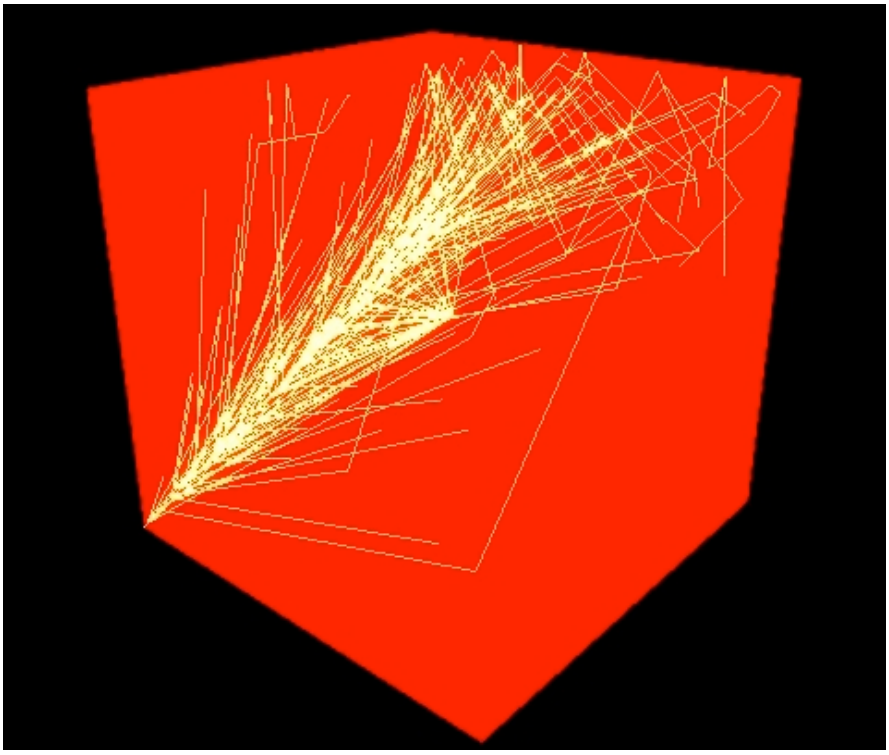
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PHOTONSIM: DEVELOPMENT OF A MONTE CARLO RAY TRACING SOFTWARE FOR THE SIMULATION OF SOLAR CONCENTRATORS.



André Kostro
Masters Thesis in Computer Science
LESO-PB, EPFL 2006.
Supervisors: Jean-Louis Scartezzini, Andreas Schüler.

TABLE OF CONTENTS

PHOTONSIM

Abstract

PhotonSim is a photon ray tracing tool designed to estimate the performance of materials to concentrate daylight. This concentration is possible due to specific optical properties of certain photo-luminescent materials such as organic colorants or nanocrystalline semi-conductors (quantum dots). The software is a Monte Carlo simulation whose inputs may be either experimental or theoretical values for the material and the light source characteristics. Inverse functions are then used to generate the random events such as reflection-refraction and absorption-reemission according to the laws of physics.

Applications

Possible applications are in day-lighting and Photovoltaics by using specific materials to concentrate light effectively no matter if it is diffuse or direct. This would offer a solution to bypass Liouville's theorem which ensures that brilliance is a property of the source and makes concentration of diffuse light difficult. An existing technic for this uses CPC (compound parabolic concentrator).

Development

The software, implemented in C++ and using the openGl © libraries, features 2D and 3D visualization of the trajectories. The results and informations about the photons (quantum yield, wavelength, number of reemissions...) may be plotted graphically, printed on the screen and/or to a file. It is object oriented and aimed at being flexible and reusable.

Existing ray tracing software is often dedicated to lighting design, architecture, or computer animation in 3D scene rendering. The more physical approach of PhotonSim makes it complementary to this type of existing software such as Radiance which is used at greater scales (windows, rooms, objects ...) in architecture or design.

Outline

In the first part, the considered problem and the physical laws associated the the behavior we want to simulate will be presented. Second, the resulting software will be presented. Last but not least, a step by step verification, some results and a validation through experimental data will be presented.

FIRST PART

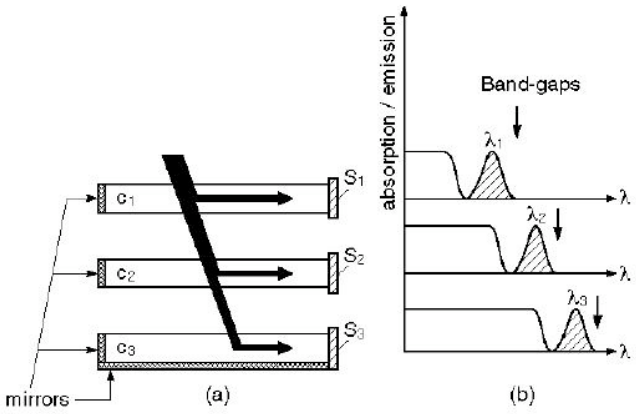
Aims, means: the model, Ray Tracing and Monte Carlo.

The aim is to simulate the behavior of light in materials with specific properties and offer a flexible tool for the design of concentrators with various photo-luminescent characteristics, proportions and layouts. Materials of particular interest are organic colorants and quantum dots which have the capacity to absorb and reemit light and thus can be used to concentrate both direct and diffuse light. The following part will introduce the object of the simulation: planar luminescent solar collectors. Their behavior will be presented briefly and a resulting model will be proposed as well as the means to realize it.

1.1. Planar Luminescent Solar Collectors

1.1.1 Götzberger's concentrators

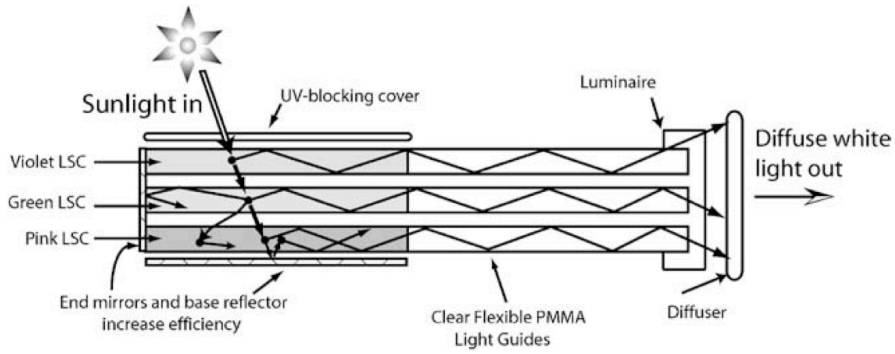
In the 1970s Götzberger [1] proposed layered fluorescent planar concentrators using the optical proprieties of organic colorants to concentrate light:



Stack of planar fluorescent solar collectors.

Each collector C absorbs a specific wavelength and is associated to a particular solar cell S optimized for the emitted wavelength. The first layer could absorb violet light: wavelengths until 400 *nm*, reemit around 450 and therefore concentrate blue light but let superior wavelengths pass. The next layer could have the same behavior switched by 100 *nm* and concentrate green light. Finally the third layer could reemit around 550 *nm* and concentrate red light. The non absorbed light is reflected back through the stack to increase the absorbed proportion of light and therefore the reemitted, collected light. This makes it possible to use less solar cell surface with more efficiency.

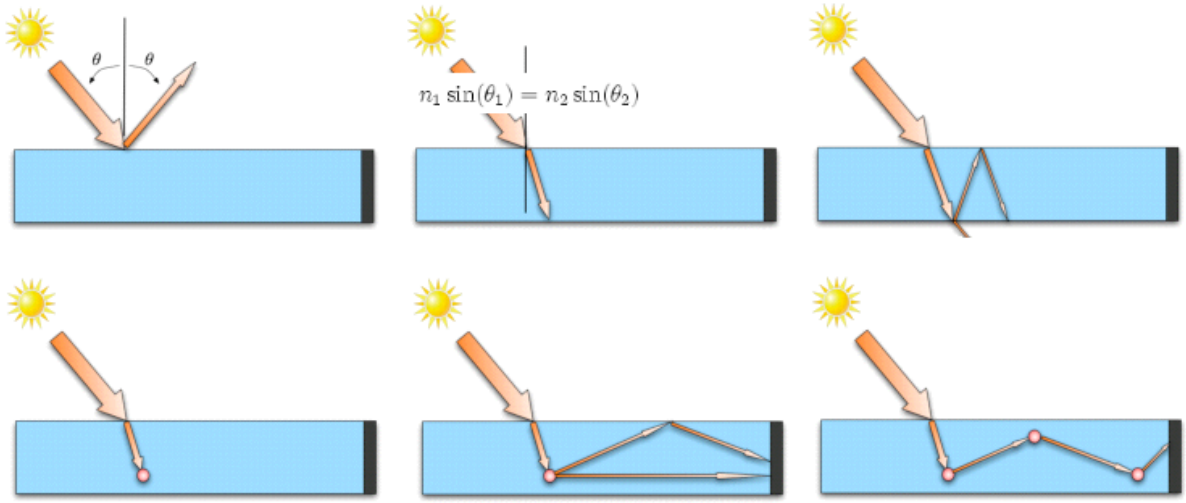
An other use of stacked fluorescent materials working using based on the same principles was imagined by Earp [2] at the technical university of Sydney to be used in day-lighting:



The major drawback of these collectors is the photobleaching, the fluorescent dies degrade when they are exposed to UV light. For this reason it would be preferable to use quantum dots in such concentrators.

1.1.2 Insight, Physical behavior

If some materials are luminescent it can be explained by the behavior of light inside them. Due to classical optic proprieties of the materials, light may be reflected at the interface with air and not enter the collector. If it is not reflected, light will be refracted inside the collector with a specific angle determined by Snell's law. Once inside the collector more specific luminescent proprieties come into play and light may be absorbed. If it is not, it traverses the material, arrives at the bottom interface where it may again be reflected or refracted. The absorbed light can be reemitted, once or several times. The following figure illustrates those events.



Different events inside a luminescent solar collector. From left to right and top to bottom: reflection, refraction, internal reflection, absorption, reemission, multiple reemission.

1.2 Concepts of the simulation

PhotonSim is a Monte Carlo ray tracing software, those two concepts will now be introduced.

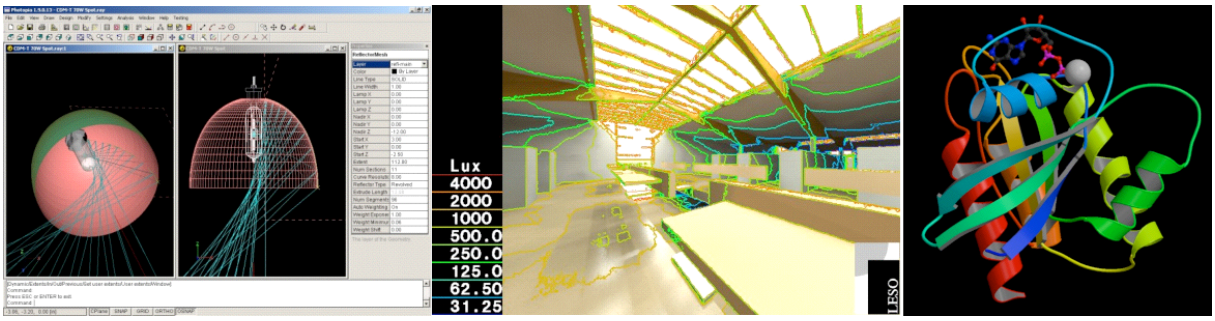
1.2.1 Ray Tracing

Ray tracing is a technique from geometrical optics to model the path taken by light in an environment by following rays of light. It may be use in the design of optical devices such as lenses and sets of lenses in microscopes, telescopes etc... It is also an approach for graphical rendering in computer graphics and produces realistic illumination of virtual scenes. Virtual reality is a large field of application for ray tracing where it is used in flight, surgery and other simulators to give an extra touch of reality. In many cases, reverse ray tracing is applied following rays from the eye-point outwards. Because the aim is to render a scene, rays that do not reach the field of view are of no interest and ignoring them increases performances.

There is a lot of existing ray tracing rendering software, more or less physically accurate, to simulate the behavior of light in complex environment. Some are aiming at producing impressive realist images (OpenRT, POV-ray) or at rendering scenes fast for real time applications such as games (RealStorm).

Screenshots from OpenRT, POV-ray and other ray tracing rendering software

Programs such as Radiance are complete and flexible tools for the use of architects that make it possible to model buildings with different types of windows containing furniture with different colors and textures and to illuminate the scene with a defined light, be it natural or artificial, direct or indirect. Others concentrate on a larger scale and are good at simulating towns or on a lower scale to design complex luminaries (Photopia) or optics (OSLO). Finally there are simple ray tracing tools used in scientific software for visualization, this is the case of Raster3D used for the rendering of proteins and molecules in Molscript.



Screenshots from Photopia, Radiance and Molscript.

An other group of ray tracing software is designed to simulate physics. A large number of those are used at CERN for the simulation of particles trajectories. These simulation are a lot more realistic at modeling the physical behavior of particles (electrons, muons ...) and their interaction with each other and with the medium they are traveling in[10].

This is the sort of simulation that is aimed at with PhotonSim. But the tools developed at CERN are aimed at simulating particles, their collisions and the chain of events following and are not appropriate for the our purpose. Furthermore because we consider quantum effects of light in very specific materials that have the capability to absorb and reemit light we need to be more accurate about what happens at each step off time and cannot use existing rendering tools. The major difference between the wide possibilities those offer and the present goals is that they model neither the absorption and reemission of light nor the dependence on wavelength. An other difference is the scale of the simulation. Existing software renders complicated scenes with many objects the light can interact with in a simple way, PhotonSim on the other hand simulates the behavior of light in very simple geometries (rectangles) but with extended accuracy.

Because we want to estimate the performance of a configuration, we need to know in detail what happens to the light hitting the collector. We are interested not only in rays that arrive at a given viewpoint but in all rays weather they are reflected, transmitted, captured or absorbed. It is important to have a good characterization of the sample, both about what appears to happen from outside and the events inside. This suggests the use of forward ray tracing.

This ray tracing approach gives the photons a particle's behavior rather then that of a wave. The photons are given a position and a direction and the future position is defined by the current position, the direction and the traveled distance.

$$\vec{P}_{i+1} = \vec{P}_i + dist \cdot \vec{v}$$

1.2.2 Light beams versus single Photons

PhotonSim simulates light beams by a single photon approach. This choice was made because as it will be explained in the next section, a light beam is partially reflected and partially refracted at interfaces, similarly it may be partially absorbed and partially reemitted in a medium. A light beam could be simulated by tracing it until the next interface, compute the reflected part and the refracted part and then trace the two resulting beams, reduced in intensity. This process would have to be repeated at each intersection for each resulting beams creating an exponentially growing number of beams to track. This is complicated and often involves a lower threshold for intensity of tracked beams and some beans are discarded. When absorption and reemission are considered, the complexity of the problem grows further. At each reemission, the beam is reemitted in several directions. This makes the above described "beam split" approach unsuitable and for this reason photons will be "shot" one by one rather than in beams. In this approach a single photon arriving at an interface may be refracted OR reflected with a particular probability and similarly, the photon may be absorbed at a random point in the medium and then reemitted in a random direction. As it will be explained shortly, numerous of those events are quantum effects and depend on the wavelength of the light, this suggests the use of a Monte Carlo algorithm where the wavelength of each incoming photon and the occurrence of numerous quantum events in the materials are drawn randomly.

1.2.3 Monte Carlo

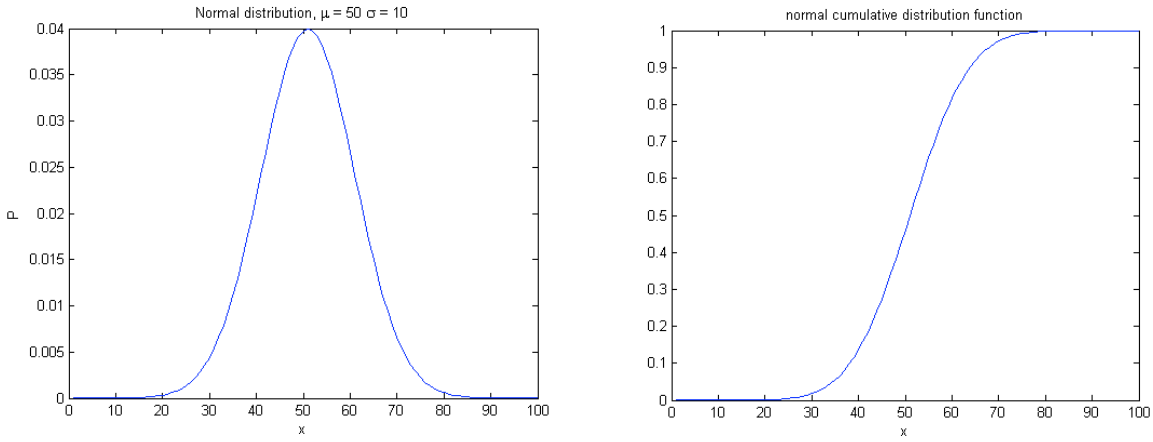
Monte Carlo algorithms are stochastic and are used to solve complex physical or mathematical problems. When the number of variables is great and the solution is complex, exploring the solution space or events space randomly can give an accurate estimate of the solution. In a typical Monte Carlo algorithm, random draws following given distributions define a chain of local events characterizing the global even and leading to a final state. By repeating this numerous times an approximation of the solution is obtained. The accuracy of this solution depends on how well the problem is modeled and how many draws are made. A trivial example would compute the probability to draw two sixes with three dices and a jack out of a 52 card set. This probability can be estimated by Monte Carlo method drawing three random numbers between one and 1 and 6 and one between 1 and 13 numerous times. The probability would be given by the count of the number of drawn combinations with at least two sixes and a jack, divided by the total number of draws.

Complex application include integral calculus, electron trajectory simulation for calibration of electron microscopy (CASINO) [3] or modeling of light transport in multi-layered tissues for biological application (MCML) [4]. MCML is the most similar application to PhotonSim found. It models multiple layers, refraction and reflection as well as absorption and scattering. However it is limited in precision by a grid size, simulates light by "packets" issued from a single unidirectional beam and most importantly

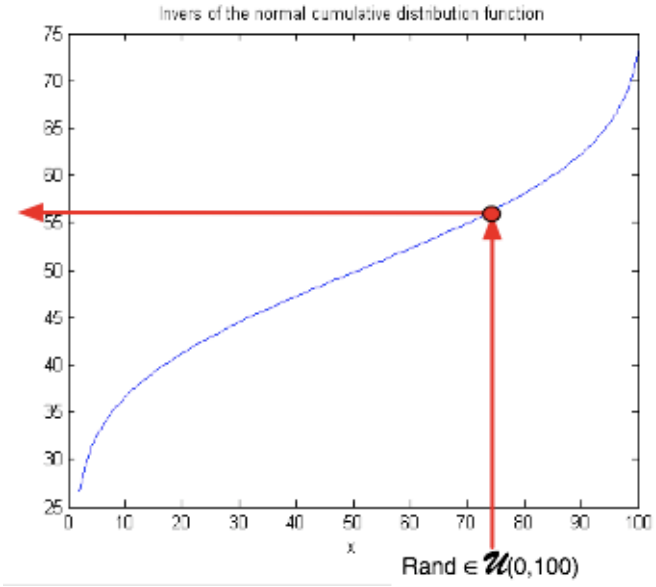
does not consider the wavelength of light. As we it was already introduced and will be explained in more details later, the wavelength of a photon is crucial to its behavior in luminescent collectors.

1.2.4 Inverse Function

The inverse function is a central method to Monte Carlo simulation. It is used to generate random number distributed as per a given function. First the distribution has to be integrated an then inverted. Drawing a uniformly distributed number between 0 and 1 the inverse function will yield a random number distributed after the original distribution function. The following figures illustrates this with a normal distribution.



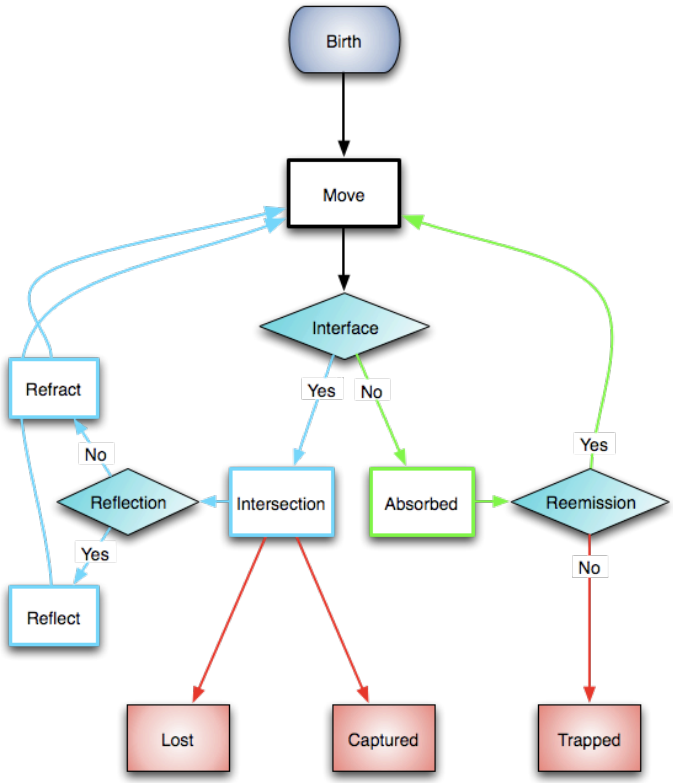
Probability distribution function (pdf) for a gaussian, cumulative probability distribution (cdf) for a gaussian (the integral of the pdf)



The inverse cdf for a gaussian and how to draw a random number from the given pdf.

1.3 Modeling light and its interactions with the medium

As stated above, PhotonSim models the desired conditions by randomly drawing photons and the events on their path. The following graph resumes the possible chains of events in a photons "life", the initial state is "*Birth*", red rectangles represent final states and rhombuses symbolize decisions.



The state machine of a photons "life": After its birth, a photon has a determined direction and travels a certain distance. If it does not strikes an interface along its path it will be absorbed, once absorbed it can be trapped and the next photon is shot. If reemitted, it will travel further and may hit an interface. At an interface the photon may either be reflected or refracted and set into motion in the corresponding direction. An interface can also mean the photon quits the system or is detected.

Those states in a photon's "life" will now be described in greater detail.

1.3.1 Birth of a Photon, the Source.

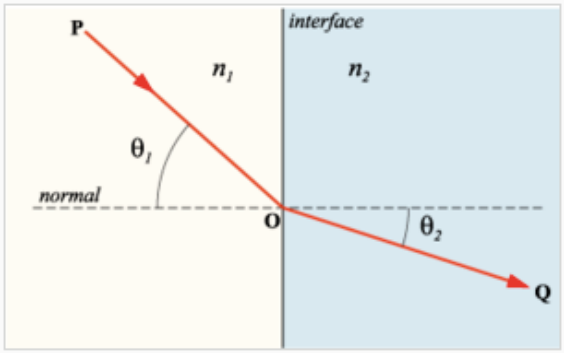
The first thing that has to be determined about a photon are it's characteristics as it hits the simulated collector. The wavelength and orientation follow specified spectrum and distribution and are drawn randomly using the inverse function. It is possible for the experimentalist to have access to some predefined standard skies or to specify the spectrum and angular distribution. The available standard skies are:

1. Clear sky diffuse light only. The spectrum used is the AM1.5 spectrum [8] for diffuse light from the National Renewable Energy Laboratory (NREL) is used with a Mooon and Spencer distribution.
2. Clear Sky, direct light only. The AM1.5 spectrum for direct light from the NREL is used with a defined angle cone in which the rays are distributed uniformly.
3. Covered Sky. The D65 Spectrum extended by the spectrum of a black body and with angles distributed along the Mooon and Spencer distribution.

1.3.2 Reflection and Refraction

Refraction is a well-known phenomenon making light "bend" at the interface between two different mediums. It can be observed as a straw is half immersed in water and appears bended. The angle of refraction of light is defined by Snell's law:

$$n_1 \sin(\theta_1) = n_2 \sin(\theta_2)$$



Refraction of light at the interface between two media of different refractive indices, with $n_2 > n_1$.

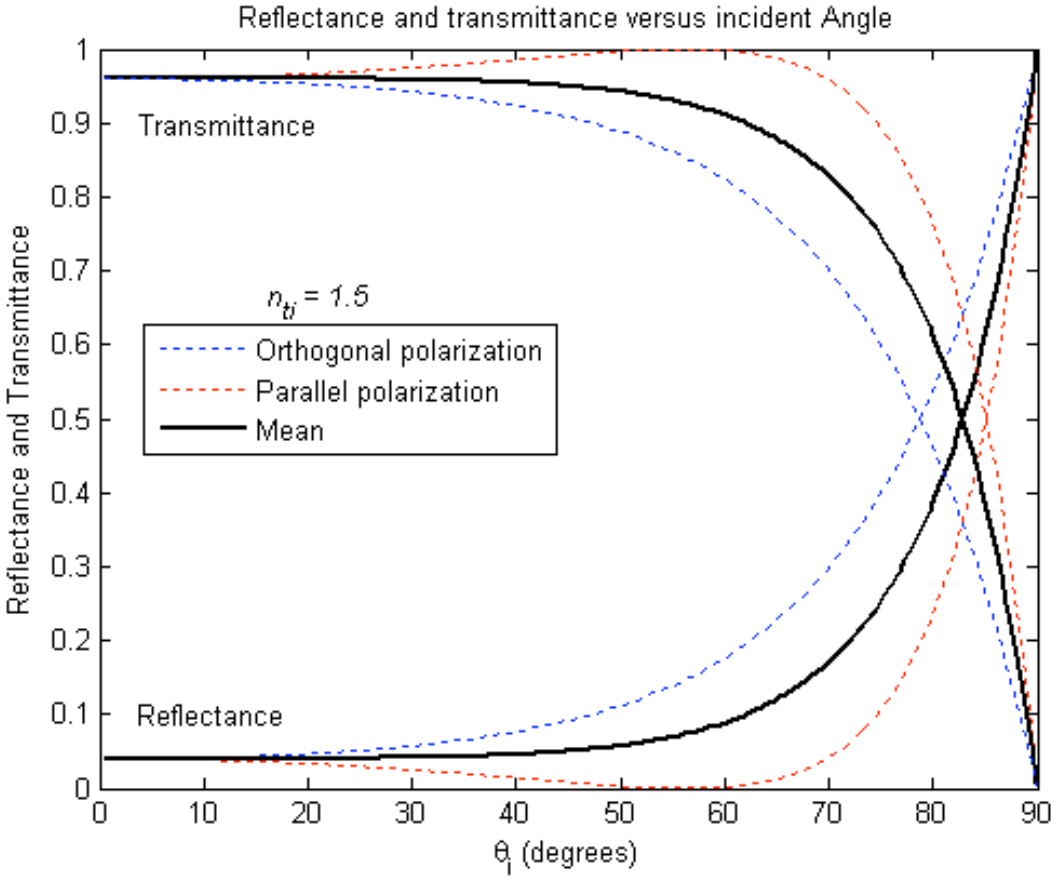
However the values for the refraction indices are not constant and depend on the wavelength. Therefore each material should define values for the refraction index at different wavelengths. This defines the refracted angle but as it is stated buy Fresnel's equations, refraction does not occur for the totality of the incident beam. A part of this beam is reflected an other part is refracted, these proportions depend on the incident angle and the refractive index of the materials at each side of the interface. The proportions also

depend on the polarization of light but as the simulation does not take polarization into account yet, the mean value between orthogonally and parallel polarized light will be used. The values of Fresnel's equations yield the proportions of reflected and refracted light in a beam depending on the incoming angle. Because we consider only a single photon at a time, a random number between 0 and 1 is drawn at every intersection with an interface and if it is greater then the reflected percentage of light R, the photon is refracted, else it is reflected.

$$R_{\perp} = r_{\perp}^{\,2} = \left(\frac{\sin(\theta_i - \theta_t)}{\sin(\theta_i + \theta_t)} \right)^2 \qquad T_{\perp} = 1 - R_{\perp}$$

$$R_{\parallel} = r_{\parallel}^{\,2} = \left(\frac{\tan(\theta_i - \theta_t)}{\tan(\theta_i + \theta_t)} \right)^2 \qquad T_{\parallel} = 1 - R_{\parallel}$$

$$\overline{R} = \left(\frac{R_{\parallel} + T_{\perp}}{2} \right) \qquad \overline{T} = 1 - \overline{R}$$

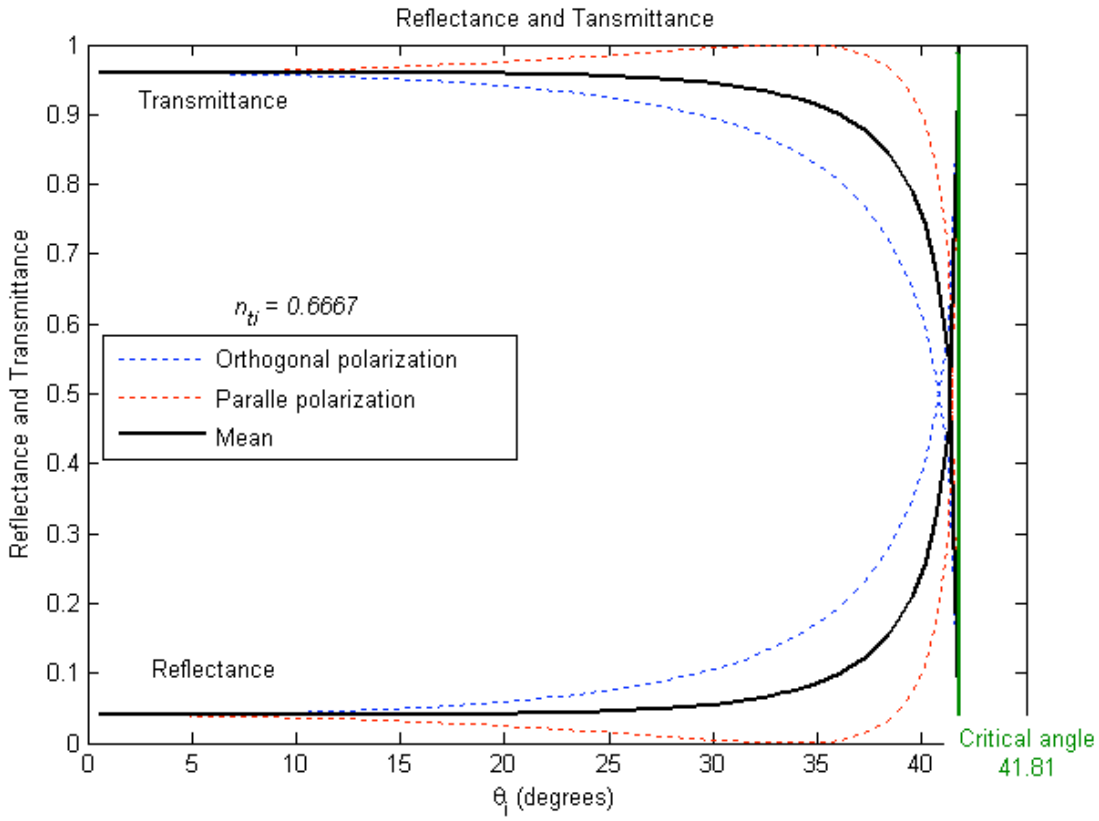


Reflectance and transmittance values depending on the incoming angle for different polarization with incoming refractive index = 1 and transmitted refractive index = 1.5.

A remarkable angle is the critical angle above which there is total internal reflection. This angle exists only if the refraction index of the incident material is greater then the one of the material beyond the interface:

$$\theta_{\text{crit}} = \arcsin\left(\frac{n_2}{n_1}\right)$$

Above this critical angle, the reflectance value is equal to one and the transmittance inexistent.



Reflectance and transmittance values for different polarization with incoming refractive index = 1.5 and transmitted refractive index = 1 . Above the critical angle, there is total internal reflection, no more light is transmitted.

1.3.3. Absorption

A central phenomenon that has to be modeled is the absorption of light by the material. This phenomenon is essential to simulate luminescent collectors but is not exclusive to them and happens in many materials. The intensity of a light beam diminishes exponentially with the distance in the material [6].

$$I(x) = I_0 e^{-\alpha x}$$

Were *alpha* is the coefficient of absorption and depends on the material and the wavelength of incoming light. It doubly depends on wavelength as k, depends on lambda :

$$\alpha \equiv 2\omega n_i / c$$
$$\Leftrightarrow \alpha = 4\pi n_i / \lambda$$
$$n_i = k$$

This law defines the intensity at a given distance inside the material and it can also be considered as the probability distribution function for the penetration depth inside the material. For each photon we can then use the inverse function to generate a random penetration depth inside the medium:

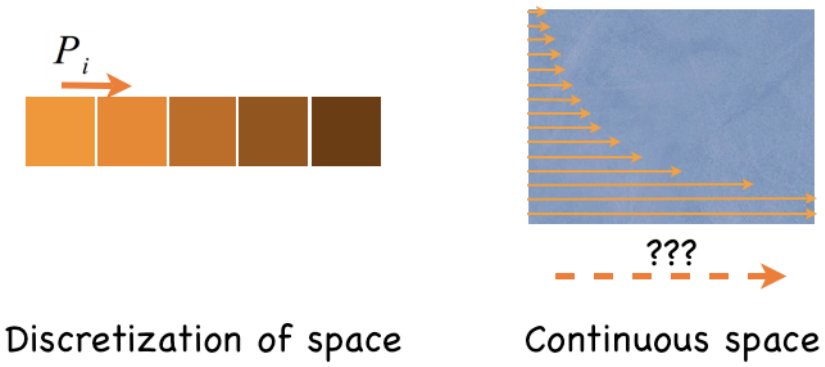
$$i(x) = i_0 e^{-\alpha x}$$

$$\int_0^d i(x)\,dx = \left[\frac{-e^{-\alpha x}}{\alpha}\right]_0^d$$
$$\Leftrightarrow I(d) = \frac{\left(1 - e^{-\alpha d}\right)}{\alpha}$$

$$\Rightarrow d = \frac{-\ln(1 - \alpha y)}{\alpha}$$

$$y \in \left[0, \frac{1}{\alpha}\right[$$

If a random penetration depth is drawn for every photon it will result in beam intensity according to the expected law at any depth in the material. This approach was chosen rather than have a discretized space where the photons have a probability to be absorbed at each step of their travel through the material.



Comparison between discrete and continuous approach of space.

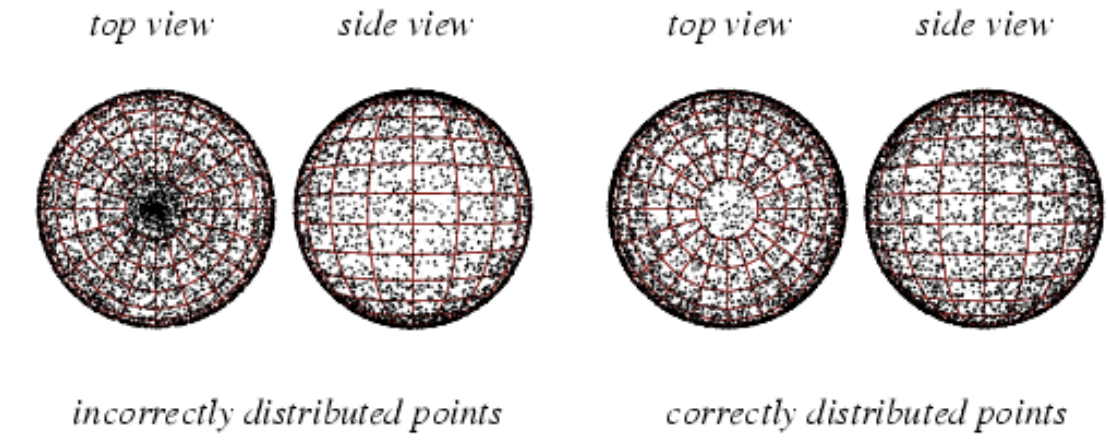
1.3.4 Reemission

The reemission is the most specific phenomenon to luminescent collectors and hence to PhotonSim . It depends on the internal quantum yield (QY), which is the number of reemitted photon divided by the number of absorbed photon for a single quantum dot. The quantum yield value depends on the wavelength of the absorbed photon. The probability for an absorbed photon to be reemitted is equal to this QY. For each photon, a random number is draw and if it is greater than the QY, the photon is lost, trapped inside the material else it is reemitted. When a photon is reemitted, its direction of reemission follows a uniform spherical distribution.

Care has to be taken when drawing the two spherical angle [A] for the reemission. *Theta* has to be draw between 0 and 2π but if *phi* is drawn randomly between $-\pi/2$ and $\pi/2$ this will result in a non uniform distribution. This happens because for equal variation of *theta* and *phi,* the surface on a sphere is greater when *phi* is close to the equator then when it is close to the poles. To avoid this, the elevation angle *phi* should be chosen by using the inverse cosine of a uniformly distributed value between -1 and 1:

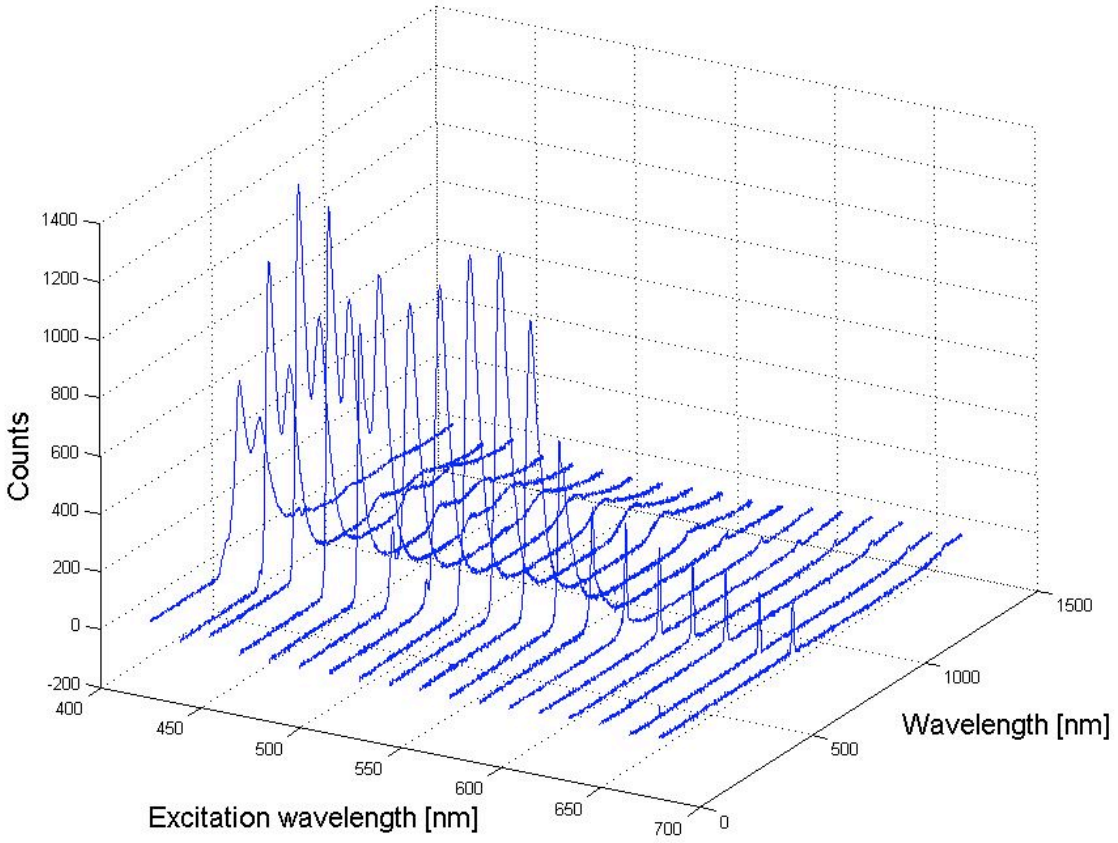
$$\theta = 2\,\pi\,u$$
$$\phi = \cos^{-1}(2\,v - 1)$$

$$u\,,\,v \in U\,[0,1]$$



Spherical distribution using correct and incorrect functions to generate angles[9].

The spectrum of reemission depends on the wavelength and is a crucial information about the material. If the reemission overlaps with the absorption, multiple absorption will occur and therefore there is a higher chance to be definitely absorbed by the material. The following measures give an example of such reemission spectrums:



The reemission spectrum at various incoming wavelengths as obtained from a plexiglas sample doped with a fluorescent dye by Miguel Valle Del Olmo [11].

1.4. Performance estimate.

It is important to have a significant output that will tell the experimentalist as much as possible about the sample and its capacity to concentrate light.

1.4.1. External Quantum yield

This value gives us the rate of photons concentrated at the edges over the total incoming. The complementary part has either been reflected at the interface before entering the concentrator, reemitted inside the concentrator but was then refracted out or absorbed and transformed into heat inside the concentrator.

1.4.2. Energetic Assessment

This defines the portion of energy concentrated to the borders by the system. It depends on the specified inputed energy flux, the incoming spectrum and the outgoing

spectrum. The first thing to do is define the duration of the simulation in real physical time:

$$\Phi_{\text{in}} = \frac{\sum E_{\text{inc photons}}}{t}$$

$$\Leftrightarrow \Phi_{\text{in}} = \frac{\sum_{i=0}^{N_{\text{in}}} \frac{hc}{\lambda_i}}{t}$$

$$\Leftrightarrow t = \frac{hc}{\Phi_{\text{in}}} \cdot \sum_{i=0}^{N_{\text{in}}} \frac{1}{\lambda_i}$$

N in is the total number of incoming photons and *lambda i* the wavelength of each of these photons. The sum of the inverse of wavelengths can be expressed as a function of the mean wavelength given by the incoming spectrum:

$$\bar{\lambda}_{\text{in}} = \sum_{\lambda=0}^{\lambda=\lambda_{max}} P_\lambda \cdot \lambda$$

$$\Rightarrow t = \frac{hcN_{\text{in}}}{\Phi_{\text{in}} \bar{\lambda}_{\text{in}}}$$

Now the real-time duration of the simulation is known, the outgoing energetic flux can be computed:

$$\Phi_{out} = \frac{\sum E_{out\ photons}}{t}$$

$$\Leftrightarrow \Phi_{out} = \frac{\Phi_{\text{in}} \cdot \bar{\lambda}_{\text{in}}}{hcN_{\text{in}}} \cdot \sum_{i=0}^{N_{out}} \frac{hc}{\lambda_i}$$

$$\Leftrightarrow \Phi_{out} = \frac{\Phi_{\text{in}} \cdot \bar{\lambda}_{\text{in}}}{N_{\text{in}}} \cdot \sum_{\lambda=0}^{\lambda_{max}} \frac{N_\lambda}{\lambda}$$

Or the efficiency may be used to characterize the collector, it is equal to the outgoing flux divided by the incoming flux:

$$\eta = \frac{\Phi_{\text{in}}}{\Phi_{out}}$$

$$\eta = \frac{N_{\text{in}} \cdot \bar{\lambda}_{\text{out}}}{N_{out} \cdot \bar{\lambda}_{\text{in}}}$$

SECOND PART

PhotonSim, the software

This part explains the choices made in architecture, offers some insight to the code and serves as a user guide explaining the use of PhotonSim. The reference manual can be found in a separate annex, it includes complete information about the program's structure and the source code.

2.1. Choice of architecture.

2.1.1. Platform, Philosophy

PhotonSim is an open source software, aimed at being cross platform, it is written in C++ because of it's power and flexibility and the numerous library existing for it. To obtain portability, the openGL and Glut libraries are used for 3D rendering. OpenGL is a powerful, efficient open source 3D library widely used and supported by most operating system. OpenGL is also supported by most graphical cards, relieving the CPU from heavy graphical computations. Glut is the openGL utility toolkit library and is used for interaction with the user and the useful feature it offers such as timers, multiple windows and predefined objects (cube, sphere etc....).

2.1.2. Graphical User Interface

PhotonSim does not offer a real graphical user interface (GUI) yet but allows multiple interaction using the Glut library. Basic commands such as starting, reset simulations or loading new configurations can be done through keyboard, mouse and a basic console implemented in openGL. The configuration of materials and light has to be done through text configuration files. Those interfaces will be explained in more detail in the next section.

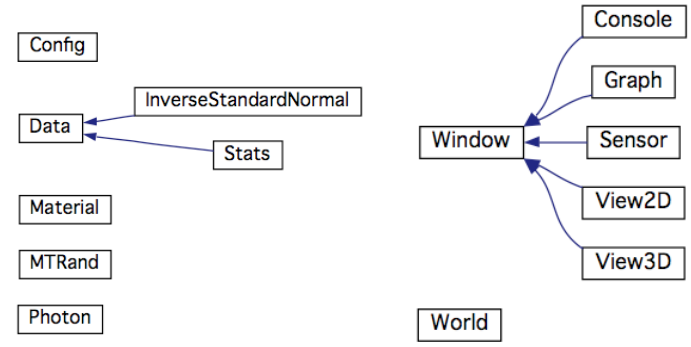
2.1.3 Object Oriented

The software has been implemented object oriented in order to be reusable, easily overtaken and extended. An object oriented (OO) software can be seen as a collection of little machine that collaborate and communicate to achieve a common goal. The objects may exchange information, act on or access shared data. The OO approach simplifies the translation of real world problems into software and is of great use to model physical behaviors. This object approach will be applied in PhotonSim. Some details of implementation will now be presented.

2.2 Implementation

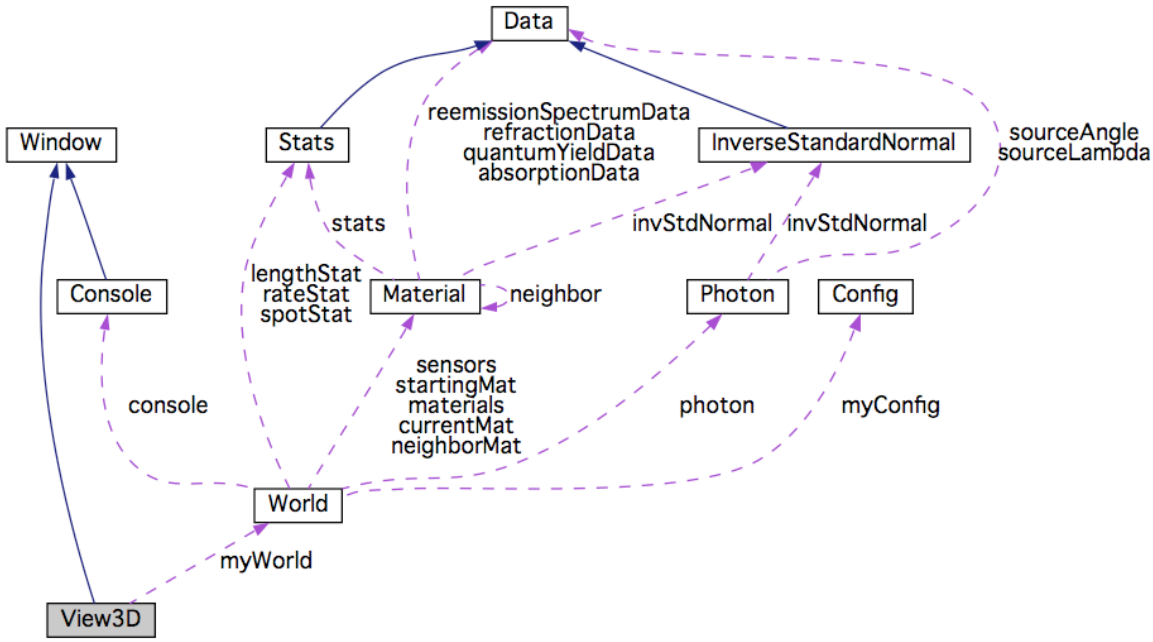
This part is a rapid presentation of the code, for more details please refer to the separate annex containing the complete description of the code and the code itself.

2.2.1 Objects



Class diagram.

Shows all the used classes in PhotonSim, MTRand is a random number generator, The Window class is for sub-windows in witch the different things will be displayed, the World class is the central class as shown by the next figure. Other Classes have self-explanatory names.



Collaboration diagram graph.

A purple dashed arrow is used if a class is contained or used by another class, the arrow is labeled with the variable through which the pointed class is accessible. A dark blue arrow is used to visualize a public inheritance relation between two classes.

2.2.2 The rendering

The rendering and interaction are managed by the "photonSim.cpp" and "photonSim.h" files who define when trajectories, graphs, sensors or the console need to be updated. There are essentially two modes: one timed where a step (calculus and rendering) is performed each given interval of time and a fast mode where the next step is computed as soon as the previous one was displayed. This part is independent from the calculus to make it possible to run only the computation without any display and increase the performance. This could be pushed further by creating a separate application with no rendering at all. It can be useful once a simulation is well defined to run it over a longer time and get more accurate results and little feedback.

2.2.3 The random Number Generator

The Mersenne twister is used to generate high quality pseudorandom numbers rapidly . This random number generator developed in 1997 by Makoto Matsumoto and Takuji Nishimura [5] was named after Mersenne prime numbers discovered by Marin Mersenne, a Philosopher, musician and mathematician of the 17th century who studied prime numbers amongst many other field of mathematics along with music theory.

2.2.4 The Next function

This function is central to the code, it is a state machine that manages the sequence of execution. It makes the calls to provided function that define the next state depending on the current photon's proprieties and the material it is in. It also manages the statistics, updating them and saving them at the end of a simulation.

```
void World::next(int timed)
// THIS IS THE MAIN STATE MACHINE
{
     stateType nextState = photon.state;
     float dist;
     float newLambda = 0;
     int draw = 0;

     if(current <= myConfig->NB_PHOTONS && !pauseBool){
          while (!draw){
               switch (photon.state){
                    case BIRTH:
                         photon.set();
                         draw = true;
                         side = BOTTOM;
                         currentMat = startingMat;
                         neighborType = currentMat->neighborType[side];
                        current++;
                         nextState = INTERSECTION;
                         break;
```

```
                    case MOVE:
                         photon.updatePos();
                         dist = currentMat->getDistance(photon.lambda);
                         if(currentMat->type != AIR)
                              lengthStat.countFloat(dist);
                         side = currentMat->intersection(&photon.pos[0], &photon.v[0],dist,
                                                         &photon.dest[0]);
                         if (side==NOSIDE)
                              nextState = ABSORPTION;
                         else{
                              nextState = INTERSECTION;
                              neighborType = currentMat->neighborType[side];
                             }
                         draw = true;
                         break;

                    case INTERSECTION:
                         switch(neighborType){
                              case SENSOR:
                                   if(myConfig->refractOut){
                                        if(currentMat->reflection(side, &photon,
                                           neighborMat->getRefractionIndex(photon.lambda)))
                                        nextState = REFLECTION;
                                        else{
                    photon.refract(currentMat->getRefractionIndex(photon.lambda),
                                        neighborMat->getRefractionIndex(photon.lambda), side);
                    nextState = DETECTION;
                                         }
                                   }
                                   else
                                        nextState = DETECTION;
                                   break;

                              case MIRROR:
                                   nextState = REFLECTION;
                                   break;

                              case NONE :
                                   nextState = LOSS;
                                   break;

                              default:
                                   neighborMat = currentMat->neighbor[side];
                                   if(currentMat->reflection(side, &photon,
                                        neighborMat->getRefractionIndex(photon.lambda)))
                                        nextState = REFLECTION;
                                   else
                                        nextState = REFRACTION;
                                   break;
                             }
                         break;

                    case REFLECTION:
                         photon.reflect(side);
                         nextState = MOVE;
                         break;
```

```
                    case REFRACTION:
                         photon.refract(currentMat->getRefractionIndex(photon.lambda),
                                        neighborMat->getRefractionIndex(photon.lambda),side);
                         currentMat = neighborMat;
                         nextState = MOVE;
                         break;

                    case ABSORPTION:
                         newLambda = currentMat->reemission(photon.lambda);
                         if (newLambda != 0){
                              photon.reemit(newLambda);
                              nextState = MOVE;
                             }
                         else{
                              photon.life = ABSORBED;
                              photon.updatePos();
                              rateStat.add(&photon);
                              nextState = BIRTH;
                             }
                         break;

                    case DETECTION:
                         photon.life = DETECTED;
                         photon.updatePos();
                         detected++;
                         rateStat.add(&photon);
                         if ( side == spotStat.side)
                              spotStat.addSpot(&photon);
                         float xy[2];
                         photon.getIncomingAngle(&xy[0], side%3);
                         currentMat->neighbor[side]->stats[GENERATION].add(&photon);
                         currentMat->neighbor[side]->stats[SPECTRUM].add(&photon);
                         currentMat->neighbor[side]->stats[POLAR].addElevation(&photon);
                    currentMat->neighbor[side]->stats[TREGENZA].countTregenza(xy[0],xy[1]);
                         currentMat->neighbor[side]->stats[ANGLE].addAngle(xy[0],xy[1]);
currentMat->neighbor[side]->stats[ANGULAR_SPECTRUM].addAngularSpectrum(xy[0],photon.lambda);
                         nextState = BIRTH;
                         draw = true;
                         break;

                    case LOSS:
                         photon.life = LOST;
                         lost ++;
                         photon.updatePos();
                         rateStat.add(&photon);
                         nextState = BIRTH;
                         break;
                  }
               photon.state = nextState;
            }
     }

     else if (!pauseBool){
          printf("*********DONE********\n");
          if (myConfig->textOutput){
               save();
            }
          done = true;
      }
}
```

2.3 Configuring a simulation

Upon startup, the software loads the "default" configuration folder located in the root directory. Root directory is the folder containing the executable file and all input and output directories. To run a custom simulation, a configuration has to be defined. This is done either by modifying the default configuration or by creating a new folder named after the simulation and located directly in the root directory. Each configuration folder must contain one "config.txt" file and one "light.txt" file. The format of all configurations and input files is explained in a template file located in the "template" folder. Each configuration file contain several commands as detailed hereafter for each type of file. The commands have to be given in capital letters. The order of configuration commands has no importance but once a command is given, **all** the arguments need to be specified in the right order or it will result in a false configuration. If an argument is missing this will also result in a confusing configuration as the argument will be given a default or zero value. Any line starting with "//" is considered as a comment line and is ignored.

2.3.1 General Configuration

The "config.txt" file is required to run a simulation, it has to be structured as explained in the config_template.txt file located in the template directory. Possible commands are (default value in parenthesis) :

- NAME (default) give the simulation a desired name, cannot contain spaces.
- NB_PHOTONS (10^6) the number of "shot" photons in a complete simulation.
- NB_DISPLAYED (500) the number of photon trajectories that will be displayed. If set to 0 no trajectories will be displayed.
- STATS_FREQ (5000) the period (in photons) at which the statistics will be displayed in the console. Too low periods will result in slowing down the simulation and makes the output unreadable.
- GRAPH_FREQ (2000) the period (in photons) at which the various graphs will be refreshed. To low periods result in flickering graphs.
- TEXT_OUTPUT (false) boolean value (true or false) defining wether or not the results of a simulation is printed to a file. Default is false because the user can use the keyboard to save results of a simulation.
- TIMED (true) a boolean value (true or false) defining wether or not the simulation is running in timed mode (step by step). If not the simulation runs at maximum speed, this makes the displayed trajectories unreadable quickly if too many are displayed.

- INTERVAL (1000) the interval in *ms* between two steps of execution in timed mode. A step stands for the photon's trajectory between two events (two interfaces for example).
- NB_MATERIALS (1) defines the number of material the collector is composed of (maximum 20). This command has to be followed by a corresponding number of filenames containing the configuration of each material. Materials can be separated by an air interval. Use the DIST command and specify the distance between two layers in *nm,* if their is no interval, specify 0. It is important that the specified filename contains no space and matches a file inside the current configuration folder (i.e. the path name is relative to the current configuration folder)
- BOTTOM_MIRROR and BOTTOM_SENSOR (false) are boolean values (true or false) defining the bottom of the collector. The last value overrides the other: they cannot be a mirror and a sensor at the bottom of a stack of concentrators. If both are false the bottom of the collector will be AIR.
- REFRACT_OUT (false) a boolean value (true or false) defining wether or not the sensors "are" the border of the material, in this case, all photons hitting the border of a material with a sensor will be counted. If not, there is an air interval between the materials and the sensors. The photons may then be refracted out or reflected at the interface. This interval has no thickness, it is only to model the change of angle or the internal reflection at the interface.
- ENERGY_FLUX(1000) the incoming energy flux.

An example of main configuration file:

```
NAME example
NB_PHOTONS          500000
NB_DISPLAYED        250
ENERGY_FLUX         800

STATS_FREQ          5000
GRAPH_FREQ          1003
TEXT_OUTPUT         true
TIMED               true
INTERVAL            1000

NB_MATERIALS        2
one.txt
DIST 5
two.txt

BOTTOM_MIRROR       true
REFRACT_OUT         true
```

2.3.2 Light Configuration

The light configuration works as explained in the light_template.txt file located in the template directory. It contains the definition of the light source. Possible commands are (default value in parenthesis) :

- LIGTH_DIST (1000) this parameter defines the length of the rays hitting the sample. It has no influence on the results of the simulation, it is only for visibility.
- CENTERED (false) a boolean value (true or false) defining wether or not all rays hit the sample at its center. It is useful to have a cleaner view of what happens in the sample and was used for the validation explained in part three where the measures where using a centered excitation beam.
- COLORED (false) a boolean value (true or false) defining wether or not the rays are displayed in color. This is another visual option making the rays appear in a color corresponding to their wavelength [7].
- SKY (CLEAR) this is the central command in the light configuration parameters. It may be set to CUSTOM, CLOUDED, CLEAR_DIRECT or CLEAR_DIFFUSE The three later were described earlier. In the case of CUSTUM, the following commands need to be specified:
    - ANGLE this command is followed by a description of the incoming light angular distribution, it may be SINGLE values, RANDOM, from a UNIFORM_DIST or a CONE. When defining an elevation angle, this is not the polar angle (angle from the normal) but the elevation angle from the surface.
    - LAMBDA this command is followed by a description of the incoming light's spectrum, it may be a SINGLE wavelength, RANDOM, follow a NORMAL distribution or from a FILE.
        - SINGLE has to be followed by the two polar angles for ANGLE description (elevation first followed by azimuth) or by a single wavelength for LAMBDA.
        - RANDOM defines two bounds in which the values will be uniformly distributed. For angle four values need to be specified the two first values are lower and upper bounds for the elevation the next two values for azimuth.
        - UNIFORM_DIST is a uniform spherical distribution.
        - FILE define the path to a DATA file containing a definition of the distribution. It is important the pathname contains no spaces and is relative to the root directory.

- CONE may be used only for angular distribution and yields rays contained in the cone defined by three values: elevation, azimuth and an angular width.
- NORMAL may be used to define normally distributed spectrum. The normal distribution is defined by two following values, the first for the mean the second for the variance.

An example of light configuration file:

```
LIGHT_DIST 1000

CENTERED   true

COLORED    false

SKY CUSTOM

ANGLE      CONE       45    0     30

LAMBDA     RANDOM     200   800
```

2.3.3 Material Configuration

The material configuration works as explained in the material_template.txt file located in the template directory. It contains the definition of the material. Possible commands are (default value in parenthesis if they exist) :

- NAME (mat) specify the name the material.
- XSIZE the size of the material along the X axis in *nm*.
- YSIZE the size of the material along the Y axis in *nm*.
- ZSIZE the size of the material along the Z axis in *nm*.
- REFRACTION defines the index of refraction *n* in the material, it may be CST or from a FILE.
- ABSORPTION defines the absorption coefficient *k* or *alpha* in the material, it may be CST or from a FILE. The command has to be followed immediatly by 'K' or 'ALPHA' to specify what format the absorption coefficient is in.
- REEMISSION_QY defines the probability of reemission (QY) in the material, it may be CST or from a FILE.

- REEMISSION_SPECTRUM defines the spectrum of reemitted light it may be CST, from a FILE or following a NORMAL distribution.
- CST this command may follow the REFRACTION, ABSORPTION, REEMISSION_QY or REEMISSION_SPECTRUM commands, it assigns to the considered value (*n, k,* QY or spectrum) the constant values that follows it.
- FILE this command may follow the REFRACTION, ABSORPTION, REEMISSION_QY or REEMISSION_SPECTRUM commands, it assigns to the considered value (*n, k,* QY or spectrum) a DATA file from which its values will be taken, depending on the wavelength. It is important the pathname contains no spaces and is relative to the root directory.
- NORMAL may be used to specify the reemission as a normally distributed spectrum. The normal distribution is defined by two following values, the first for the mean the second for the variance.
- NEAR, FAR, LEFT and RIGHT command may be used to define the neighbors others then top and bottom which are defined by the config.txt file. they can be followed by MIRROR, SENSOR or AIR commands.

An example of material configuration:

```
NAME       oneMat

XSIZE      5000
YSIZE      5000
ZSIZE      200

REFRACTION           CST        1.2
ABSORPTION           K    FILE       data/stair_550_0.9_0.02_abs.txt
REEMISSION_QY        CST        0.95
REEMISSION_SPECTRUM  NORMAL     400   80

LEFT       SENSOR
RIGHT      SENSOR
NEAR       MIRROR
FAR        AIR
```

2.3.4 Statistics Configuration

A configuration file defines the characteristics such as limits and precision of each statistic. The "stats_template.txt" file explains this configuration. Possible commands are (default value in parenthesis) :

- SPECTRUM (200 2 1000) followed by the first wavelength, the interval between two wavelengths and the size of the statistic (last value = first + (size-1) * interval).
- ANGLE (30) defines the accuracy over 90° for the angles statistic.
- POLAR (30) defines the accuracy (subdivisions) over 90° for the polar angles statistic.
- GENERATION (20) the maximum number of generation counted (greater values will be counted as this value).
- SPOT (no default) is followed by numerous values:
  - first wavelength, wavelength interval, size
  - the position at which the statistic is recorded (x, y, z, side)
  - the spot radius
  - the cone for which values will be recorded (elevation, azimuth, opening)
  - the distance between the material and the "spot sensor".

TREGENZA is a defined statistic it cannot be configured.

An example of statistics configuration:

2.3.5 Data Files

One more important configuration file type remains, the data file, it can be referred to by the other configuration files to define one or two dimensional arrays of values. It may stand for a spectrum, represent a table, a multidimensional table or an array of spectrums. It can also be used for statistics. Its definition is the most delicate and care has to be taken when defining it. Possible commands are (some commands marked by a * have to be present):

- NAME (data) the name of the data
- NB_VALUES(*) the total number of values
- START (0) the index corresponding to the first value (for example if the data is a spectrum this will be the wavelength corresponding to the first value).
- INTERVAL (1) the interval between two indices.

The next thing that has to be defined is the type of the data, by default it is a table were the START values stand for the first index and the INTERVAL value defines the precision of the table. The values given (y) have to correspond to regularly spaced values (x). Other possible types for the definition of two dimensional arrays are :

- MULTI_REGULAR is followed by the number of columns, the value for the first column and the interval between two columns. This can be used to define several reemission spectrums corresponding to regularly spaced excitation spectrums.
- MULTI_INDEXED followed by the number of columns and the corresponding number of values for the columns indic. This may be used in case of non regularly spaced reemission spectrums.

Some actions may be performed on a set of values: they may be normalized, integrated and acceded inversely: instead of looking for a value after its indic, look for the index corresponding to a value. If the data has to be used to generate random numbers you will have to normalize, integrate and inverse it, if it is already a cumulative distribution function you only need to inverse it. It is always best to specify an already normalized, integrated and inverted set of values. If needed the action of normalization, integration and inversion can be performed with the commands NORMALIZE, INTEGRATE and INVERSE.

Finally, the VALUES(*) command has to precede the set of values. An example of data file based on the Matlab colormap that is used by the program to associate a color to a ratio (between 0 and 1):

```
NB_VALUES 192              // 3*64
NAME      color64
START     0
INTERVAL  0.015625
MULTI_REGULAR   3    1    1

VALUES
0    0    0.5625
0    0    0.625
0    0    0.6875

..   ..   .. (not complete)
..   ..   ..

0.62 0    0
0.56 0    0
0.5  0    0
```

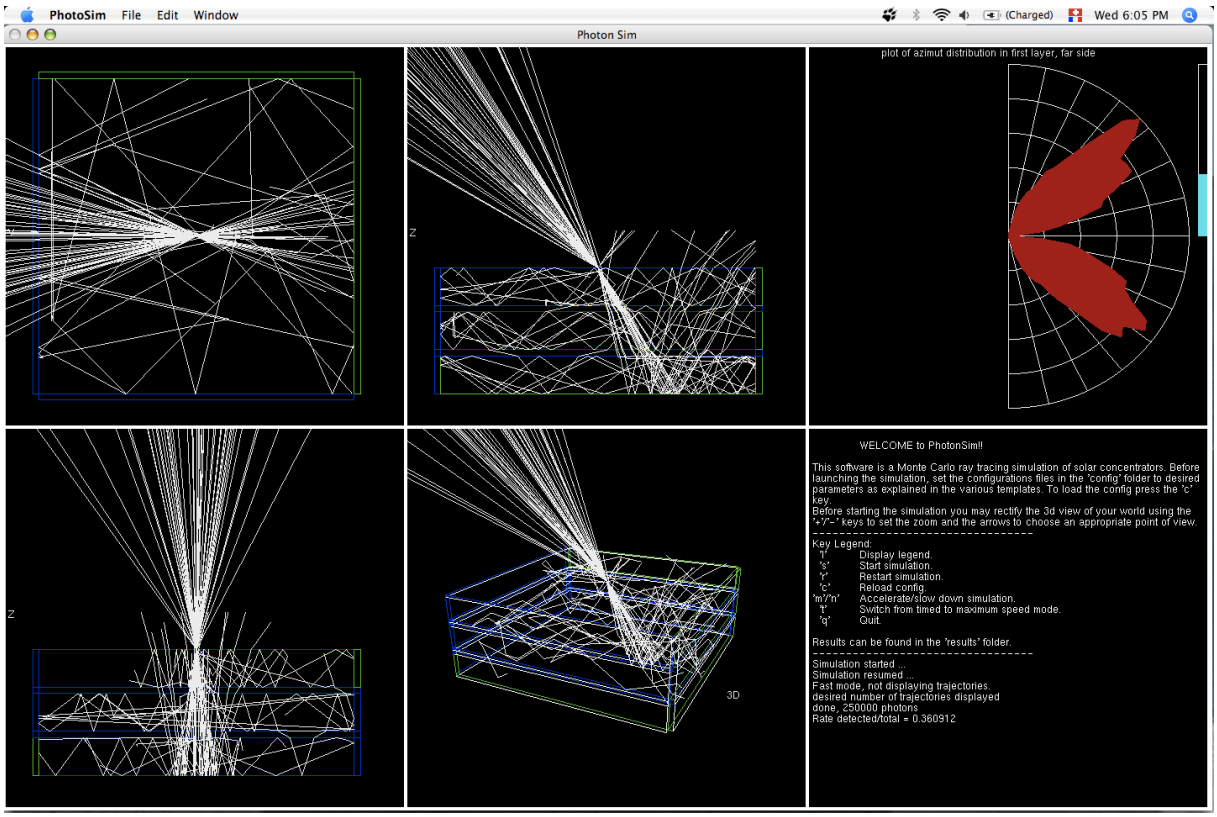
2.3.6 CONSISTENCY

There is not much support to check the validity of the inputs. Some nonsenses might not be reveled. Wrong path names will be signaled in the console when loading a new configuration.

Particular care has to be taken for the spot statistic when defining the position and the side, the values have to match a material's side coordinates. This is also true for any data used to define materials proprieties, they have to be prepared carefully in order to get the desired characteristics.

2.4 Running and interacting with PhotonSim

2.4.1 The PhotonSim Windows



A screen shot from PhotonSim.

The windows represent from the left to the right top to bottom, a two dimensional view of trajectories (x and y, view from top), an other two dimensional view of trajectories (x and z, view from the side), a graph in this case plotting the distribution of polar angles at the output in the first layer on the right. Then two dimensional window show the trajectories from the front (y and z), a three dimensional view and finally the console which displays some basic informations about what is going on and can be used by the user to interact with the software. Materials are displayed in red, sensors in green, mirrors in blue and air is not displayed. The red color of materials may be hide by the green of sensors or the blue of mirrors.

2.4.2 Loading a Configuration

As stated before the default configuration is loaded upon startup, to load a custom configuration, the console has to be selected with the mouse and the name of the configuration folder containing the complete definition of the collector to simulate typed in. Hitting the return key of the keyboard will load this configuration. An message will indi-

cate success or failure of the operation. If a mistake is made while typing the user may always reset the complete simulation name with the backspace key. If the user wishes to modify the configuration, this can be done in an editor while PhotonSim is running and once saved the configuration may be reloaded by typing the "c" key.

The console is mainly designed to output basic information and input new configurations. Therefore when performing any other action using the keyboard an other window has to be selected or the mouse's right click used.

2.4.3 Managing the simulation

Once a configuration is loaded with success, the simulation may be started, restarted, paused, reconfigured, slowed down, fastened up and more. The following table states possible action acting on the simulation:

Key	Function
s	Starts the simulation using the values in the configuration file
r	Restarts the simulation
p	Pauses the simulation
c	Reloads the configuration
t	un times the configuration, it will run at maximum speed
f/v	If the simulation is not at full speed 'f' will fasten it up, 'v' slow it down.
d	Sets the display of trajectories on or off
e	Erases all trajectories
w	Writes statistics and results to file
q	Quits the simulation

Some of these action can also be performed by using the right clic off the mouse and selecting the desired action.

2.4.4 Changing the 3D view.

For a better visualization, the 3D view can be rotated an zoomed in and out. This can be done using the keyboard:

Key	Function
+/-	Zoom in / out
←↑→↓	Move the camera around the world.

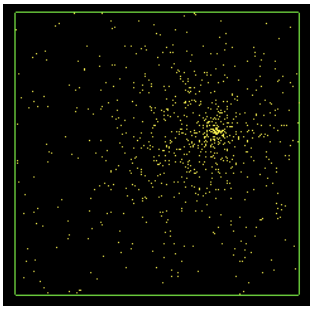
2.4.5 Displaying Statistics

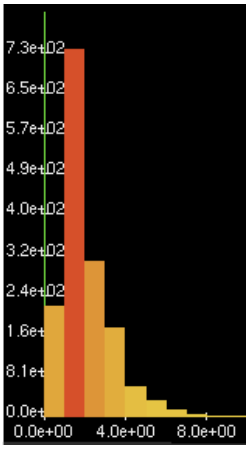
PhotonSim offers various statistics, they can be displayed in the upper right window of the display. It is possible to switch from one statistic to the other using either the keyboard or the right clic of the mouse:

Key	Function
1	Displays a sensor and the collisions of photons with it.
2	Displays the number of generations.
3	Displays the spectrum.
4	Displays the angular distribution.
5	Displays the polar angle distribution.
6	Displays the angular distribution using the Tregenza representation.
7	Displays several spectrum depending on the outgoing polar.
8	Displays the spectrum at a defined spot with a given angle.

All these statistics except "spot" (8 key) are given for a particular sensor, it is possible to switch between a statistics for on sensor to the next either by repeatedly hitting the corresponding key or by using the right click and selecting "next" or "previous". The colored bar on the right side of all plots indicates the rate of total captured photons over total "shot" photons.

1. This is not really a statistic but rather a visualization of collisions. The sensors borders are displayed in green and from the time one where it is selected, yellow dots are displayed at every collision. It can give an idea of the distributions of collisions on each side, in each layer of the collector.



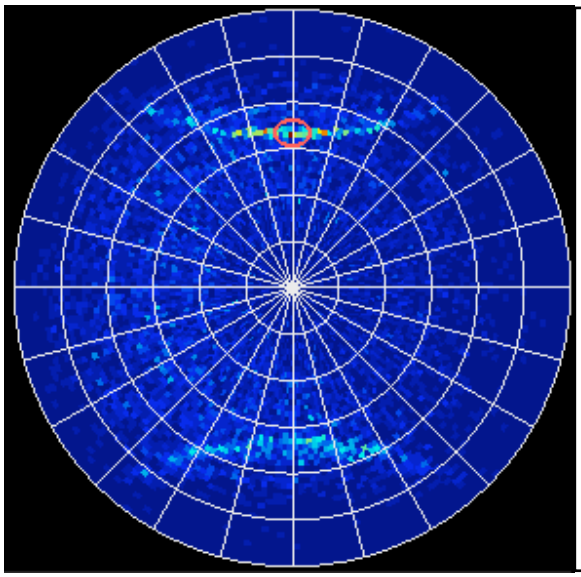


2. The number of generations stands for the number of absorptions followed by reemissions a photon went trough before being captured. It is displayed in a bar graph. The color of the bars give an extra representation of the proportion to the total number of counted photons. The numbers on the horizontal axis stand for the number of generation and those on the vertical axis for the count of photons captured with this number of generation. In the present example, 730 photons of generation 1 were recorded.

3. The spectrum simply represents the count of the wavelengths of photons captured. It is useful to know the amount of energy concentrated by the collectors and have a more accurate way to choose the appropriate solar cells if the application is in photovoltaics.

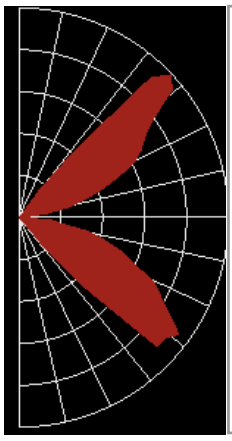
4. The angular distribution displays the distribution of polar and azimuth angles. The polar angle is displayed on the radius, starting at zero for the center of the plot and going until 90° at the circumference. The azimuth is represented by an equal displacement from the vertical (0° = North) . The highest concentration of rays will result in a dark red zone. The color then decreased to clear red, into yellow, light blue and finally dark blue which stands for no rays at all. This statistic can be used to generate bidirectional scattering distribution functions.

Example:



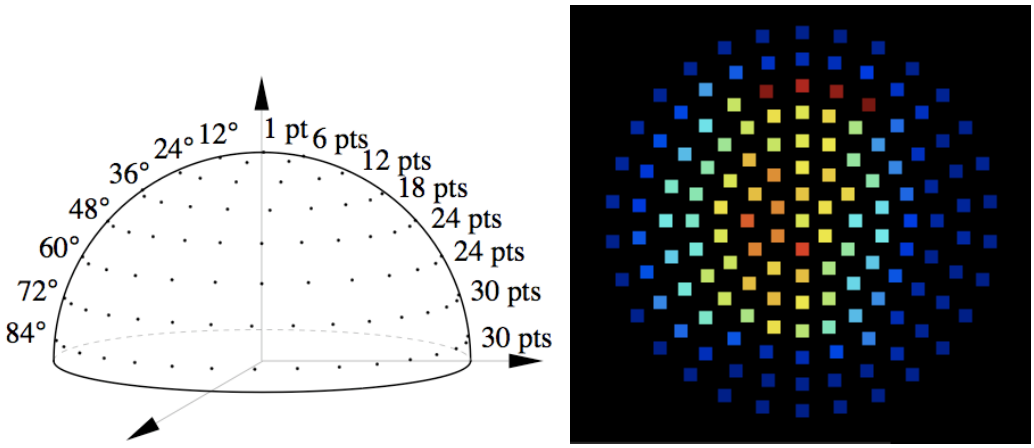
This plot shows two arcs with a strong concentration. The highest concentration among all is at about 50° from the zenith directed to the north (circled in red). More generally there is a higher concentrations of rays to the left side.
The axis lines stand for 15°.

5. The polar distribution represents the polar angle, taking into account if it is directed to the north or to the south. The further the line is from the center, the higher the concentration is. This could be handy if the collector is to be used in day-lighting. Example:



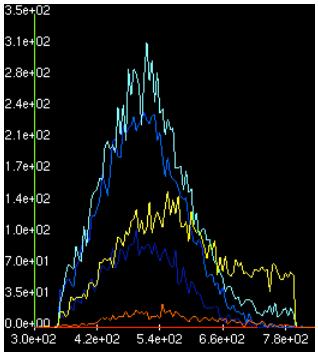
In this case, the the polar distribution is strongly symmetric and most polar angles are between 25° and 45°, some between 12° and 25° but none above 45° and very few at 0°.

6. The Tregenza representation is an other way to represent the repartition of light depending on polar and azimuth angle. It divides the sky into 145 defined sections and is widely used in day-lighting as it is a standard for measurements with sky scanners.

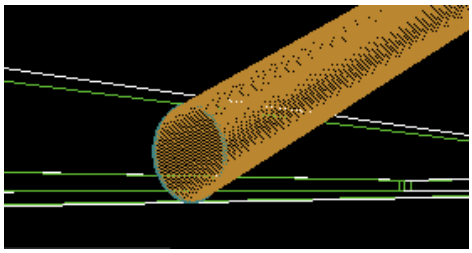


The distribution of the Tregenza sample points. A screenshot from photonSim for the same distribution as in 4.

7. The spectrum depending on the angle. Represents the spectrum but depending on the polar angle. Dark blue stands for close to normal angles and dark red for close to 90° angles. The interval between two lines is regular and can be set in the configuration. In the presented screen shot, they are five lines, the dark blue line stands for angles from 0° to 18°, marine blue for angle from 18° to 36°, clear blue for angles from 36° to 54°, yellow from 54° to 72° and finally red for angles from 72° to 90°. This statistic shows both the distribution of the polar angles and the wavelengths. It should also allow to verify the model by observing a red shift.



8. The spectrum at a specified spot. This statistic is mainly for a verification purpose. It allows to observe rays coming out of the collector in a precise zone and with a certain angle:



Screenshot showing the representation of the "spot sensor" on a side of the material, rays exiting the material and traveling in the brown cone only are counted.

2.5 Output Files

PhotonSim generates text files containing the statistics and an extra file containing a resume of the simulation and corresponding results.

2.5.1 Synthesis

The synthesis contains the ratio of reemitted energy to total energy and the count of detected, lost and absorbed photons. Detected photons are all the photons that hit a sensor, on any side in any layer. Lost photons are all the photons the exited the system, they may have been refracted out or reflected away when they first hit the concentrator. Absorbed photons are the photons that were absorbed and never reemitted.

An example of synthesis file:

```
******** Synthesis for the demo simulation number 0 *******
************************************************************

Incoming energy flux : 1000.000000
Real time equivalent: 0.000000 sec
Overall Outgoing energy flux: 0.000000 (for all photons cap-
tured by all sensors) these values might be to small as it
takes about 10^20 photons to simulate a second therefore the
following gives a ratio of outgoing flux over incoming flux :
0.142141

% Count of the detected/lost/Absorbed/total photons
%    Detected     Lost     Absorbed     Total
     33374        25705    8732         67811
```

2.5.2 STATISTICS

Every statistic is written to a file with a self-explanatory name: "[name of simulation] # [run number] [name of stat] L[layer number] S[side].txt". The counted photons are displayed in a single or multiple columns.

An example of statistic output for the angles distribution:

```
/// Count of the different outgoing angles of the photons through the sensor, azimuth in
/// rows, polar angle in columns

            0.0   9.0   18.0  27.0  36.0  45.0  54.0  63.0  72.0  81.0

0.0         14    48    66    94    90    191   142   54    6     0
9.0         12    43    75    82    85    147   134   48    7     3
18.0        13    37    51    63    66    77    166   57    10    0
27.0        12    44    55    70    76    45    86    71    28    4
36.0        13    38    61    49    41    59    30    22    38    8
45.0        7     32    55    46    46    29    28    21    6     1
54.0        10    26    45    54    46    36    26    14    9     4
63.0        12    32    61    49    36    31    18    11    6     1
72.0        19    32    31    33    44    23    19    6     4     0
81.0        13    44    40    36    31    25    24    8     4     2
90.0        17    28    51    56    34    34    16    8     5     0
99.0        11    32    36    31    43    33    14    14    4     8
108.0       12    33    53    38    43    27    14    9     1     0
117.0       16    37    66    45    35    44    24    11    3     2

…           …     …     …     …     …     …     …     …     …     …
…           …     …     …     …     …     …     …     …     …     …

225.0       16    37    68    82    94    86    77    55    40    5
234.0       14    38    65    76    95    83    77    61    26    3
243.0       18    36    66    75    90    91    64    51    26    4
252.0       18    43    59    94    78    78    73    51    22    6
261.0       14    46    66    86    79    77    89    58    16    6
270.0       13    34    53    81    79    82    51    48    21    5
279.0       13    46    66    86    92    74    68    47    29    2
288.0       15    47    73    72    93    90    74    53    22    4
297.0       10    33    64    86    90    66    92    48    22    3
306.0       14    51    62    91    94    106   77    60    26    6
315.0       11    42    67    87    82    94    76    67    59    6
324.0       14    49    59    84    100   99    140   143   39    2
333.0       16    43    63    76    104   134   196   81    14    2
342.0       13    35    55    75    84    153   175   47    7     0
351.0       11    38    62    89    97    186   124   61    8     2
///   TOTAL COUNTED                 #20161
```

Note the indexes stand for the lower bound angle, the counted values are for angle between this lower bound and the next index.

2.5.3 IES Files

It is also possible to write .IES files to be used in radiance or other softwares to integrate a simulated collector and it's behavior in a complex environment. The file define the distribution of intensity over angle as well as an approximation of the overall illumination. An example of such a file:

```
IESNA91
[TEST]polar distribution in third layer, right side
[MANUFAC]PhotonSim simulation
TILT=NONE
1
-1
0.012560
61 1 1 2
0 0 0
1.0 1.0 0.0
0 3.000000 6.000000 9.000000 12.000000 15.000000 18.000000 21.000000 24.000000
27.000000 30.000000 33.000000 36.000000 39.000000 42.000000 45.000000 48.000000
51.000000 54.000000 57.000000 60.000000 63.000000 66.000000 69.000000 72.000000
75.000000 78.000000 81.000000 84.000000 87.000000 90.000000 93.000000 96.000000
99.000000 102.000000 105.000000 108.000000 111.000000 114.000000 117.000000
120.000000 123.000000 126.000000 129.000000 132.000000 135.000000 138.000000
141.000000 144.000000 147.000000 150.000000 153.000000 156.000000 159.000000
162.000000 165.000000 168.000000 171.000000 174.000000 177.000000 180
0
2.000000 9.000000 33.000000 36.000000 69.000000 91.000000 99.000000 117.000000
141.000000 165.000000 177.000000 210.000000 196.000000 259.000000 258.000000
227.000000 244.000000 237.000000 204.000000 187.000000 165.000000 173.000000
166.000000 110.000000 112.000000 85.000000 90.000000 53.000000 44.000000
3.000000 12.000000 38.000000 61.000000 73.000000 77.000000 115.000000
135.000000 142.000000 166.000000 178.000000 217.000000 239.000000 218.000000
245.000000 259.000000 236.000000 224.000000 239.000000 190.000000 177.000000
154.000000 157.000000 126.000000 99.000000 75.000000 67.000000 47.000000
27.000000 4.000000 3.000000 0
```

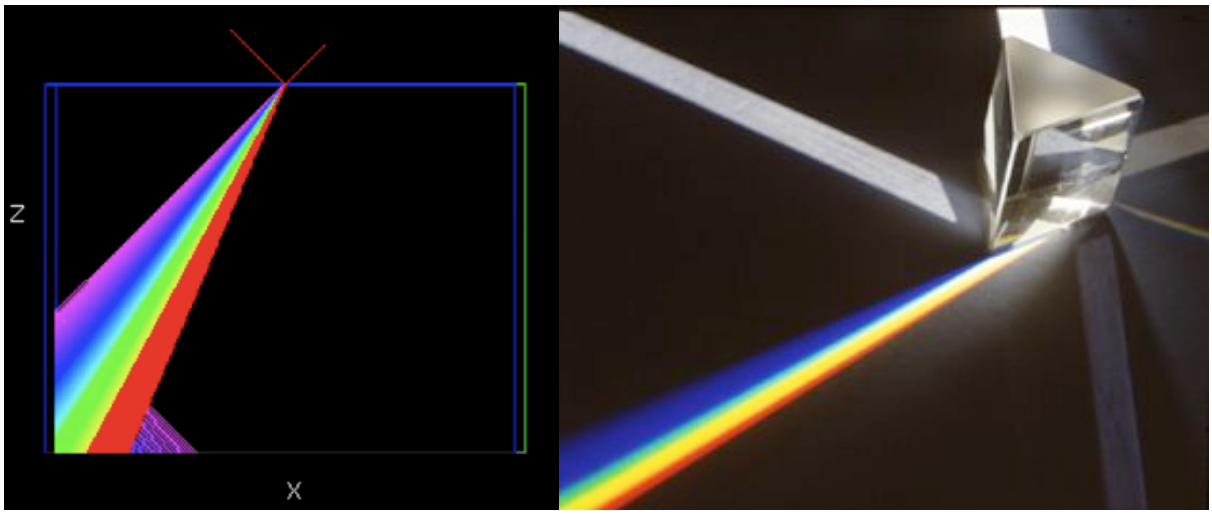
THIRD PART

Validation and some experiments.

The software will be tested step by step to verify it complies with expectations. Also, result of a simulation based on experimental values characterizing a single material will be compared to the measured performance values on a Rhodamine and Hesa-glass sample. These measures were made by Miguel Valle del Olmo at LESO.

3.1 Step by step verification

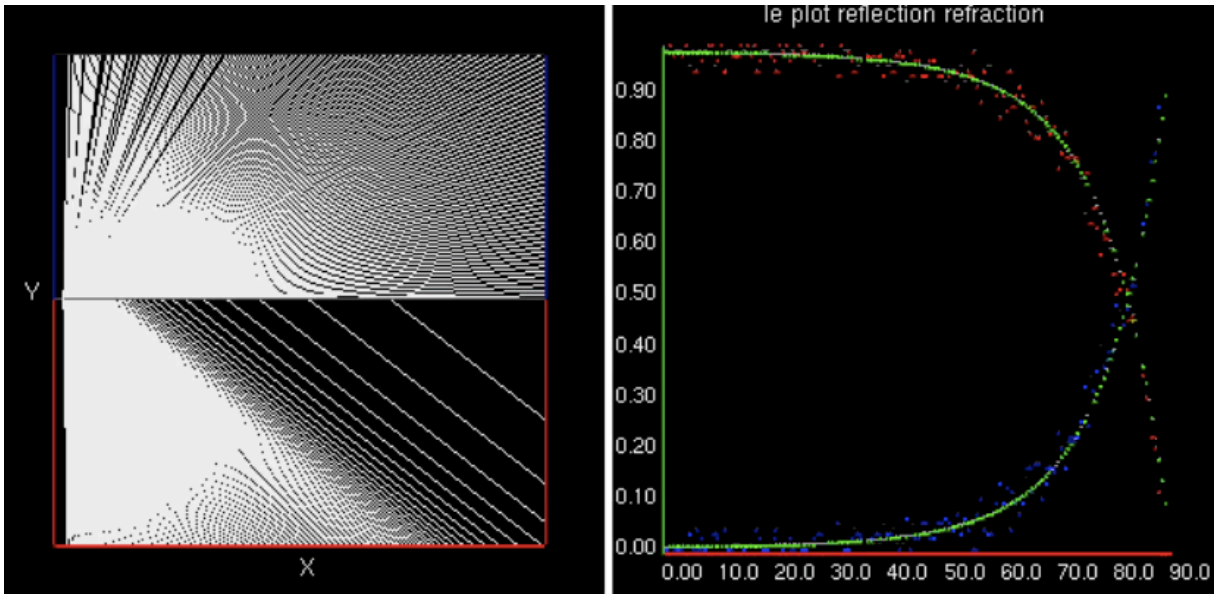
3.1.1 Refraction angle



The separation of light in PhotonSim and with a prism

This sample test was run shooting photons with the same trajectory and increasing wavelength. The photons are all shot in the same plane and the view is a 2D view in the direction of the normal to that plane. The phenomenon was exaggerated by incrementing the refraction index by 0.005 every 2 *nm*. It can be noticed the reflected rays have the same trajectory, no matter the wavelength. The lack of a clear yellow band in the simulation is due to the approximative function used the associate a color to a wavelength[7].

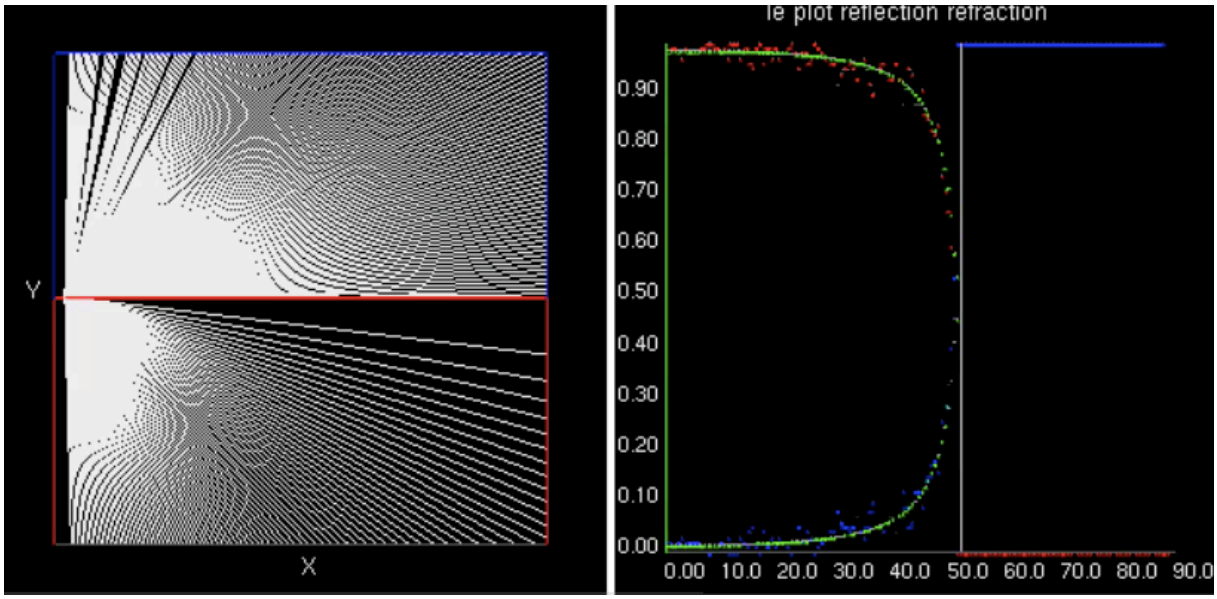
3.1.2 Reflection / Transmission



Transmitted and reflected light.

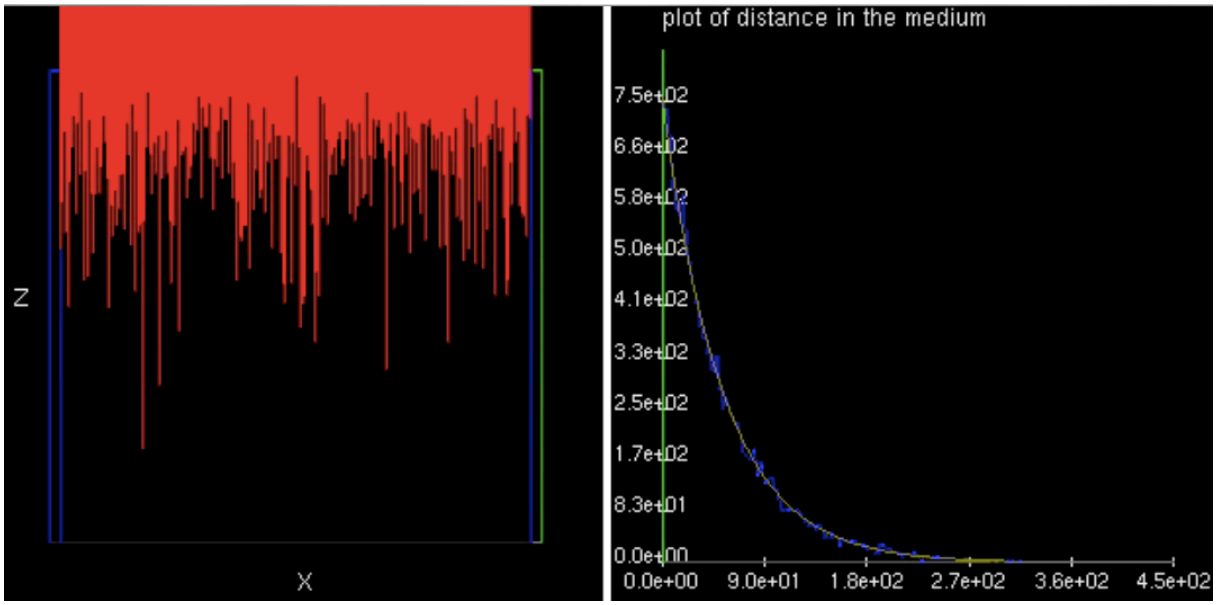
In this sample simulation, the incoming polar angle is decreased progressively until is is equal to the normal. The view is again along the normal to the plan the photons are traveling in. In this simulation, the indices of refraction were constant: 1.5 for the incident (top) material and 1.9 for the transmitting (bottom) material. The only changing parameter is the angle and 1000 rays were shot for each angle. The resulting portion of reflected and transmitted photons is plotted in the right-hand-side sub-window. The green line indicates theoretical values.

The same experiment was repeated with inverted indices of refraction:



Transmitted and reflected light, with expected critical angle at 53.13°

3.1.3 Distance in the medium



The penetration depth in the medium. "shot" photons left, simulated vs. theoretical distribution right.

In this test, the photons are shot vertically to the interface at increased positions (from left to right) to allow a good visualization of the various penetration depth in the material. A constant absorption coefficient was used with a QY equal to zero in order to observe only absorption. The plot shows the expected distribution of distances in the medium (in brown) and the simulated distances (in blue).

3.1.4 Reemission Direction

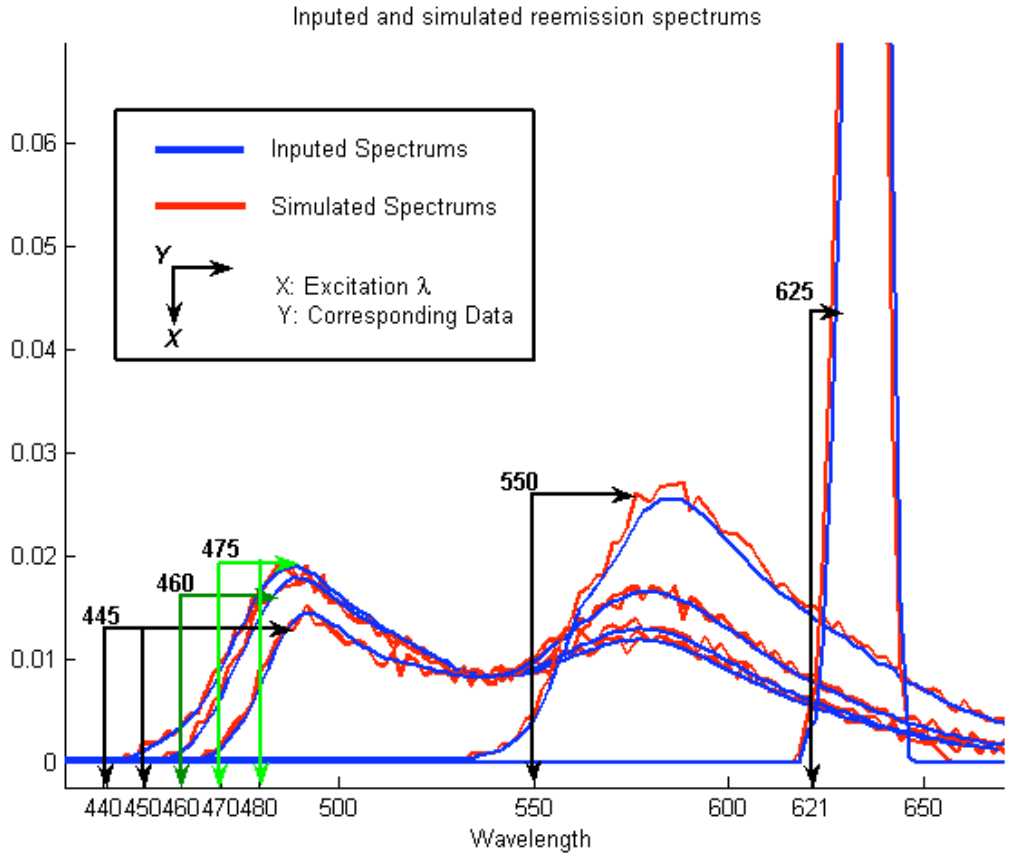
To check the reemission direction is equally distributed, all photons are absorbed in the middle of a square sample and the counts of photons at the far, near, left, right and bottom sides are compared:

Side:	Left	Right	Near	Far	Bottom	Total	Top
# Photons:	16894	16460	16712	16723	16532	100000	16679
% error	1.3%	1.3%	0.3%	0.3%	0.8%	-	0.1%

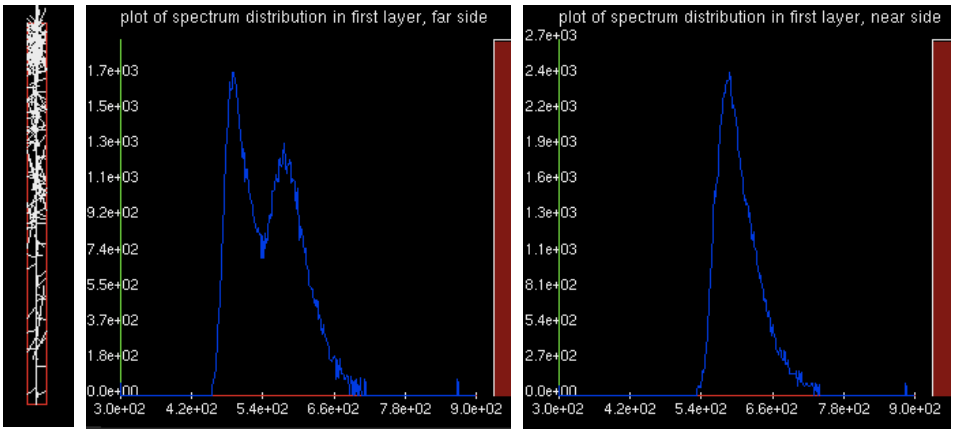
The error relatively to the expected count (16.666) on each side is small. PhotonSim can not run this type of simulation simply on a configuration file. To obtain this simulation, the source code was slightly modified.

3.1.5 Reemission Spectrum

To avoid multiple absorptions and observe the spectrum of the first reemission, the material is set to be long in the direction of the incoming photons. The photons hit the sample perpendicularly on a very small surface compared to the thickness of the material. Therefore most photons will be absorbed as the distance to travel through the whole material is big. But most reemitted photons will not be absorbed as the mean distance to the borders is small. The reemission spectrum used was measured on the orange Hexa-Glass sample. Seven simulations with excitation wavelength of 440, 450, 460, 470, 480, 550 and 621 *nm* confirmed the reemitted spectrum corresponds to the desired one. The spectrum chosen for the simulation of reemission is the closest measured one. Available spectrums for this sample start at 415 *nm* and go by 15 *nm* steps until 670 *nm* (415, 430, 445, ..., 655, 670). The following graph illustrates this verification:



Reemission verification for seven excitation wavelengths.

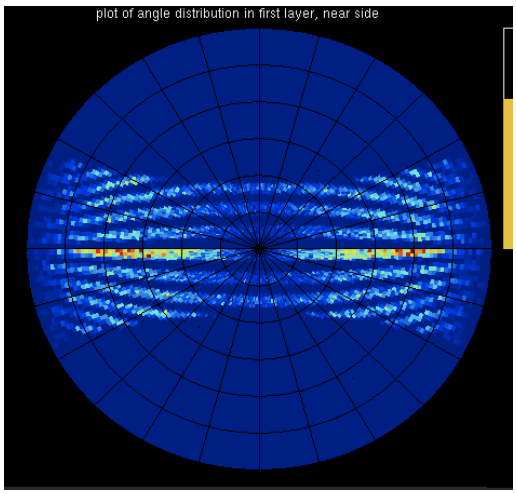


Some screenshots from this simulation: a side view and the reemission spectrums at 460 and 550 nm.

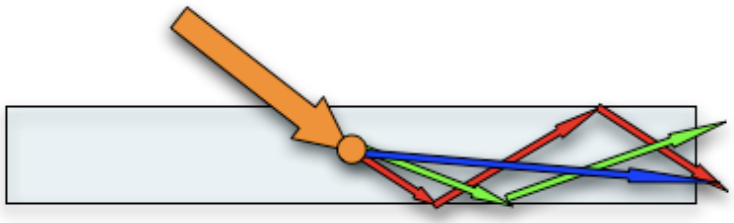
3.1.6 Explaining some Results

At a first glance, some simulation might seem wrong. First it can appear from one off the two dimensional views that there is an internal reflection at a very small angle however if this internal reflection is looked at from a different angle, it turns out to be normal as the angle to the normal is greater then the critical angle.

Also a configuration with a fixed incoming angle and centered in the material resulted in some fringes in the outgoing angles:



Those fringes are complementary, each corresponds to an interval of reemission angles. For certain reemission angles, the ray will reflect internally a number of times and will hit the sensor southwards.
When the reemission angle increases, the distance between two reflection increases until there is one reflection less and the rays hit the sensor northwards.



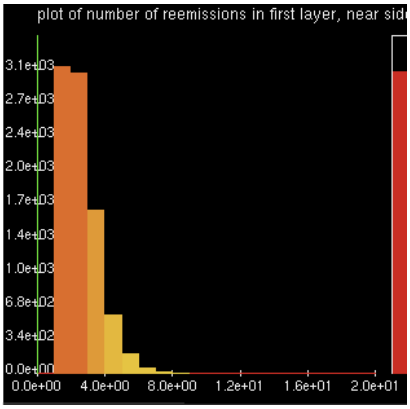
Different trajectories explaining the of northbound and southbound rays. Hence the fringes.

From a single point of reemission, rays with incidence greater then the critical angle will be internally reflected. The first rays (red in the illustration), with angles slightly greater then the critical angle will be reflected *n* times before exiting the concentrator. As the angle increases, the horizontal distance between two reflection increases until they are only n-1 reflections (green rays in the illustration). If the first ray were hitting the border of the concentrator southbound, the second will hit it northbound. After an other increase of the angle, they will be *n-2* reflection and the rays will hit the border southbound again (blue rays) etc... Because the angle at the output depends directly on the angle at reemission and varies in the opposite direction but proportionally, this explains that on the edges of a material stimulated at a single point the polar angles alternate between northbound and southbound.

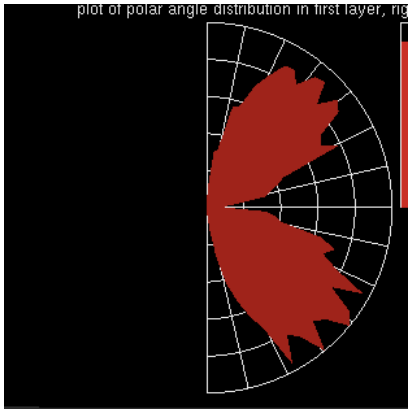
3.2 Simulating the orange Hexa-Glass Sample

The absorption coefficient of the samples have been characterized and are ready to be simulated. This offers a great opportunity to verify wether or not PhotonSim correctly models planar luminescent solar collectors. There was not enough time to complete a full comparison between measures and simulation. However here are some results for the sensor on the side of a simulated orange Hexa-Glass sample (incoming perpendicular beam with uniform wavelength between 300 and 900)

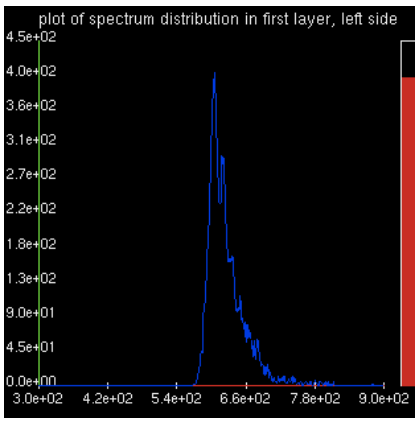
3.2.1 Concentration



The number of generations on a sensor. They are no photons of first generation, this is normal as the excitation beam hits the sample vertically and the photons have to be absorbed at least once to reach the sensor.



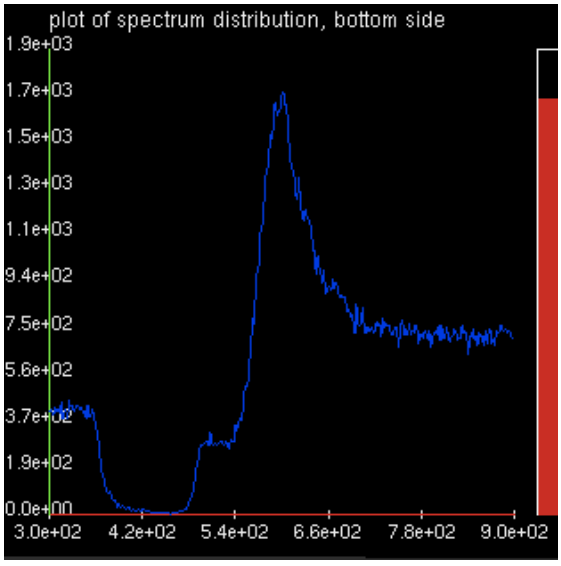
The polar angle is distributed in two lobes above and bellow the normal. This can be explained as the photons hitting the border of the sample have to be internally reflected and can therefore only hit the border within some angle. In reality the normal shows more intensity but this can be explained by the diffusion of the border in the real sample due to the small irregularities on it's borders which are not modeled here.

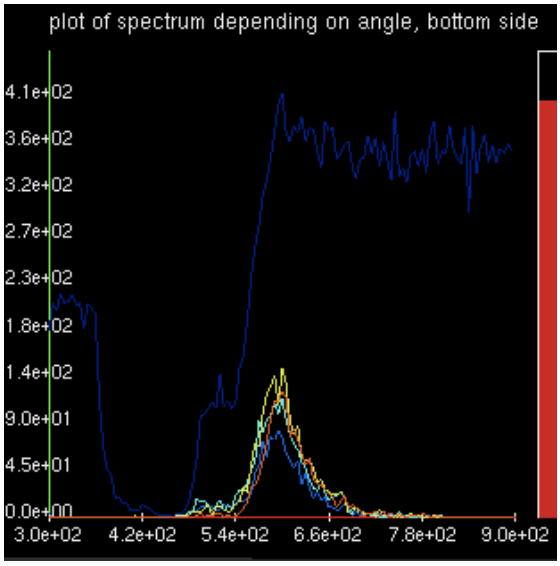


The spectrum at the samples border shows only certain wavelength were selected. The selected wavelengths result from the absorption and reemission data. The peak between 600 and 610 nm corresponds to the measured peak.

3.2.2 Transmission

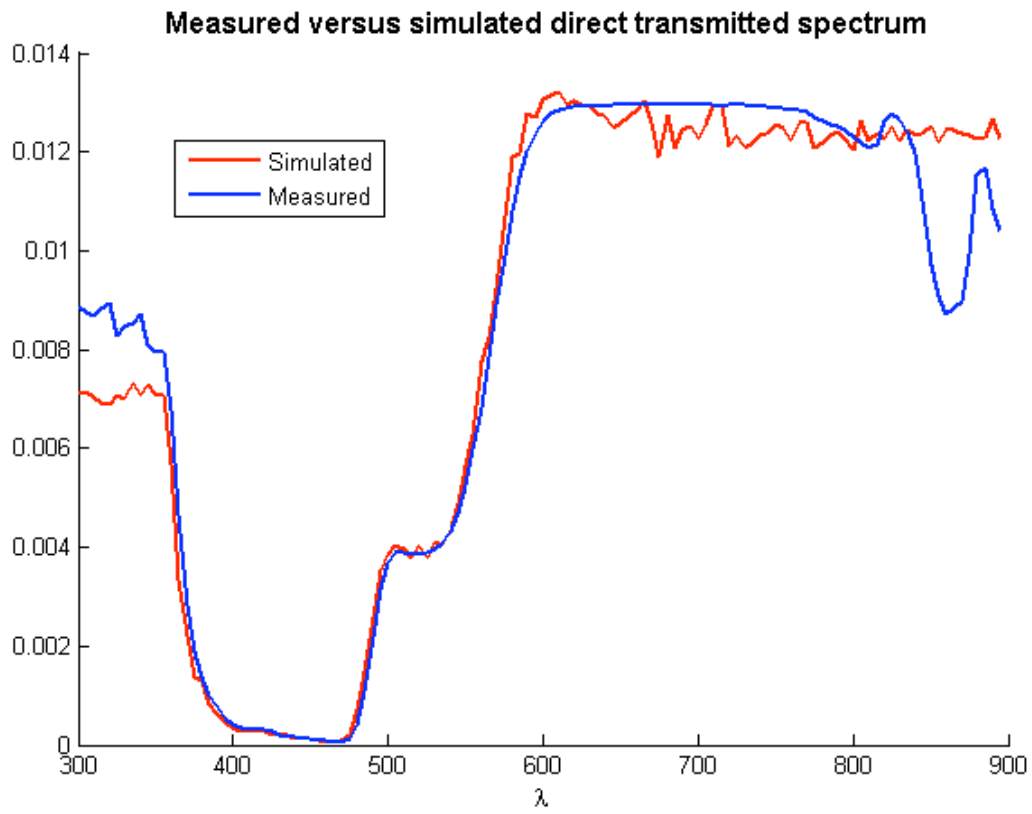
The values measured on the other side off the sample show the transmission spectrum:





Left the total transmitted spectrum is shown, right the transmitted spectrum depending on the angle.

There is a selection on the incoming spectrum, certain wavelength are completely absorbed (between 400 and 500 nm) some are transmitted. The peak around 600 nm is the consequence of the reemission spectrum distribution (shown in section 1.3.4). If we look at the direct transmitted spectrum, this peak is not visible (right hand side screen shot, dark blue line: the spectrum for polar angles between 0° and 18°). Whereas the only transmitted light at angles different from the normal, the diffuse transmission shows only this peak.



If these spectrum are compared to the measured ones, they correspond. The simulated direct transmission spectrum matches exactly the measured one. The difference beneath 380 and above 800 is due to the way the absorption coefficient was characterized: due to the incorrect measures out of this range, it was fixed to the last known value (correct measures can be obtained for other wavelengths but they are not as interesting to us and where ignored for now).

The total transmitted spectrum corresponds qualitatively but the peak around 600 *nm* due to fluorescence is exaggerated. this quantitative difference is not surprising as the quantum yield used in those simulation was determined arbitrarily. It was esteemed to high and therefore there have been to many reemissions, resulting in an exaggerated peak.

This is one more good validation of the simulation and the model.

CONCLUSION

PhotonSim is now ready to be used. Experimental measures were already reproduced but some more validations can still be done. More data to do this is becoming available.

PhotonSim offers a good model for luminescent solar concentrators. However, some variable are not always known, the internal QY for example has an influence on the results and explains that they correspond qualitatively but not exactly if it was under or over estimated. The developed software offers simulation of entities which are directly accessible in an experiment. By selecting the right measures, some unknowns for a partially characterized material may be estimated by running a simulation, comparing the outputs with the measures, adapting the unknowns and doing this repeatedly until there is a good match between simulation and measures.

In materials with luminescent proprieties, the traditional optical values used in ray tracing such as refraction indices, have lost of their importance. The absorption spectrum and the reemission spectrum play a greater role in defining the optical proprieties of those materials.

With a well characterized luminescent material, the engineer can use PhotonSim to easily modify the dimensions, layout and optical characteristics of a concentrator and estimated the impact on performance. For example the length over thickness ratio or the concentration of the dyes may be varied and optimized. It also offers the possibility to try out different configuration: a volume concentrator could be compared to a layered concentrator where the external layers only have absorbing properties whereas the internal layer conducts light. Stacks of concentrators with different properties using Götzberger's principle can also be simulated.

Outlook

Besides more validation, the adaptation of the software to tune some parameters to match measurements, polarization of light can be added. The approach used is a good approximation as the polarization of light does not change the critical angle, essential in concentrators. However it can result in more accurate and realistic results. Of course optimization of the source code are always possible.

On the long run, to be used with more comfort, a complete GUI may be added to easily tune input parameters.

Acknowledgments

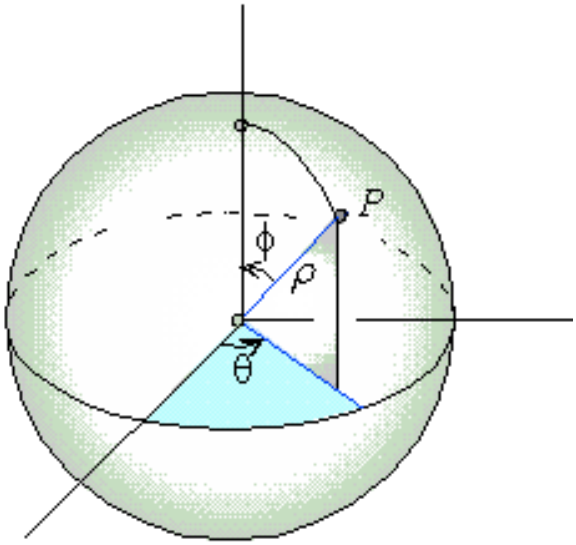
I would like to thank everybody at LESO for welcoming me. Especially Andreas Schüler for his enthusiasm, encouragements and support all along the duration of this project. Jean Louis Scartezzini for accepting me as a masters student in his laboratory, Jérôme Kämpf for his help to get some understanding of Radiance. Miguel Valle Del Olmo, Charudatta Galande, Jessen Page and everybody in the LE hall for the warm environment and the good times we had.

I would also like to thank me friends and family for their support in difficult times and dedicate this report and the work behind it to my remembered friend Sabry.

ANNEXES

A Spherical Angles

The definition for used angles: azimuth, polar and elevation.



Theta ,the **azimuth** angle is the displacement from the north or the top. It is between 0 and 2π.

Phi , the **polar** angle is the angle to the zenith. It is between π/2 and -π/2.

π/2 - Theta , the **elevation** is the angle from the horizontal plane, it is positive and lower than π/2.

B Source Code

Due to its volume, it is available separately, in the PhotonSim reference manual.

C MATLAB code

C-1. MATLAB code to import outputted statistics

```
%// The stat name
nameRoot= 'test_reem';
values = [440 450 460 470 480 550 621];
nameMidle = ' #0 spectrum distribution in first layer, ';
nameEnd = ' side.stat';
outRoot = 'NZO_SIM_';           %//The desired start for outputs
sides = ['L' 'R' 'F' 'N' 'B'];
nbSides = 5;
s = size(values,2);

%// loads the set of data for chose sides, comment undesired lines.
for i = 1:s
    L = load([nameRoot ,num2str(values(i)),nameMidle,'left', nameEnd]);
    R = load([nameRoot ,num2str(values(i)),nameMidle,'right', nameEnd]);
    F = load([nameRoot ,num2str(values(i)),nameMidle,'far', nameEnd]);
    N = load([nameRoot ,num2str(values(i)),nameMidle,'near', nameEnd]);
    B = load([nameRoot, num2str(values(i)),' #0 spectrum distribution, bottom side.stat']);
    for j = 1:nbSides
        evalin('base', [outRoot, num2str(values(i)), '_', sides(j), '=', sides(j)]);
    end
end

%//Writes the sum without bottom into matrix.
for i = 1:s
    evalin('base', [outRoot, num2str(values(i)), '=', outRoot, num2str(values(i)), '_',
                sides(1), ';']);
    for j = 2:nbSides-1
        evalin('base', [outRoot, num2str(values(i)), '(:,2)' '='
                outRoot, num2str(values(i)), '(:,2)+',
                outRoot, num2str(values(i)), '_', sides(j), '(:,2);']);
    end
end
```

C-2. MATLAB CODE TO TRANSFORM RE-EMISSION SPECTRUMS INTO VALID A DATA FILE FOR PHOTONSIM

```
%// Parameters.
rootName = 'NZO';     %// The name of the data
first = 415;          %// The first excitation wavelength also the complement to name
step = 15;            %// -> the concatenation of rootName and first gives the name of the
datas = 18;                   %// matrix containing the reemission spectrum.
INT = 1;                      %// The values to be interpolated at.
acc = 1000;                   %// The accuracy for inverted data.
spectName = 'reemNotzOrange'; %// The name with no spaces!

% // Output File, inverted distribution function.
invFile = [spectName '_INV.txt'];
invFid = fopen(invFile, 'wt');
fprintf(invFid,'NAME %s \n',spectName);
fprintf(invFid,'NB_VALUES %i\n', (acc+1) * datas);
fprintf(invFid,'START %i \n', 0);
fprintf(invFid,'INTERVAL %f \n', 1/acc );
fprintf(invFid,'MULTI_REGULAR %i %f %f\n', datas, first, step);
fprintf(invFid,'\nVALUES \n');

%// Loop an data
for j = 0:(datas-1)
        %// Load the spectrum from data and append starting and ending values.
        name = [rootName num2str(first + j*step)];
        evalin('base', ['Y=',name,'_(:,2);']); Y = [0; Y; 0];
        evalin('base', ['X=',name,'_(:,1);']); X = [300; X; 900];

        for i = 1:size(Y,1)
               if Y(i)<0      Y(i) = 0;      end;
        end;

        s = sum(Y);                     %// Normalize
        Y = Y./s;
        Y = cumtrapz(Y);                % // Integrate using trapeze methode
        Y(size(Y,1)) = 1.00001;         %// avoid having two last values equal

        %// eliminate any other duplicates in the value set by smoothing it.
        for i = 1:size(Y,1)-2
            if Y(i) == Y(i+1)
                k = 2;
                while Y(i) == Y(i+k)
                    k = k + 1;
                end;
                add = (Y(i+k)-Y(i)) / k;
                for h = 1:(k-1)
                    Y(i+h) = Y(i+h) + h*add;
                end;
            end;
        end;
        Y = interp1([Y],[X],[0:1/acc:1]);              %// Inverse and interpolate
        evalin('base', [name '_INV_ = Y;']);
        fprintf(invFid,'% f',Y);
        fprintf(invFid, '\n');
end;

fclose(invFid);
```

BIBLIOGRAPHY

[1] Götzberger et al. "Solar Energy conversion with fluorescent collectors, Institute für Angewandte Festkorperphysik, Frauenhofer-Gesellschaft, Freiburg, 1977.

[2] Earp Alan et al, "Optimisation of three-color luminescent solar concentrator daylighting system" UTS Sydney, 2004.

[3]P. Hovingthon, D. Drouin, R. Gauvin. "CASINO: A new monte Carlo Code in C Language for Electron Bean Interaction - Parts 1", Departement de Genie Mechanique, Université de Sherbrooke, Sherbrook, Québec, Canada. Scanning Vol. 19 1-14 (1997)

[4] L.Wang, S.L Jacques L.Zheng. " MCML Monte Carlo modeling of light transport in multi-layered thissues", Computer Methods & programs in Biomedicine, 47, 131-46 (1995).

[5] M .Matsumoto and T. Nishimura, "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator", ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, January pp. 3-30 (1998)

[6] Eugene Hecht "Optics" (1987)

[7] Dan Bruton "Spectra Code, approximate RGB values for visible Wavelengths" http://www.physics.sfasu.edu/astro/color/spectra.html

[8] National Renewable Energy Laboratory "Reference Solar Spectral Irradiance: Air Mass 1.5"

[9] Eric Weisstein "Wolfram MathWorld" http://mathworld.wolfram.com

[10]CERN "Proton, Pio and Photon Density Functions Library User Manual".

[11] Miguel Valle Del Olmo "Characterization of external quantum yield and lateral energy losses in photoluminescent solar concentrator" EPFL LESO (2006)

Georges S. Fishman "Monte Carlo, Concepts, Algorithms and Apllications" Springer (1999)

Willian H. Press, Saul A. Teukolsky, William T. Vetterling Brian P. Flannery "Numerical Recipes in C"

Marilyn Andersen "Innovative Bidirectional Video-Goniophotometer for advanced fenestration Systems" (2004)

Raphaël Compagnon, "Simulation numériques de systèmes d'éclairage naturel a pénetration laterale" (1993)

"IES lighting Handbook, Reference Volume" (1984)

Laurent Michel "Méthode expérimentale d'évaluation des performances lumineuses de bâtiments" (1999)

Wikipedia www.wikipedia.org.

"The OpenGL Reference Manual: The official Reference Document to OpenGL" aka "The OpenGL bluebook"

"The OpenGL Programming Guide: The Official Guide to Learning OpenGL" aka "The OpenGL redbook"

Eric Veach "Robust Monte Carlo Methods for Light Transport Simulation" (1997)

Jhon J. Streicher, William C. Culverhouse, Martin, S. Dulberg, Robert J. "Modeling the Anatomical Distribution of Sunligh" Photochemistry and Photobiology: Vol. 79, No. 1, pp. 40-47 (2004)

Richard Mitanchey, Pierre Laforgue, Marc Fontoynont, "Lighting Calculations on the Internet using Genelux-Web" Right Light 4. Vol. 1(1997)