated by PhotonLyX (Santander,Spain).

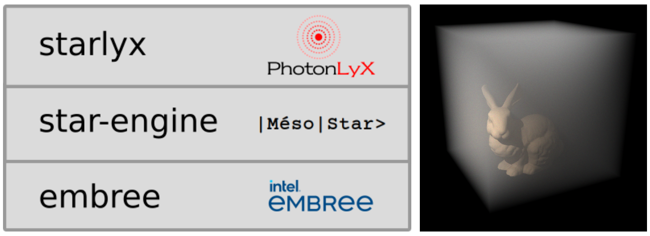

Figure 1: starlyx relies on star-engine (meso-star) and Embree (Intel). Illustrative examples: rendering of the Stanford bunny in a foggy cube.

The last release, examples, source code, installation instructions and this doc are available in: https://gitlab.com/photonlyx/starlyx

2 Detailed description of the "minimal-cube" example

Run starlyx to create the render of the scattering cube:

```
cd examples/minimal-cube
starlyx scene.xml
```

See the result in cam0.png (figure 2).

Let's comment the file scene.xml of this example step by step. The main object of the XML file has the type Scene:

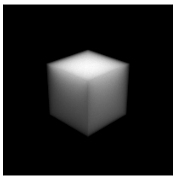

Figure 2: image rendered by the example minimal-cube

```
<Scene
  ALGORITHM="reverse_1"
  VERBOSE="1"
  NB_PHOTONS="100"
  STORE_TRAJECTORIES="0"
  STORE_ERROR_TRAJECTORIES="0"
  MAX_PATH_LENGTH="1000"
>
```

The attribute ALGORITHM is set to "reverse_1" to specify that the Monte-Carlo algorithm starts the ray tracing from the sensor (the camera). If ALGORITHM is set to "direct", the ray tracing will start from the sources.

The attribute VERBOSE is set to 1 to get printed output of the calculation process. The attribute STORE_TRAJECTORIES can be set to 1 to get the photon trajectories in the file paths.obj viewable by any 3D visualisation software like blender of paraview. Use this option only for debugging only since the file "paths.obj" might be very large.

The attribute STORE_ERROR_TRAJECTORIES can be set to 1 to get the photon trajectories that have given an error (ray tracing error, infinite loop,...) in the file "paths_error.obj".

The first son of the Scene is the object of type source named "sourceSun0":

```
<source
  NAME="sourceSun0"
  RADIANCE="1e7" <!-- W m-2 sr-1 -->
  TYPE="sun"
  ANGLE="0.5" <!-- deg -->
  SPECTRUM="red"
>
  <dir X="0.2" Y="0.3" Z="-1.0"></dir>
</source>
```

The attribute TYPE refers to the source type: here the type "sun" which is an infinitely wide source illuminating in a defined direction. This direction is mentioned in the son object of type "dir". The direction does not need to be normalised since it will be automatically done by the software. The attribute ANGLE set the angular diameter of the sun (typically 0.5^{o}). The attribute POWER refers to the source power, in W/m^2 for this type. The attribute "SPECTRUM" refers to the source power for each wavelength in case of a spectral simulation. In this case the simulation is monochromatic and the spectrum refers to the object of type "spectrum" called "red" which has a single wavelength of 650nm with the value of 1 (W/m^2).

The camera that produce the rendering is described in the object of type "camera" and NAME "cam0"

```
<camera
  NAME="cam0"
  NBPIXELSY="200"
  NBPIXELSX="200"
  FIELD="30.0" <!-- degrees -->
  FILTERS="filterR"
>
  <lens Z="3.0" Y="-3.0" X="-3.0">
  </lens>
  <viewPoint Z="0.5" Y="0.5" X="0.5">
  </viewPoint>
</camera>

<spectrum
  NAME="filterR"
```

```
  DATA="650 1"
>
</spectrum>
```

The attributes NBPIXELSX and NBPIXELSY describe the size the image rendered. The attribute "FIELD" indicates the camera field in degrees. The attribute "FILTERS" refers to the spectral filters used by the different spectral channels of the camera. In this case, we have a monochromatic camera using a monochromatic filter called "filterR". This filter is defined by the object of type "spectrum" named "filterR", placed later in the XML description. The object "cam0" has 2 sons, of type "lens" and viewPoint. The son of type "lens" refers to the position of the camera lens in scene's reference frame. The son of type "viewPoint" indicates the position of the central point targeted by the camera in the scene's reference frame.

The scene contains only one physical object of cubic shape. The surface of this shape is defined by obj file indicated in the attribute "FILE":

```
<surface
  NAME="cube_surface"
  MATERIALS="dielectric1"
  FILE="cube.obj">
</surface>
```

The material of the surface is indicated in the attribute "MATERIALS" which may contain a list of surface materials describing the behaviour of the photon when it hits this surface. In this case, "MATERIALS" refers to the object of name "dielectric1" which type is "dielectric":

```
  <dielectric NAME="dielectric1" ></dielectric>
```

As a consequence, the photon will reflect or refract following the Fresnel laws (ignoring polarization). The needed refractive indices are indicated in the volumes descriptions of the XML file (next). For the moment, the other types of surface material availables are "mirror" and "lambert". They are explained later in this paper. Note that the attribute "MATERIALS" might be empty and in this case the photon will have no interaction with the surface.

The unique volume in the scene is delimited by only one surface: the one which name is "cube_surface". This is indicated in the attribute "SURFACES". The name is in the attribute "NAME", the refractive index in the attribute "N". We will see later that "N" could also refer to a spectrum. The radiative transfer properties of this volume are described in the attribute "MATERIALS" which refers here to another XML object of name "hg2" and type "Henyey-Greenstein".

```
<volume
  NAME="cube_volume"
  N="1"
  MATERIALS="hg2"
  SURFACES="cube_surface">
  </volume>
```

In this example, the radiative transfer properties are defined by the transport length "LSTAR", the asymetry factor "G" anf the absorption length LA. We will see later that other sets of parameters are possible. These parameters may also be spectral.

```
<Henyey-Greenstein
  NAME="hg2"
  LA="10000000"  <!-- mm -->
  LSTAR="0.1" <!-- mm -->
  G="0"
>
</Henyey-Greenstein>
```

The last line must close the XML object of type "Scene".

```
</Scene>
```

3 User guide

The geometry of the scene is described by surface and volumes. The volumes must be entirely closed by one or many surfaces (figure 3).

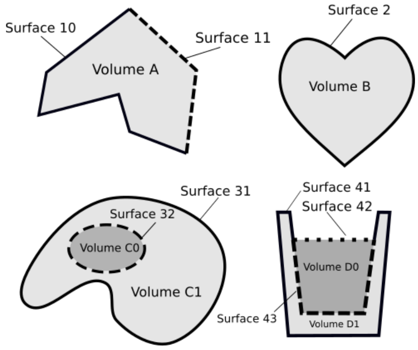

Figure 3: The surfaces delimit the volumes. Volume A is delimited by the surfaces 10 and 11. Volume B is delimited by surface 2. Volume C0 is delimited by surface 32 and volume C1 by surfaces 31 and 32. Volume D0 is delimited by surfaces 42 and 43. Volume D1 by surfaces 41 and 43.

To each volume is associated a material volume (absorption and Henyey-Greenstein scattering properties). To each surface is associated a material surface.

3.1 Surfaces

A surface in the scene is defined by a geometry file with format OBJ or STL. The name of the file is specified in the attribute FILE. A surface doesn't have to be closed and doesn't need any normal orientation convention. The material of the surface is defined in the attribute "MATERIALS". Indicate just one material for the moment. Examples:

```
<surface
  NAME="cube_surface"
  MATERIALS="dielectric1"
  FILE="cube.obj">
  </surface>
  <surface
    NAME="bunny"
    MATERIALS=""
    FILE="bunny.stl">
</surface>
<surface
  NAME="sphere"
  MATERIALS="mirror1"
  FILE="sphere.stl">
</surface>

```

3.1.1 Surfaces modelized with Openscad

A surface can also be directly designed in the scene description using the openscad software [openscad.org]. Openscad (version 2021.01) must be installed and the PATH environment variable must point to it.

```
<openscad
    NAME="cube"
  CODE="cube();"
  MATERIALS="lambert1"
>
</openscad>
<openscad
  NAME="cyl2"
  CODE="translate([0,0,0.5])  cylinder(d1=1,d2=2,h=2,center=false,$fn=100);"
  MATERIALS="dielectric1"
>
</openscad>
```

Note that when the openscad code has some double quotes ("), replace them by single quotes like in this example:

```
<openscad
  NAME="starlyx1"
  CODE="text1='Starlyx'; linear_extrude(1){text(text1,1);} "
  MATERIALS="dielectric1" >
</openscad>
```

3.2 Volumes

A volume is defined by joining one or several surfaces. When more that one surface are used, the surfaces must hermetically close the volume. The radiative transfer properties of the volume is defined in the attribute "MATERIALS" which may contains the names of several objects defining these properties(see chapter "Volume radiative transfer properties"). The attribute "N" specify the refractive index of the material in the volume.

```
<volume
  NAME="cube"
  N="1.33"
  MATERIALS="henyey-greenstein1"
  SURFACES="cube_surface">
</volume>
```

3.3 Sources

3.3.1 Sun

A sun at zenith with angular diameter of 0.5 degrees can be coded like that:

```
<source
  NAME="sourceSun0"
  RADIANCE="1e7" <!-- W m-2 sr-1 -->
  TYPE="sun"
  ANGLE="0.5"  <!-- degrees -->
  SPECTRUM="red">
  <dir X="0" Y="0" Z="-1"></dir>
</source>
```

3.3.2 Spot

A spot emitting light of diameter 1 mm, a total angle of emission 10 degrees, at position (0,0,0) and emiting light in the direction of X is coded like that:

```
<source
  NAME="sourceSpot0"
  POWER="1.0" <!-- W -->
  TYPE="spot"
  DIAMETER="1" <!-- mm -->
  ANGLE="10"   <!-- degrees -->
  SPECTRUM="red">
  <pos X="0" Y="0"Z="0"></pos>
  <dir X="1" Y="0" Z="0"></dir>
</source>
```

3.3.3 Source geometry defined by any extended surface

Here is he code for a spherical source with geometry is set in the file source_sphere.obj. The light is emitted in a direction around the local normal. The total emission angle around the normal is defined by the attribute "ANGLE". The light is emitted in the normal direction. See example "minimal-cube-source-surface" for the case of reverse algorithm. See example "photodiodes-source-sphere" for the case of reverse algorithm.

```
<source
  POWER="1.0"
  NAME="source0"
  TYPE="surface"
  FILE="source_sphere.obj"
  ANGLE="180"
  SPECTRUM="red"
  VOLUME="World" >
</source>
```

3.4 Sensors

3.4.1 Camera

A camera bundles many sensors (pixels) and is associated with a rendered image. For the moment, the camera sensor can be used only for the reverse path algorithms. See example "minimal-cube".

- NBPIXELSX,NBPIXELSY: number of pixels of the camera in horizontal and vertical direction.
- FIELD: field of view of the camera in degrees.
- FILTERS: list of the filters used for each pixel. A filter is an xml object of type "spectrum" that defines the spectral response of the filter. A typical color camera will have 3 filters for red, green and blue channels.
- NA: the numerical aperture of the camera: sin of the angle form axis to the aperture edge seen from the focal point.
- FOCAL: the focal length of the camera in mm.
- VOLUME: name of the volume where is the camera. For the moment, Starlyx doesn't detect automatically the volume. Put "World" when the camera is in the external volume.
- lens: the position of the lens centre
- viewPoint: the position of the focused point. This point will be the centre of the depth of field.

```
<camera
  NAME="cam0"
  NBPIXELSY="200"
  NBPIXELSX="200"
  FIELD="30.0"
  FILTERS="filterR"
  NA="0.2"
  FOCAL="2"
  VOLUME="World"
>
<lens Z="3.0" Y="-3.0" X="-3.0"></lens>
<viewPoint Z="0.5" Y="0.5" X="0.5"></viewPoint>
</camera>

<spectrum
  NAME="filterR"
  DATA="650 1">
</spectrum>
```

3.4.2 Probe

A probe is a disk in space representing a single detector. It is used to calculate the luminance around one point and around one direction. The acceptance is a cone with a defined angular aperture. The probe can be used only for the reverse path algorithms.

- DIAMETER: diameter of the disk in mm

- ANGLE: angle of the acceptance in degrees (full angle)
- FILTERS: list of the filters used for each pixel. A filter is an xml object of type "spectrum" that defines the spectral response of the filter.
- VOLUME: name of the volume where is the probe. For the moment, Starlyx doesn't detect automatically the volume. Put "World" when the probe is in the external volume.

```
<sensor
  NAME="probe1"
  DIAMETER="1"
  ANGLE="30"
  FILTERS="650 530 450"
  VOLUME="World">
  <pos X="0" Y="0" Z="10" > </pos>
  <viewPoint X="0" Y="0" Z="0" ></viewPoint>
</sensor>

```

3.4.3 Sensor defined by any surface

For the case of direct path algorithm only, all the surfaces are considered as a sensor. The Monte-Carlo weight is set to 1 when the path hits the surface and to 0 when not. At the end of the calculation the file sensors.csv is produced and contains the average weights and their standard deviations. See example "photodiodes". Note: this option will be replaced by a generalised probe with any surface shape, not only a disk, for direct and reverse algorithm.

3.5 Volume radiative transfer properties

A volume has a specific refraction index N and may contain various materials. There are 3 ways to describe these properties:

- (k,g,ka): scattering coefficient (mm-1), asymmetry coefficient, absorption coefficient (mm-1)
- (l*,g,la): transport length (mm), asymmetry coefficient, absorption length (mm)
- (d,nr,ni,phi): micro-spheres diameter (mm), refractive index real part, refractive index imaginary part, volume concentration of particles.

The triplet of parameters (k,g,ka) is the one finally used by the Monte-Carlo algorithm. The other parameters options are converted. l* is defined by the formula l*=1/k/(1-g). The set of parameters (d,nr,ni,phi) is converted to the triplet (k,g,ka) using the Mie theory. Here is an example of volume cumulating the 3 types of materials:

```
<volume
  NAME="cube"
  N="1"
  MATERIALS="hg1  hg2 mie1 "
>
</volume >

<Henyey-Greenstein
  NAME="hg1"
  K="1"
  KA="0.1"
  G="0"
>
</Henyey-Greenstein >

<Henyey-Greenstein
  NAME="hg2"
  LSTAR="1"
  LA="0.1"
  G="0"
>
```

```
</Henyey-Greenstein >

<Mie
  NAME="mie1"
  D_UM="0.001"
  NR="2.54"
  NI="0.001"
  PHI="0.01"
>
</Mie >
```

Let's take the example of milk. In a simplified view, milk can be modelised by a suspension of 5% in volume of fat particles of 1 μm size (refractive index 1.46) and 2% in volume of casein micelles of size 100 nm (refractive index 1.57).

```
<volume
  NAME="cube"
  N="1"
  MATERIALS="fat  casein">
</volume >

<Mie
  NAME="fat"
  D_UM="0.001"
  NR="1.46"
  NI="0"
  PHI="0.05" >
</Mie >

<Mie
  NAME="casein"
  D_UM="0.0001"
  NR="1.57"
  NI="0"
  PHI="0.02">
</Mie >
```

3.5.1 Diffusive medium modelised by Henyey-Greenstein using k, ka and g

A volume material can be defined by its scattering (k and g) and absorption (ka) properties. The phase function is defined by Henyey-Greenstein formula [?] and determined by the asymmetry coefficient g. Here is an example of medium with a scattering coefficient k = 1 mm-1, an asymmetry coefficient g=0 and an absorption coefficient ka= 0.1 mm-1 :

```
<Henyey-Greenstein
  NAME="hg1"
  K="1"
  KA="0.1"
  G="0"
>
</Henyey-Greenstein >
```

All the values can refer to the name of a spectrum object including this parameter as a function of the wavelength λ.

3.5.2 Diffusive medium modelised by Henyey-Greenstein using l*, g and la

Another way to define scattering and absorption properties can be using l* (transport length in mm) and la (absorption length = 1/ka). Here is an example with l* = 1 mm, g=0 and la = 0.1 mm.

```
<Henyey-Greenstein
  NAME="hg1"
  LSTAR="1"
  LA="0.1"
  G="0"
>
</Henyey-Greenstein >
```

All the values can refer to the name of a spectrum object including this parameter as a function of the wavelength λ.

3.5.3 Spherical particle suspension modelised by Mie theory

If the scattering volume is made of micro-metrical spherical particles, the properties can be modelised using the Mie theory [?, ?] with the sphere of diameter D and the complex refractive index (real part NR and imaginary part NI) of the sphere's material. Here is an example for a suspension of micro-spheres of Titane-dioxyde of diameter 1um :

```
<Mie
  NAME="mie1"
  D_UM="1"
  NR="2.54"
  NI="0"
>
</Mie >
```

All the values can be replaced by the name of a spectrum object including this parameter as a function of the wavelength λ.

3.5.4 Diffusive medium modelised by a phase function

A volume material can be defined by its scattering (k) and absorption (ka) properties. The phase function is defined by the user with a sampling for some angles (in degrees from 0 to 180). The values of the phase function for the missing angles are estimated by using linear interpolation. Here is an example of medium with a scattering coefficient k = 1 mm-1, a phase function defined by the curve pf1 and an absorption coefficient ka= 0.1 mm-1 :

```
<Scattering
  NAME="mv1"
  K="1"
  PHASE_FUNCTION="pf1"
  KA="0.1"
>
</Scattering >

<sampled_data
  NAME="pf1"
  DATA=" 0 10    10 10    20 10    30 0    60 0     90 0    120
  0    150 0     180 0    "
>
</sampled_data>
```

The phase function will automatically be resampled with a regular step of 1 degree from 0 to 180 degrees.

3.6 Surface properties:

A surface is described by the xml object of type "surface". The attribute "NAME" defines the xml object name. The attribute "FILE" contains the name of the obj file defining the 3D mesh of the surface. Its attribute "MATERIALS" contains the material describing the light behaviour when intersecting with it. The attribute "MATERIALS" can be empty: the surface won't interact with light. Here is an example of surface which 3D mesh is defined by the file "cube.obj" and which optical behaviour is defined by the xml object of name "dielectric1":

```
<surface
  NAME="cube_surface"
  MATERIALS="dielectric1"
  FILE="cube.obj">
</surface>
<dielectric NAME="dielectric1" > </dielectric>
```

The available types of material properties are "lambert", "dielectric" and "mirror".

3.6.1 Lambertian surface: lambert

A lambertian (matt) surface is described by the xml object of name lambert. Its attribute "ALBEDO" is in the [0,1] range. For example, if albedo=0.5, the light ray has 50% probability to be absorbed by the surface when hitting it. The albedo attribute can be replaced by the name of a spectrum object describing the albedo as a function of the wavelength λ.

```
<lambert NAME="lambert1" ALBEDO="1"> </lambert>
```

3.6.2 Dielectric interface: dielectric

An interface between 2 dielectric media is described by the xml object of name "dielectric".

```
<dielectric NAME="dielectric1" > </dielectric>
```

According to the refractive index of the 2 media, light may refract, reflect using the Fresnel laws or totally reflect is incidence is beyond the Brewster angle. Let's note i1 the incident angle in the medium of refractive index n1 and i2 the exit angle in the medium of refractive index n2. The exit angle i2 follows the Snell-Descartes law:

$$n1sin(i1) = n2sin(i2)$$

The reflexion coefficient R is calculated using the Fresnel law making an average of both polarisations:

$$Rs = \left(\frac{(n1 * cos(i1) - n2 * cos(i2))}{(n1 * cos(i1) + n2 * cos(i2))}\right)^2$$

$$Rp = \left(\frac{(n1 * cos(i2) - n2 * cos(1))}{(n1 * cos(i2) + n2 * cos(i1))}\right)^2$$

$$R = (Rs + Rp)/2$$

When n1 sin(i1)>n2 , i1 is the greater than the Brewster angle: the reflexion is total, R=1 and i2=i1.

3.6.3 Mirror

A reflecting surface is described by the xml object of name "mirror". The attribute R indicates the reflection coefficient. R attribute can be the name of a spectrum described in another place of the xml file.

```
<mirror NAME="mirror1" R="0.95" > </mirror>
```

3.6.4 Surface emission

To simulate a surface that emits light, just put an object of type "emission" in its attribute "MATERIALS". The unit of the corresponding spectrum is W/m^2. Note: this type of surface is not yet implemented for the direct algorithm (see section **??**).

```
<emission NAME="emission1"
SPECTRUM="spectrum_emission1" > </emission>
```

3.7 Spectral data

Raw spectral data are configured in the scene with objects of type <spectrum> </spectrum> . The attribute DATA contains the spectral values in format "λ1 v1 λ2 v2 .....". The unit for wavelengths is nm. Here is an example of RGB like spectrum with twice more power on green than other wavelengths:

```
<spectrum
  NAME="white"
  DATA="450 0.25 530 0.5 650 25"
>
</spectrum>
```

For the case of the "direct path" algorithm, the number of wavelength simulated is guided by the camera configuration by the way of its channels defined in the xml attribute "FILTERS". We have 3 typical cases: monochromatic simulation with on single channel defined for example by FILTERS="650", RGB simulation with 3 channels defined by FILTERS = "filterR filterG filterB" (referring to xml object of type "spectrum") or directly by FILTERS = "650 530 450", spectral simulation defined for example by FILTERS="700 680 660 640 620 600 580 560 540 520 500 480 460 440 420 400".

For the case of "direct path", the number of wavelength simulated is guided by the source spectrum configuration.

3.7.1 Spectral camera

A 3 wavelengths camera can be directly specified including the 3 wavelengths (in nm) in the attribute "FILTERS" of the xml object of type "camera".

```
<camera
  NAME="cam0"
  NBPIXELSY="50"
  NBPIXELSX="50"
  GAMMA="1"
  FIELD="30"
  FILTERS="650 530 450"
>
<lens X="1" Y="1" Z="1"></lens>
<viewPoint X="0" Y="0" Z="0"></viewPoint> </camera>
```

The filters for each camera channel can be specified addressing the 3 spectrums of each channel, like in the next example (see curves in figure 4):

```
<camera
  NAME="cam0"
  NBPIXELSY="200"
  NBPIXELSX="200"
  FIELD="30"
  FILTERS="filterR filterG filterB">
  <lens X="1" Y="1" Z="1"></lens>
  <viewPoint X="0" Y="0" Z="0"></viewPoint>
</camera>

  <spectrum
    NAME="filterR"
    DATA="400 0 410 0 420 0 430 0 440 0 450 0 460 0 470 0 480 0 490 0
    500 0 510 0 520 0 530 0 540 0 550 0 560 0 570 10 580 30 590 50 600 70 610 80 620
    90 630 90 640 80 650 70 660 50 670 30 680 10 690 0 700 0"
  >
  </spectrum>

  <spectrum
    NAME="filterG"
    DATA="400 0 410 0 420 0 430 0 440 0 450 0 460 0 470 0 480 0 480 0
    490 10 500 30 510 50 520 70 530 80 540 90 550 90 560 81 570 70 580 50 590 30 600
    10 610 0 620 0 630 0 640 0 650 0 660 0 670 0 680 0 690 0"
  >
  </spectrum>
  <spectrum
    NAME="filterB"
    DATA="400 0 410 10 420 30 430 50 440 70 450 80 460 90 470 90 480
    81 490 70 500 50 510 30 520 10 530 0 540 0 550 0 560 0 570 0 580 0 590 0 600 0
    610 0 620 0 630 0 640 0 650 0 660 0 670 0 680 0 690 0 700 0"
  >
</spectrum>
```

For a 3 channels camera, starlyx will create a color image cam0.ppm compatible with any image viewer. Starlyx will also create the raw format image cam0.raw (format explained in annex).

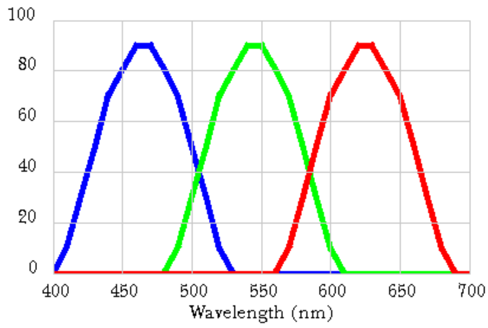

Figure 4: Example of filters for a RGB camera

3.7.2 Spectral source

The source description must contain the name of a spectral object in the attribute SPECTRUM. In the next example: the source uses the spectrum called "white":

```
<source
  NAME="source0"
  POWER="1.0"
  TYPE="spot"
  DIAMETER="0"
  ANGLE="180"
  SPECTRUM="white"
>
<pos X="0" Y="0" Z="0"></pos>
</source>

<spectrum
  NAME="white"
  DATA="450 1 530 1 650 1"
>
</spectrum>
```

For spectral simulation, the camera spectrum can be directly set to the CIE D65 illuminant [?] using the syntax:

```
<source
  POWER="1"
  NAME="source0"
  TYPE="sun"
  SPECTRUM="d65"
>
<pos X="0" Y="0" Z="0"></pos>
</source>
```

3.8 Path Algorithms

Three path algorithms are available: direct path, simple reverse path and In the scene XML description, the choice of algorithm is indicated by the attribute "ALGORITHM" of the XML object "Scene". It is set to "direct" for direct path algorithm and "reverse_1" or "reverse_2" for reverse path algorithm.

3.8.1 Direct path

For this algorithm the paths start at the sources. All the surfaces of the scene are considered as sensors. When the path is absorbed by a surface (albedo <1) the Monte-Carlo weight is incremented by 1. The surfaces with no material, dielectric or having an albedo of 1 will always get a resulting weight of 0. The results are in the file "sensors.csv" as a list of the weights and errors for each surface.

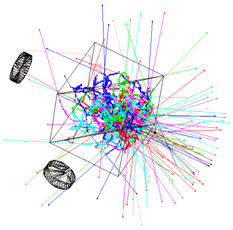

Figure 5: Example of simulation using direct path (see example "photodiodes")

3.8.2 Reverse path

Like the previous one, to use this algorithm, the scene must contain a camera or some sensors ("probes"). The paths start from the probes inside the acceptance cone or at the camera lens aperture in the direction given by each pixel. The available source type for this algorithm is "spot" or "surface". To accelerate Monte-Carlo convergence for these unextended sources, at each photon scattering inside the volumes, the Monte-Carlo weight is incremented by the source contribution along the direct path from the source to the scattering point. Note that for the moment this algorithm assumes that the path from the scattering point to the source is not deviated, which means that the refractive index of the volume is set to 1. If the index is higher than 1 the rendered image gives a realistic result but not quantitatively exact. The output images files are in png format and raw binary format (see annexes for the detail of this format).

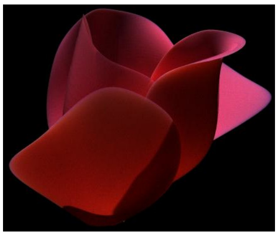

Figure 6: Example of rendering using reverse path

4 Examples

4.1 Example "minimal-cube"

This example illustrates a monochromatic render of a diffusive cube illuminated by the sun.

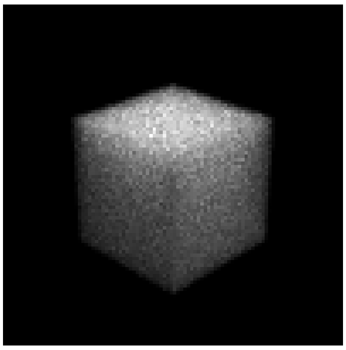

Figure 7: output of the example minimal-cube

```
<?xml version="1.0"?>
<Scene
NB_THREADS="1"
ALGORITHM="reverse_1"
VERBOSE="1"
NB_PHOTONS="100"
STORE_TRAJECTORIES="0"
STORE_ERROR_TRAJECTORIES="0"
MAX_PATH_LENGTH="1000"
MAX_NB_SCATTERING="-1"
>
<source
NAME="sourceSun0"
TYPE="sun"
RADIANCE="1"
SPECTRUM="red"
VOLUME="World"
>
<dir X="0.2" Y="0.3" Z="-1.0"></dir>
</source>
<spectrum
NAME="red"
DATA="650 1"
>
</spectrum>
<camera
NAME="cam0"
FOCAL="1"
NA="0"
NBPIXELSX="100"
NBPIXELSY="100"
GAMMA="1"
FIELD="30.0"
FILTERS="filterR"
VOLUME="World" >
<lens X="-3.0"  Y="-3.0" Z="3.0"></lens>
<viewPoint X="0.5" Y="0.5" Z="0.55"></viewPoint>
</camera>
<spectrum
NAME="filterR"
DATA="650 1"
>
 </spectrum>
<surface
NAME="cube_surface"
MATERIALS="dielectric1"
FILE="cube.obj"
>
</surface>
<dielectric NAME="dielectric1" > </dielectric>
<volume
NAME="cube_volume"
N="1"
MATERIALS="hg2"
SURFACES="cube_surface"
>
</volume>
<Henyey-Greenstein NAME="hg2"
LA="10000000"
LSTAR="0.1"
G="0"
>
</Henyey-Greenstein>
</Scene>
```

4.2 Example "minimal-cube-RGB"

This example illustrates a RGB color render. The source (sun) spectrum is white: same power for red, green and blue.

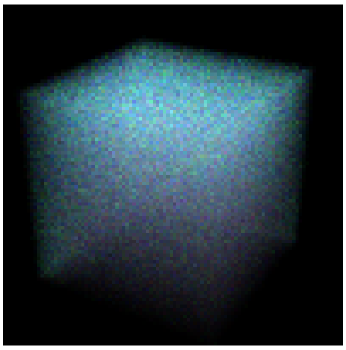

Figure 8: image rendered of the example minimal-cube with 3 wavelengths

```
<?xml version="1.0"?>
<Scene
NB_THREADS="6"
ALGORITHM="reverse_1"
VERBOSE="1"
NB_PHOTONS="100"
STORE_TRAJECTORIES="0"
STORE_ERROR_TRAJECTORIES="0"
MAX_PATH_LENGTH="1000"
MAX_NB_SCATTERING="10000000"
>
<source
NAME="sourceSun0"
TYPE="sun"
SPECTRUM="white"
VOLUME="World"
>
<dir X="0.2"  Y="0.3" Z="-1.0"></dir>
</source>
<camera
NAME="cam0"
VOLUME="World"
NBPIXELSY="100"
NBPIXELSX="100"
GAMMA="1"
FIELD="40"
FILTERS="650 530 450">
<lens X="3.0"  Y="2.0" Z="2.0"></lens>
<viewPoint X="0.5" Y="0.5" Z="0.5" ></viewPoint>
</camera>
<spectrum
NAME="white"
DATA="450 1 530 1 650 1"
>
</spectrum>
<volume
NAME="cube_volume"
N="1"
MATERIALS="hg1"
SURFACES="cube_surface"
>
</volume>
<surface
NAME="cube_surface"
MATERIALS="dielectric1"
FILE="cube.obj"
>
</surface>
<dielectric NAME="dielectric1" > </dielectric>
<Henyey-Greenstein
NAME="hg1"
LA="la"
LSTAR="lstar"
G="0"
>
</Henyey-Greenstein>
<spectrum
NAME="lstar"
DATA="450 0.5  530 0.5  650 1"
>
</spectrum>
<spectrum
NAME="la"
DATA=" 450 5   530 2.5   650  2.5  "
>
</spectrum>
</Scene>
```

4.3 Example "minimal-cube-spectral"

This example illustrates a render with multi-spectral (7 wavelengths) material properties. The source (a sun) spectrum is the standard D65 spectrum.

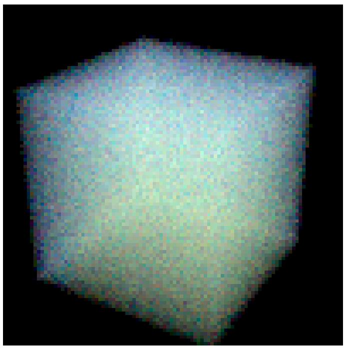

Figure 9: image rendered of the example minimal-cube with 7 wavelengths

```
<?xml version="1.0"?>
<Scene
NB_THREADS="6"
ALGORITHM="reverse_1"
VERBOSE="1"
NB_PHOTONS="100"
STORE_TRAJECTORIES="0"
STORE_ERROR_TRAJECTORIES="0"
MAX_PATH_LENGTH="1000"
MAX_NB_SCATTERING="-1"
>
<source
NAME="sourceSun0"
RADIANCE="1.0"
TYPE="sun"
SPECTRUM="d65"
ANGLE="0"
VOLUME="World"
>
<dir Z="-1.0" Y="0.3" X="0.2"></dir>
</source>
<spectrum
NAME="white"
DATA="400 1 450 1  500 1  550 1  600 1  650 1  700 1 "
>
</spectrum>
<camera
NAME="cam0"
VOLUME="World"
NBPIXELSY="100"
NBPIXELSX="100"
GAMMA="1"
FIELD="40"
FOCAL="1"
NA="0"
FILTERS2="700 650  600  550  500  450 400 "
FILTERS="400 450  500  550  600  650 700 "
FILTERS1="650 530 450"
 >
<lens X="3.0"  Y="2.0" Z="2.0"></lens>
<viewPoint Z="0.5" Y="0.5" X="0.5" ></viewPoint>
</camera>
<volume
NAME="cube_volume"
N="1"
MATERIALS="hg1"
SURFACES="cube_surface"
>
</volume>
<surface
NAME="cube_surface"
MATERIALS="dielectric1"
FILE="cube.obj">
</surface>
<dielectric NAME="dielectric1" > </dielectric>
<Henyey-Greenstein
NAME="hg1"
LA="la"
LSTAR="lstar"
G="g"
>
</Henyey-Greenstein>
<spectrum
NAME="lstar"
DATA="400 1  450 1.5  500 2  550 2.5  600 3  650 3.5  700 4 "
>
</spectrum>
<spectrum
NAME="la"
DATA="400 1.0 450 1.5  500 20  550  25  600 30  650 35   700 40 "
>
</spectrum>
<spectrum
NAME="g"
DATA="400 0  450 0  500 0  550 0  600 0  650 0   700 0 "
>
</spectrum>
</Scene>
```

4.4 Example "minimal-cube" illuminated by a spot source

This example illustrates a render of a cube illuminated by a spot, using the reverse path algorithm.

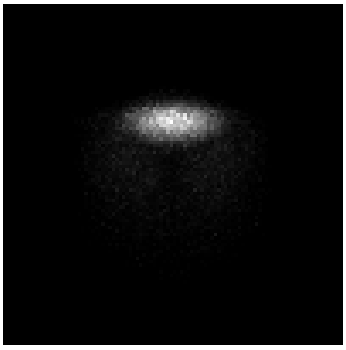

Figure 10: image rendered of a cube illuminated by a spot

```
<?xml version="1.0"?>
<Scene
NB_THREADS="1"
ALGORITHM="reverse"
VERBOSE="1"
NB_PHOTONS="100"
STORE_TRAJECTORIES="0"
STORE_ERROR_TRAJECTORIES="0"
MAX_PATH_LENGTH="1000"
MAX_NB_SCATTERING="-1"
>
<source
NAME="sourceSpot0"
TYPE="spot"
RADIANCE="1"
DIAMETER="0.5"
ANGLE="20"
SPECTRUM="red"
VOLUME="World"
>
<pos X="0.5" Y="0.5" Z="2.0"></pos>
<dir X="0" Y="0" Z="-1.0"></dir>
</source>
<spectrum
NAME="red"
DATA="650 1"
>
</spectrum>
<camera
NAME="cam0"
FOCAL="1"
NA="0"
NBPIXELSX="100"
NBPIXELSY="100"
GAMMA="1"
FIELD="30.0"
FILTERS="filterR"
VOLUME="World" >
<lens X="-3.0"  Y="-3.0" Z="3.0"></lens>
<viewPoint X="0.5" Y="0.5" Z="0.55"></viewPoint>
</camera>
<spectrum
NAME="filterR"
DATA="650 1"
>
 </spectrum>
<surface
NAME="cube_surface"
MATERIALS="dielectric1"
FILE="cube.obj"
>
</surface>
<dielectric NAME="dielectric1" > </dielectric>
<volume
NAME="cube_volume"
N="1"
MATERIALS="hg2"
SURFACES="cube_surface"
>
</volume>
<Henyey-Greenstein NAME="hg2"
LA="10000000"
LSTAR="0.1"
G="0"
>
</Henyey-Greenstein>
</Scene>
```

4.5 Example "minimal-cube" illuminated by 2 spherical sources

This example illustrates a render of a cube illuminated by 2 spherical sources, using the reverse path algorithm.

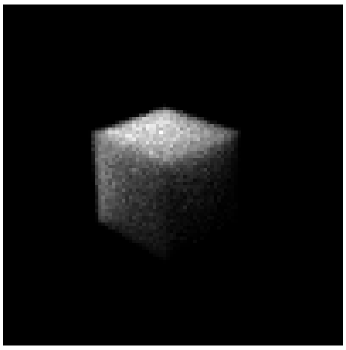

Figure 11: image rendered of a cube illuminated by 2 spherical sources

```
<?xml version="1.0"?>
<Scene
NB_THREADS="1"
ALGORITHM="reverse"
VERBOSE="1"
NB_PHOTONS="100"
STORE_TRAJECTORIES="0"
STORE_ERROR_TRAJECTORIES="0"
MAX_NB_EVENTS="1000"
>
<source
NAME="source1"
TYPE="surface"
POWER="1.0"
FILE="source.obj"
ANGLE="180"
SPECTRUM="red"
VOLUME="World"
>
</source>
<spectrum
NAME="red"
DATA="650 1
">
</spectrum>
<camera
NAME="cam0"
NBPIXELSY="100"
NBPIXELSX="100"
GAMMA="1"
FIELD="40.0"
FOCAL="1"
NA="0"
FILTERS="filterR"
VOLUME="World"
>
<lens X="-3.0" Y="-3.0"  Z="3.0"></lens>
<viewPoint X="0.6" Y="0.5" Z="0.5"></viewPoint>
</camera>
<spectrum
NAME="filterR"
DATA="650 1"
>
</spectrum>
<surface
NAME="cube_surface"
MATERIALS="dielectric1"
FILE="cube.obj"
>
</surface>
<dielectric NAME="dielectric1" > </dielectric>
<volume
NAME="cube_volume"
N="1"
MATERIALS="hg2"
SURFACES="cube_surface"
>
</volume>
<Henyey-Greenstein
NAME="hg2"
LA="10000000"
LSTAR="0.1"
G="0"
>
</Henyey-Greenstein>
</Scene>
```

4.6 Example "minimal-cube" illuminated by a laser

This example illustrates a render of a cube illuminated by a laser beam, using the reverse path algorithm. In this case, the reverse path algorithm will get the luminance at each scattering step from the laser. But the luminance is not taken directly from the laser since the laser is just one ray and the luminance obtained would be always zero. In this case, the reverse algorithm will get the lumimance from a secondary source sorted inside the cube volume starting from the laser.

Note that, due to the high dynamic range of th render, we use GAMMA="0.5" in the object camera.

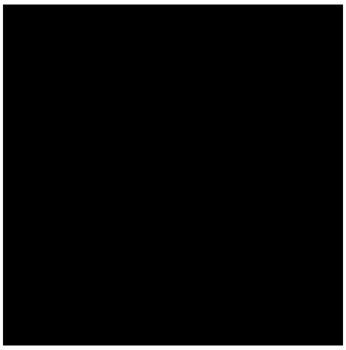

Figure 12: image rendered of a cube illuminated by a laser using the reverse path algorithm

```
<?xml version="1.0"?>
<Scene
NB_THREADS="6"
ALGORITHM="reverse_2"
VERBOSE="1"
NB_PHOTONS="100"
STORE_TRAJECTORIES="0"
STORE_ERROR_TRAJECTORIES="0"
MAX_PATH_LENGTH="1000"
MAX_NB_SCATTERING="-1"
>
<source
NAME="laser0"
RADIANCE="1"
TYPE="spot"
ANGLE="0"
DIAMETER="0.2"
SPECTRUM="red"
VOLUME="World" >
<pos X="0.5" Y="0.5" Z="2"></pos>
<dir X="0" Y="0" Z="-1.0"></dir>
</source>
<spectrum
NAME="red"
DATA="650 1"
>
</spectrum>
<camera
NAME="cam0"
FOCAL="1"
NA="0"
NBPIXELSX="100"
NBPIXELSY="100"
GAMMA="0.5"
FIELD="50.0"
FILTERS="filterR"
VOLUME="World" >
<lens X="-1"  Y="-1" Z="3.0"></lens>
<viewPoint X="0.5" Y="0.5" Z="0.5"></viewPoint>
</camera>
<spectrum
NAME="filterR"
DATA="650 1"
>
 </spectrum>
<surface
NAME="cube_surface"
MATERIALS="dielectric1"
FILE="cube.obj"
>
</surface>
<dielectric NAME="dielectric1" > </dielectric>
<volume
NAME="cube_volume"
N="1"
MATERIALS="hg2"
SURFACES="cube_surface"
>
</volume>
<Henyey-Greenstein NAME="hg2"
LA="10000000"
LSTAR="0.1"
G="0"
>
</Henyey-Greenstein>
</Scene>
```

4.7 Example "photodiodes"

This example illustrates the algorithm called "direct".

A laser starting at (0,0,0) and in the direction of Z illuminates a diffusive cube.

A photodiode called "photodiode1" is placed in front of the laser, at the other side of the cube.

Another photodiode, called "photodiode2" is placed at a side of the cube.

The surfaces objects of names "photodiode1" and "photodiode2" simulate these photodiodes. They have the MATERIALS property set as "black" which means a lambertian surface of "ALBEDO" 0. All the photons that hits these surfaces will end their path here.

To apply the "direct" algorithm, the xml object of type "scene" has its attribute "ALGORITHM" set to "direct".

Its attribute "NB_PHOTONS" indicates that 10000 paths are sorted by the Monte-Carlo.

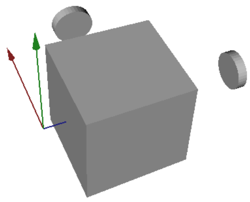

Figure 13: scene view of the diffusing cube and the 2 photodiodes. Viewed by opening the file scene.FCStd with FreeCAD (https://www.freecad.org).

After running starlyx with the command "starlyx scene.xml", check the file "sensors.csv" :

```
lambda(nm): 650
surface_of_laser0_weight 0
surface_of_laser0_sigma 0
photodiode1_weight 0.0051
photodiode1_sigma 0.000712319
photodiode2_weight 0.0017
photodiode2_sigma 0.00041196
cube_weight 0
cube_sigma 0
```

This indicates that, for a laser power of 1, the power received by the photodiode 1 is 0,0051 +/- 0.00071 (sigma) and the power received by photodiode 2 is 0.0017 +/- 0.000412 (sigma). The cube surface has no materials so the paths never end on the cube.

```
<?xml version="1.0"?>
<Scene
NB_THREADS="1"
ALGORITHM="direct"
VERBOSE="1"
NB_PHOTONS="10000"
STORE_TRAJECTORIES="0"
STORE_ERROR_TRAJECTORIES="0"
MAX_PATH_LENGTH="-1"
MAX_NB_SCATTERING="-1"
>
<source
POWER="1.0"
NAME="laser0"
TYPE="spot"
DIAMETER="4"
ANGLE="3"
SPECTRUM="red"
VOLUME="World" >
<pos X="-10" Y="0" Z="7"></pos>
<dir X="1" Y="0" Z="0"></dir>
</source>
<spectrum NAME="red"
DATA="650  1"
>
</spectrum>
<surface NAME="photodiode1"
FILE="photodiode1.obj"
MATERIALS="black"
>
</surface>
<surface NAME="photodiode2"
FILE="photodiode2.obj"
MATERIALS="black"
>
</surface>
<volume  NAME="cube0"
N="1.33"
MATERIALS="hg1"
SURFACES="cube"
>
</volume>
<surface NAME="cube"
FILE="cube.obj"
MATERIALS=""
>
</surface>
<Henyey-Greenstein NAME="hg1"
KS="1"
KA="0"
G="0"
>
</Henyey-Greenstein>
<lambert NAME="black"
ALBEDO="0"
>
</lambert>
</Scene>
```

4.8 Example "sensors"

This example illustrates the algorithm called "reverse path". A cube made with scattering material is illuminated by a sun type light source. Three sensors with acceptance of 4PI steradians are placed inside the cube: one at top, one at middle and one at bottom. The results shows how irradiance decreases when going from top to bottom.

```
reverse_path
sensor1_channels: 650 530 450
sensor1_weight 975768 1.27978e+06 663658
sensor1_sigma 296807 616688 147690
sensor2_channels: 650 530 450
sensor2_weight 975768 1.27978e+06 663658
sensor2_sigma 296807 616688 147690
sensor3_channels: 650 530 450
sensor3_weight 975768 1.27978e+06 663658
sensor3_sigma 296807 616688 147690
```

```
<?xml version="1.0"?>
<Scene
NB_THREADS="6"
DIRECT_PATH="0"
VERBOSE="1"
NB_PHOTONS="10000"
STORE_TRAJECTORIES="0"
STORE_ERROR_TRAJECTORIES="0"
MAX_PATH_LENGTH="1000"
>
<!--  sun at zenith  -->
<source NAME="sourceSun0"
TYPE="sun"
SPECTRUM="white"
>
<dir Z="-1.0" Y="0" X="0"></dir>
</source>
<spectrum NAME="white"
DATA=" 450 1 530 1 650  1  ">
</spectrum>
<!--  sensor placed at top inside the cube  -->
<sensor NAME="sensor1"
DIAMETER="0"
ANGLE="360"
FILTERS="650 530 450"
VOLUME="cube_volume" >
<pos X="5" Y="5" Z="9" > </pos>
<viewPoint X="25" Y="5" Z="5" ></viewPoint>
</sensor>
<!--  sensor placed at middle inside the cube  -->
<sensor NAME="sensor2"
DIAMETER="0"
ANGLE="360"
FILTERS="650 530 450"
VOLUME="cube_volume" >
<pos X="5" Y="5" Z="5" > </pos>
<viewPoint X="25" Y="5" Z="5" ></viewPoint>
</sensor>
<!--  sensor placed at bottom inside the cube  -->
<sensor NAME="sensor3"
DIAMETER="0"
ANGLE="360"
FILTERS="650 530 450"
VOLUME="cube_volume" >
<pos X="5" Y="5" Z="1" > </pos>
<viewPoint X="25" Y="5" Z="5" ></viewPoint>
</sensor>
<spectrum NAME="filterR"
DATA=" 650  1 ">
</spectrum>
<surface NAME="cube_surface"
MATERIALS="dielectric1"
FILE="cube.obj">
</surface>
<dielectric NAME="dielectric1" >
</dielectric>
<volume  NAME="cube_volume"
N="1"
MATERIALS="hg2"
SURFACES="cube_surface">
</volume>
<Henyey-Greenstein NAME="hg2"
LA="1000"
LSTAR="10"
G="0.99">
</Henyey-Greenstein>
</Scene>
```

5 Annexes

5.1 Format of raw image file

All data is written in float format (4 bytes), little endian.

Header with 7 floats:

w=image width (float)

h=image height (float)

nbc=number of channels (wavelenghts) (float)

xmin (0) (float)

xmax (1) (float)

ymin (0) (float)

ymax (1) (float)

The pixels data:

w*h*nbc floats with pixel values. Read it with 3 for loops: for (i=0;i<w;i++) for (j=0;j<h;j++) for (k=0;k<nbc;k++)

The wavelenght values:

nbc floats with wavelength values in nm

References

[Mie 1908] Mie, Gustav (1908). Beiträge zur Optik trüber Medien, speziell kolloidaler Metallösungen .Annalen der Physik. 330 (3): 377-445.

[Hen1947] Henyey, L.G., and Greenstein, J.L. (1941), Ap.J.,93, 70.

[Metropolis 1949] Metropolis, N.; Ulam, S. The Monte Carlo Method. Journal of the American Statistical Association 1949, 44 (247), 335–341. https://doi.org/10.1080/01621459.1949.10483310.

[Chandrasekhar 1960] S. Chandrasekhar. 1960. Radiative transfer. Dover Publications.

[Bohren 2010] Bohren, C. F.; Huffmann, D. R. (2010). Absorption and scattering of light by small particles. New York: Wiley-Interscience. ISBN 978-3-527-40664-7.

[Wald 2014] Wald, I., Woop, S., Benthin, C., Johnson, G. S., & Ernst, M. (2014, July). Embree: A kernel framework for efficient cpu ray tracing. ACM Trans. Graph., 33 (4), 143:1 143:8. doi: 10.1145/2601097.2601199

[Piaud] Piaud B. , Eymet V., Forest V., Coustet C.,star-engine, gitlab.com/meso-star/star-engine

[FreeCAD] freecad.org