

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

Laurent Brunel, PhotonLyX

July 10, 2023

Contents

1	Introduction	1
2	Detailed description of the "minimal_cube" example	1
3	User guide	3
3.1	Path Algorithms	3
3.1.1	Direct path	4
3.1.2	Reverse path	4
3.2	Sources	4
3.2.1	Sun	4
3.2.2	Spot	4
3.2.3	Source geometry defined by any extended surface	5
3.3	Sensors	5
3.3.1	Camera	5
3.3.2	Probe	5
3.3.3	Sensor defined by any surface	6
3.4	Volume radiative transfer properties	6
3.4.1	Diffusive medium modelised by Henyey-Greenstein using k , k_a and g	6
3.4.2	Diffusive medium modelised by Henyey-Greenstein using l^* , g and l_a	7
3.4.3	Spherical particle suspension modelised by Mie theory	7
3.4.4	Diffusive medium modelised by a phase function	7
3.5	Surface properties:	7
3.5.1	Lambertian surface: lambert	7
3.5.2	Dielectric interface: dielectric	8
3.5.3	Mirror	8
3.5.4	Surface emission	8
3.6	Spectral data	8
3.6.1	Spectral camera	9
3.6.2	Spectral source	9
4	Examples	10
4.1	Example "minimal_cube_RGB"	10
4.2	Example "minimal_cube_spectral"	10
4.3	Example "spot"	10
4.4	Example "photodiodes"	10
4.5	Example "sensors"	11
5	Annexes	12
5.1	Format of raw image file	12

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[Piaud]) with itself stands on Embree (Intel)[Wald 2014]. 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).

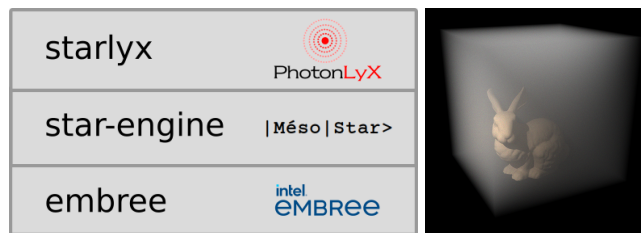


Figure 1: starlyx relies on star-engine (meso-star) and Embree (Intel). Illustrative examples: rendering of the Stanford bunny in a foggy cube.

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:

```
1 cd examples/minimal_cube
2 starlyx scene.xml
```

See the result in image.ppm (figure 2).

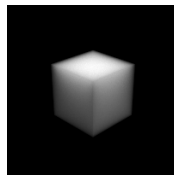


Figure 2: image rendered by the example minimal_cube

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

```
1 <Scene DIRECT_PATH="0" VERBOSE="1" NB_PHOTONS="100"
2 STORE_TRAJECTORIES="0" STORE_ERROR_TRAJECTORIES="0">
```

The attribute DIRECT_PATH is set to 0 to specify that the Monte-Carlo algorithm starts the ray tracing from the sensor (the camera). If DIRECT_PATH is set to 1, 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 or 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":

```
1 <source NAME="sourceSun0" POWER="1.0" TYPE="sun" ANGLE="0.5" SPECTRUM="red">
2 <dir Z="-1.0" Y="0.3" X="0.2"></dir>
3 </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°). 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"

```
1 <camera NAME="cam0" NBPIXELSY="200" NBPIXELSX="200" FIELD="30.0" FILTERS="filterR">
2 <lens Z="3.0" Y="-3.0" X="-3.0"></lens>
3 <viewPoint Z="0.5" Y="0.5" X="0.5"></viewPoint>
4 </camera>
5 <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":

```
1 <surface NAME="cube_surface" MATERIALS="dielectric1" FILE="cube.obj">
2 </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":

```
1 <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 "Henryey-Greenstein".

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

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

```

1 <Henye y - Greenstein NAME="hg2" LA="10000000" LSTAR="0.1" G="0">
2 </Henye y - Greenstein>

```

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

```

1 </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).

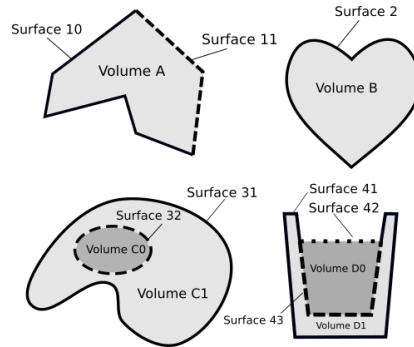


Figure 3: The surfaces delimit the volumes. Volume A is delimited by the surfaces 10 and 11. Volume B is delimited by surface 2. Volume C0 is delimited by surface 32 and volume C1 by surfaces 31 and 32. Volume D0 is delimited by surfaces 42 and 43. Volume D1 by surfaces 41 and 43.

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

3.1 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 "DIRECT_PATH" of the XML object "Scene". It is set to "1" for direct path algorithm and "0" for reverse path algorithm.

3.1.1 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.

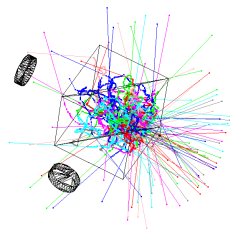


Figure 4: Example of simulation using direct path (see example "photodiodes")

3.1.2 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).

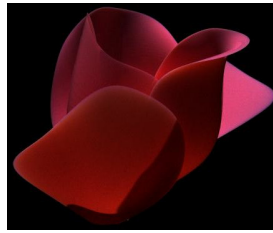


Figure 5: Example of rendering using reverse path

3.2 Sources

3.2.1 Sun

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

```
1 <source NAME="sourceSun0" POWER="1.0" TYPE="sun" ANGLE="0.5" SPECTRUM="red">
2 <dir X="0" Y="0" Z="-1"></dir>
3 </source>
```

3.2.2 Spot

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

```
1 <source NAME="sourceSpot0" POWER="1.0" TYPE="spot" DIAMETER="1" ANGLE="10"
   SPECTRUM="red">
2 <pos X="0" Y="0" Z="0"></pos>
3 <dir X="1" Y="0" Z="0"></dir>
4 </source>
```

3.2.3 Source geometry defined by any extended surface

Here is the 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.

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

3.3 Sensors

3.3.1 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 from 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.

```
1 <camera NAME="cam0" NBPIXELSY="200" NBPIXELSX="200" FIELD="30.0" FILTERS="filterR"  
2     NA="0.2" FOCAL="2" VOLUME="World" >  
3   <lens Z="3.0" Y="-3.0" X="-3.0"></lens>  
4   <viewPoint Z="0.5" Y="0.5" X="0.5"></viewPoint>  
5 </camera>  
6 <spectrum NAME="filterR" DATA="650 1"> </spectrum>
```

3.3.2 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.

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

3.3.3 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.

3.4 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:

```
1 <volume NAME="cube" N="1" MATERIALS="hg1 hg2 mie1 "></volume >
2 <Henyey-Greenstein NAME="hg1" K="1" KA="0.1" G="0" ></Henyey-Greenstein >
3 <Henyey-Greenstein NAME="hg2" LSTAR="1" LA="0.1" G="0" ></Henyey-Greenstein >
4 <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).

```
1 <volume NAME="cube" N="1" MATERIALS="fat casein"></volume >
2 <Mie NAME="fat" D_UM="0.001" NR="1.46" NI="0" PHI="0.05" ></Mie >
3 <Mie NAME="casein" D_UM="0.0001" NR="1.57" NI="0" PHI="0.02"></Mie >
```

3.4.1 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 [Hen1947] and determined by the asymmetry coefficient g. Here is an example of medium with a scattering coefficient $k = 1 \text{ mm}^{-1}$, an asymmetry coefficient $g=0$ and an absorption coefficient $ka = 0.1 \text{ mm}^{-1}$:

```
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 .

3.4.2 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 $l^* = 1 \text{ mm}$, $g=0$ and $la = 0.1 \text{ mm}$.

```
1 <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 .

3.4.3 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 [Mie 1908, Bohren 2010] 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 1 μm :

```
1 <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 .

3.4.4 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 $k = 1 \text{ mm}^{-1}$, a phase function defined by the curve pf1 and an absorption coefficient $ka = 0.1 \text{ mm}^{-1}$:

```
1 <Scattering NAME="mv1" K="1" PHASE_FUNCTION="pf1" KA="0.1" ></Scattering >
2 <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.

3.5 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":

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

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

3.5.1 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 .

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

3.5.2 Dielectric interface: dielectric

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

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

According to the refractive index of the 2 media, light may refract, reflect using the Fresnel laws or totally reflect if 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:

$$n1 \sin(i1) = n2 \sin(i2)$$

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

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

$$R = (R_s + R_p) / 2$$

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

3.5.3 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.

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

3.5.4 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/m². Note: this type of surface is not yet implemented for the direct algorithm (see section ??).

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

3.6 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 " λ_1 v1 λ_2 v2". The unit for wavelengths is nm. Here is an example of RGB like spectrum with twice more power on green than other wavelengths:

```
1 <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.

3.6.1 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".

```
1 <camera NAME="cam0" NBPIXELSY="50" NBPIXELSX="50" GAMMA="1" FIELD="30"
2 FILTERS="650 530 450">
3 <lens X="1" Y="1" Z="1"></lens>
4 <viewPoint X="0" Y="0" Z="0"></viewPoint>
5 </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 6):

```
1 <camera NAME="cam0" NBPIXELSY="200" NBPIXELSX="200" FIELD="30"
2 FILTERS="filterR filterG filterB">
3 <lens X="1" Y="1" Z="1"></lens>
4 <viewPoint X="0" Y="0" Z="0"></viewPoint>
5 </camera>
6 <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>
7 <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 70
   590 30 600 10 610 0 620 0 630 0 640 0 650 0 660 0 670 0 680 0 690 0"> </
   spectrum>
8 <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).

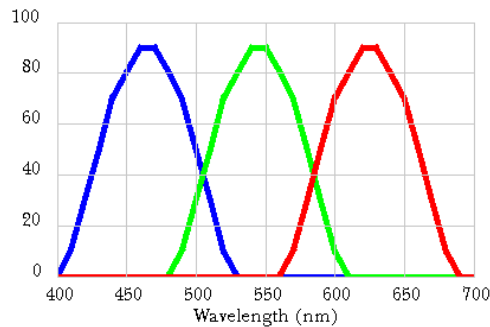


Figure 6: Example of filters for a RGB camera

3.6.2 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":

```

1 <source POWER="1.0" NAME="source0" TYPE="spot" DIAMETER="0" ANGLE="180" SPECTRUM="
  white">
2   <pos X="0" Y="0" Z="0"></pos>
3 </source>
4 <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:

```

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

```

4 Examples

4.1 Example "minimal_cube_RGB"

```

1 <?xml version="1.0"?>
2 <Scene DIRECT_PATH="0" VERBOSE="1" NB_PHOTONS="1000" STORE_TRAJECTORIES="0"
  STORE_ERROR_TRAJECTORIES="0">
3   <source NAME="sourceSun0" POWER="1.0" TYPE="sun" SPECTRUM="white">
4     <dir X="0.2" Y="0.3" Z="-1.0" ></dir>
5   </source>
6   <spectrum NAME="white" DATA=" 450 1 530 1 650 1"> </spectrum>
7   <camera NAME="cam0" NBPIXELSY="300" NBPIXELSX="300" GAMMA="1" FIELD="30" FILTERS="
  650 530 450">
8     <lens X="3.0" Y="2.0" Z="2.0"></lens>
9     <viewPoint X="0.5" Y="0.5" Z="0.5" ></viewPoint>
10  </camera>
11 <volume NAME="cube_volume" N="1" MATERIALS="hg1" SURFACES="cube_surface"></volume>
12 <surface NAME="cube_surface" MATERIALS="dielectric1" FILE="cube.obj"></surface>
13 <dielectric NAME="dielectric1" > </dielectric>
14 <henyey-greenstein NAME="hg1" LA="1a" LSTAR="lstar" G="0"></Henyey-Greenstein>
15 <spectrum NAME="lstar" DATA="450 1 530 1 650 2"> </spectrum>
16 <spectrum NAME="1a" DATA=" 450 10 530 5 650 5 " > </spectrum>
17 </Scene>

```

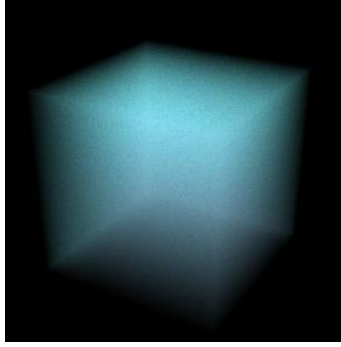


Figure 7: image rendered of the example minimal_cube with 3 wavelengths

4.2 Example "minimal_cube_spectral"

4.3 Example "spot"

4.4 Example "photodiodes"

This example illustrates the algorithm called "direct path". 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 "DIRECT_PATH" set as "1". Its attribute "NB_PHOTONS" indicates that 10000 paths are sorted by the Monte-Carlo.

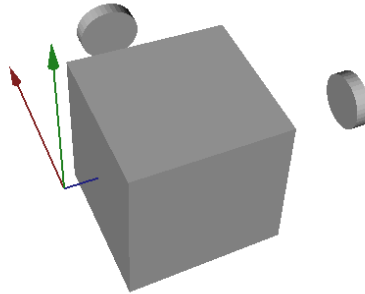


Figure 8: scene view of the diffusing cube and the 2 photodiodes. Viewed by opening the file scene.FCStd with FreeCAD (<https://www.freecad.org>).

```

1 <?xml version="1.0"?>
2 <Scene DIRECT_PATH="1" VERBOSE="0" NB_PHOTONS="10000" STORE_TRAJECTORIES="0"
  STORE_ERROR_TRAJECTORIES="0">
3 <source POWER="1.0" NAME="laser0" TYPE="spot" DIAMETER="0" ANGLE="0" SPECTRUM="red"
  >
4   <pos X="0" Y="0" Z="0"></pos>
5   <dir X="0" Y="0" Z="1"></dir>
6 </source>
7 <spectrum NAME="red" DATA="650 1" > </spectrum>
8 <surface NAME="photodiode1" FILE="photodiode1.obj" MATERIALS="black" > </surface>
9 <surface NAME="photodiode2" FILE="photodiode2.obj" MATERIALS="black" > </surface>
10 <volume NAME="cube0" N="1.33" MATERIALS="mie1" SURFACES="cube" > </volume>
11 <surface NAME="cube" FILE="cube.obj" MATERIALS="" > </surface>
12 <Henyey-Greenstein NAME="mie1" PHI="0.0003" KA="0" D_UM="2" N="2.54" > </Henyey-
  Greenstein>
13 <lambert NAME="lambert1" ALBEDO="1" > </lambert>

```

```

14 <lambert NAME="black" ALBEDO="0" > </lambert>
15 </Scene>

```

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

```

1 name      weight  sigma
2 photodiode1 0.0076  0.00086
3 photodiode2 0.0122  0.00109
4 cube      0.0000  0.00000

```

This indicates that, for a laser power of 1, the power received by the photodiode 1 is 0,0076 +/- 0.00086 and the power received by photodiode 2 is 0.0122 +/- 0.00109. The cube surface has no materials so the paths never end on the cube.

4.5 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.

```

1 Probe probe1
2 Channel:  650 530 450
3 weight  0.154698  0.254962  0.213281
4 sigma  0.0295834  0.0368101  0.0455356
5 Probe probe2
6 Channel:  650 530 450
7 weight  0.0456784  0.0408456  0.113885
8 sigma  0.0174192  0.014925  0.035159
9 Probe probe3
10 Channel:  650 530 450
11 weight  0.0315209  0.00481895  0.00160809
12 sigma  0.0141981  0.00257279  0.000672896

```

```

1
2 <?xml version="1.0"?>
3 <Scene NB_THREADS="1" DIRECT_PATH="0" VERBOSE="1" NB_PHOTONS="20"
4   STORE_TRAJECTORIES="1"
5   STORE_ERROR_TRAJECTORIES="0" MAX_PATH_LENGTH="1000">
6
7 <!-- sun at zenith -->
8 <source NAME="sourceSun0" POWER="1.0" TYPE="sun" SPECTRUM="white">
9   <dir Z="-1.0" Y="0" X="0"></dir>
10 </source>
11 <spectrum NAME="white" DATA=" 450 1 530 1 650 1  "> </spectrum>
12
13 <!-- sensor placed at top inside the cube -->
14 <probe NAME="probe1" DIAMETER="0" ANGLE="360" FILTERS="650 530 450" VOLUME="
15   cube_volume" >
16   <pos X="5" Y="5" Z="9" > </pos>
17   <viewPoint X="25" Y="5" Z="5" ></viewPoint>
18 </probe>
19 <!-- sensor placed at middle inside the cube -->
20 <probe NAME="probe2" DIAMETER="0" ANGLE="360" FILTERS="650 530 450" VOLUME="
21   cube_volume" >
22   <pos X="5" Y="5" Z="5" > </pos>
23   <viewPoint X="25" Y="5" Z="5" ></viewPoint>
24 </probe>
25 <!-- sensor placed at bottom inside the cube -->
26 <probe NAME="probe3" DIAMETER="0" ANGLE="360" FILTERS="650 530 450" VOLUME="
27   cube_volume" >
28   <pos X="5" Y="5" Z="1" > </pos>
29   <viewPoint X="25" Y="5" Z="5" ></viewPoint>
30 </probe>
31

```

```

32 <spectrum NAME="filterR" DATA=" 650 1 "> </spectrum>
33
34 <surface NAME="cube_surface" MATERIALS="dielectric1" FILE="cube.obj">
35 </surface>
36
37 <dielectric NAME="dielectric1" > </dielectric>
38
39 <volume NAME="cube_volume" N="1" MATERIALS="hg2" SURFACES="cube_surface">
40 </volume>
41
42 <Henyey-Greenstein NAME="hg2" LA="100" LSTAR="1" G="0"></Henyey-Greenstein>
43
44 </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 (wavelengths) (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 wavelength 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](https://www.freecad.org)