TOULOUSE III - PAUL SABATIER UNIVERSITY

MASTER 2 - COMPUTER GRAPHICS AND IMAGE ANALYSIS

NATIONAL INSTITUTE OF INFORMATICS

TOKYO, JAPAN

MASTER THESIS

# Rendering of fluorescent materials using spectral path tracing : Niixtracer, a custom rendering engine

Nolan MESTRES

16 March - 29 August 2018

*Academic supervisor*
Pr. Mathias PAULIN

*Internship supervisor*
Pr. Imari SATO


# Acknowledgements

There are many people I would like to thank, for various reasons.

For the warm welcome within the Digital Content and Media Sciences of the National Institute of Informatics, her expertise, valuable advice and directions, the professor Imari Sato.

The whole research team under the supervision of Pr. Imari Sato, and especially Ilyes Kacher, research assistant at NII, for his help and his enthusiasm, but also for his sense of direction and good places in Tokyo.

Agata Soccini, the best Japanese language student to whom I have ever taught, for her friendship and wisdom.

Also for her friendship, her excellent command of the Japanese language and her patience, Célia Boffo, without whom administrative procedures would have been much more complex.

For her kindness and her personnal commitment to her current and former students, Laurence Bergamini.

Lastly, for their attentive ears and their unfailing support during both sunny and rainy days, Joëlle Mestres and Manon Boschard.

# Contents

# Part 1

# Context of the internship

This master thesis is part of the Computer Graphics and Image Analysis Master's education I followed at Toulouse III - Paul Sabatier university, France. It is based on the work I did as an intern during my last year of Master degree, in a Japanese national research laboratory, the National Institute of Informatics (NII), in Tokyo.

This first part presents the context of the internship and therefore the environment in which I have worked during these six months. This is not directly about the tasks done during this period, but I find this information relevant and necessary for a good understanding of the internship and the research world in Japan, and more particularly in Tokyo.

## 1.1 Tokyo, Japan

Before studying computer science, I studied at the University of Language, Literature and Foreign Civilization (LLCE), with a specialization in Japanese. So it was very soon after my reorientation that I asked about internship opportunities abroad, especially in Japan. It happens that the Institut de Recherche en Informatique de Toulouse (IRIT) has a partnership with the NII in Tokyo. Thus, after having passed the various selection stages, I had the chance to complete my six months of internship in this very special setting.

Tokyo in a few figures: it is more than 39 million inhabitants for an area twenty times larger than that of Paris. The population density is therefore surprisingly much lower. However, it is easy to feel crushed by skyscrapers and some neighbourhoods remain crowded at any time of the day or night.

Fortunately, near the site on which I worked, in the main branch of the laboratory, there are some green spaces, in particular the Imperial Palace and its garden.

Transport and rent being the two highest budgets in Tokyo's life, it was necessary to choose between a long journey time with an apartment of about ten square metres or a tiny room close to the workplace. I opted for the second option but I still do not know if I made the right choice.


Figure 1.1: Views of Tokyo. On the left, distant view of the luxury district, Ginza. On the right, the Korean working-class district Shin-Ōkubo

Figure 1.2: On the left, view from the 18th floor of the laboratory, on the right the building itself

## 1.2 National Institute of Informatics

Once the housing-transport dilemma was resolved, I was able to begin my work as part of the team led by Professor Imari Sato in the Digital Content and Media Sciences division of the laboratory. The team includes several doctoral candidates, research assistants and post-doctoral fellows, for a total of 10 people. Although it was made up of Japanese and foreigners, I only had to interact with non-Japanese people. Indeed, we did not work in the same offices. What I thought would be a good opportunity to practice my Japanese proved to be unfounded, and although a little disappointing, it must be said that communication in English was still much easier.

The team's main field of research is computer vision (especially inverse problems of shape reconstruction), whose state of the art is nowadays strongly influenced by Machine Learning. This is the direction many team members are taking. On my side, after having asked before my arrival about my supervisor's work, I wanted to orient my subject towards another discipline, computer graphics, and more specifically rendering. Indeed, since Professor Imari Sato had worked extensively on fluorescence in the past and was

very interested in this visual effect, we agreed that it would be interesting to expand the team's research area.

## 1.3 A different approach to research

Regarding the various insights I had into the world of research in Europe, whether in Toulouse at the Paul Sabatier University during the Supervised Research Work of Bachelor's Degree third year, in Master's Degree during the different projects, or in Poland at the University of Science and Technology in Krakow, the approach was similar. Here, I was quite surprised to find that the works are rarely individual (which is why my internship is a bit of an exception, working alone), and it is the group that will prevail over the individual (even if it means adding co-authors to publications for no apparent reason).

Nevertheless, even if the benefits of a work must be reflected on the whole group, the environment remains very free and autonomy is a quality *sine qua non* to succeed. In my case, I was totally free from the direction of my work to the definition of the subject itself. Although very appreciable, this freedom in the work also had something to be frightened of, given my lack of experience in conducting research alone and the issues and expectations that weighed on my shoulders. I do not want to convey any misconceptions, but according to my colleagues (mainly of Chinese and Thai nationalities), what I have seen in Japan about the importance of the group can be generalized quite systematically to most parts of Asia.

# Part 2

# Problem and contributions

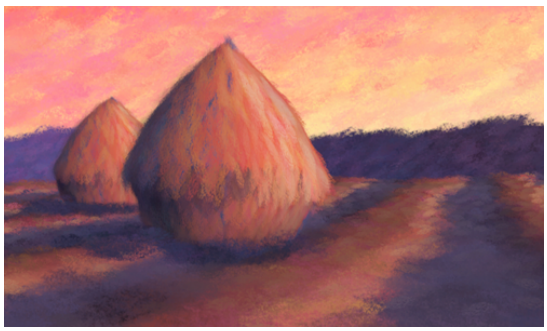
In this section we will first present the field of image synthesis and more particularly that of physically-based rendering, then the subject and objectives of the internship as well as their evolution. Finally, the contributions resulting from this work will be reviewed.

## 2.1 Image synthesis and physically-based rendering

Image synthesis, or rendering, is a discipline of computer graphics whose goal is to generate a two-dimensional image using a computer program, called a rendering engine in the present case. This rendering engine takes as input the description of a 3D scene and produces the corresponding synthetic image as output. The result can be photorealistic or non-photorealistic (Figure 2.1), depending on the chosen technique. These two types of rendering, physically-based rendering (PBR) on the one hand and expressive rendering on the other, address very different issues.

As for us, this master thesis is part of the physically-based rendering. The ultimate goal of a synthetic image resulting from a photorealistic rendering process is to be impossible for a human observer to distinguish from a real-world photography. To do this, the physically-based rendering relies on different disciplines, including optics (the branch of

(a) Rendering of diamonds with reflections and refractions by Josh Wiseman



(b) Painterly rendering for animation from Barbara Meier's project for Disney

Figure 2.1: Examples of physically-based rendering (2.1a) and expressive rendering (2.1b)

physics that deals with light), but also mathematics (linear algebra, probability, numerical integration, trigonometry, etc.), colorimetry, and of course computer science. One of the reference books on physically-based rendering as a whole, on which I relied a lot during this internship, is the book *Physically Based Rendering: From Theory to Implementation* by Matt Pharr and Greg Humphreys [PH10].

More precisely, the synthetic images we generated during this work are the result of an offline rendering process (as opposed to real-time rendering). The fields of application for offline rendering are mainly in the film industry, where the use of computer-generated imagery (CGI) is now omnipresent. Likewise, the images of the animated films are generated in huge computer clusters over several months. Many video game cinematics are also pre-calculated. But entertainment is not the only field to use photorealistic rendering. It is also used in industry, for precise simulation, in architecture for some stages of projects where illustrations and line drawings are insufficient, but also in research where reference renderings are sometimes necessary to verify results, perform error calculations, do 3D reconstruction, etc.

## 2.2 Definition of the subject and objectives

Upon my arrival in Japan, the subject was not clearly defined. Everything had remained very open during the preliminary discussions I had with my supervisor. Unlike the subjects issued by companies in France that I had the opportunity to consult or even those of the Institut de Recherche en Informatique de Toulouse attached to my university, it has never been formulated to me a subject with context and problems. So I was quite confused during that first week in the laboratory, because no one gave me any instructions or leads.

So I understood that the initiative had to come from my side and that if I were to wait for a clear and precise subject to be prepared for me, nothing would happen. What I was sure of was that I wanted to work on rendering and fluorescence. So I prepared a presentation on the subject for the team, at the end of which my supervisor, Professor Imari Sato, and I agreed that it would be interesting to develop an internal rendering engine. This engine should adopt characteristics of physically-based systems and allow the laboratory to perform offline rendering of 3D scenes. Until then, the laboratory used open-source rendering engines such as Mitsuba or PBRT. These systems, although documented, are not easy to handle, and it is even less straightforward to modify them for someone who does not come from the rendering community (which is not the case for the majority of the team members).

### 2.2.1 Evolutions of the subject

Not really knowing what to expect by developing a physically-based rendering engine on my own, I proceeded in successive steps. The objectives therefore evolved throughout the internship, becoming more refined and complex. I started from an empty text file to a simple ray tracing system that used a lot of approximations and simplifications, then to a more complete system, taking into account the usual optical effects (reflections, refractions), color bleeding[1], caustics, spectral rendering, fluorescence, etc.

[1]The color of an object affects those of other nearby objects. These are diffuse inter-reflections due to indirect illumination.

We will later come back to these different features, what they mean and how they work. In the end, I did develop a physically-based rendering engine, in its entirety.

It is through these successive evolutions that the title of my internship became: "**Rendering of fluorescent materials using spectral path tracing: Niixtracer, a custom rendering engine.**"

## 2.3 Contributions

### 2.3.1 Rendering engine

As is it already obvious to everyone, the main contribution that has been implemented over the past six months is a physically-based rendering engine that allows 3D scenes to be rendered. These 3D scenes are described in fine detail by the user in separate scene files, allowing a high degree of freedom in their design. The objects they contain can be made of different materials, fluorescent or not. The scenes described may have different illuminants, as we will discuss later.

The engine itself, developed in C++ from scratch, is a real asset for the team because it will in the future provide many reference images to solve inverse problems in computer vision. It will also allow other team members, current and future, to resume and improve this work, and why not to derive publications from it.

It is also an access for the team to computer graphics and rendering, perhaps leading to interdisciplinary projects and collaborations, as the NII has a direct partnership with IRIT..

### 2.3.2 Rendering of fluorescent liquids

Finally, another contribution we discussed and started working on is the rendering of fluorescent liquids. To do this, it is necessary to extend the existing engine to enable it to render participating media[2] [Rus88]. This ongoing work is based, among other things, on the article by Marwann Abdellah et al. published in June 2017 [Abd+17]. At the moment this feature is not operational, but a code base is present.

[2] A participating media is a media that directly affects light going through it: for example, interactions between photons and suspended particulate matter in the media as it can be the case for smoke, liquids with impurities, etc.

# Part 3

# State of the art

In this third part and its constituent sections, we present and detail the major findings in the field that made the work described in this thesis possible. As the field of offline rendering is vast and relatively old at the scale of computer science, we will not describe here all the contributions related to our work, but only those on which we have relied directly or which have had a crucial impact on the synthesis of photorealistic images.

## 3.1 Rendering equation

It is the integral equation that describes the radiative transfers in a scene (equation 3.1). In our case, it describes the interaction of photons with matter (absorption, emission, diffusion), i.e. the radiance leaving a point on a surface.

It is this equation that the different rendering algorithms attempt to solve. It was simultaneously first introduced in 1986 at the SIGGRAPH conference in Dallas, USA, by David Immel et al. [ICG86], as well as by James T. Kajiya [Kaj86].

It is based on the physical law of conservation of energy: in other words, the distribution of radiance in the scene is in a state of equilibrium. Radiance is the power of radiation emitted or propagated at a point on a surface, per unit area, per unit solid angle (steradian). The unit of the International System of Units for radiance is therefore the $W.m^{-2}.sr^{-1}$.

Thus, this principle of conservation of energy states that the energy leaving a point on a surface is equal to the sum of the energy emitted and the energy reflected from that point.

### 3.1.1 General form

In its most common form, the rendering equation can be written as follows:

$$L_o(p, \omega_o) = L_e(p, \omega_o) + \int_\Omega f_r(p, \omega_o, \omega_i) L_i(p, \omega_i)(\omega_i \cdot \vec{n}) d\omega_i \tag{3.1}$$

With:

- $L_o(p, \omega_o)$ the radiance leaving point $p$, in the direction $\omega_o$;
- $L_e(p, \omega_o)$ the radiance emitted from $p$, in the direction $\omega_o$;
- $L_i(p, \omega_i)$ the radiance coming inward $p$, from the direction $\omega_i$;
- $\Omega$ the integration domain of the directions $\omega_i$, corresponding to the unit hemisphere centered in $\vec{n}$;
- $f_r(p, \omega_o, \omega_i)$ the Bidirectional Reflectance Distribution Function (BRDF) that we will explain in the following section;
- $(\omega_i \cdot \vec{n})$ the dot product between normal $\vec{n}$ at $p$ and the incident direction of light $\omega_i$. This term comes from Lambert's cosine law, which states that the radiance observed at a point on a surface is directly proportional to the cosine of the angle between the direction of the incident light and the normal at the surface for that point.

### 3.1.2 Bidirectional Reflectance Distribution Function

Bidirectional Reflectance Distribution Functions (BRDF) are functions that specify the reflectivity of an opaque surface in a non-ambiguous manner, depending on the geometry of the light rays that hit it. That is, a BRDF is a function of both the direction of the incident light and the direction of the reflected light (see Figure 3.1). They make it possible to define a ratio between the radiance reflected in one direction $\omega_o$ and the incident irradiance from a direction $\omega_i$ (Equation 3.2) [Nic65].

$$f_r(p, \omega_r, \omega_i) = \frac{dL_r(p, \omega_r)}{dE_i(p, \omega_i)} = \frac{dL_r(p, \omega_r)}{L_i(p, \omega_i) \cos\theta_i d\omega_i} \tag{3.2}$$

They are ruled by physical principles:

- Helmholtz's reciprocity, which states that for all pairs of directions $\omega_r$ and $\omega_i$ we have:

$$f_r(p, \omega_r, \omega_i) = f_r(p, \omega_i, \omega_r)$$

This implies that it would be possible to swap source and sensor in a scene. The result of the energy flux measurement (a quantity that describes the power of electromagnetic radiation, in watts) would remain unchanged.

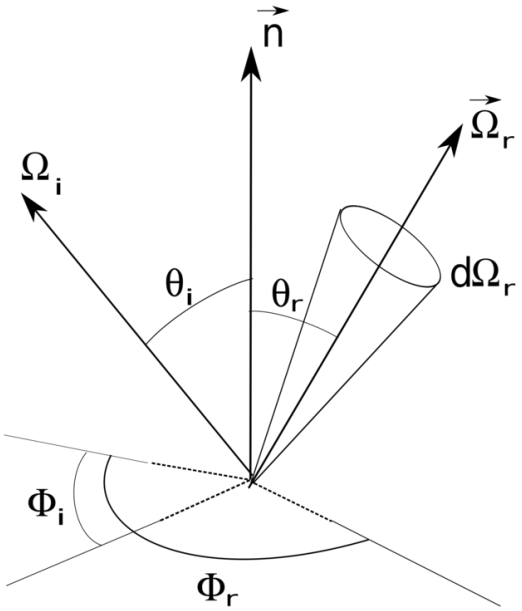



Figure 3.1: Définition et représentation des termes qui composent la BRDF

- the conservation of energy, which states that the reflected radiance is less than or equal to the incident radiance, regardless of the direction of the incident light being considered. Hence:

$$\int_{\Omega} f_r(p, \omega_r, \omega_i) \cos\theta_r d\omega_r \leq 1$$

  Less than or equal, not equal. This seems counter-intuitive considering the fact that this is about conservation of energy, but it should not be forgotten that one of the possible interactions of photons with matter is the absorption of part of the energy by matter, and therefore the chance that not all incident radiance is reflected.

We now have all the keys to understand the rendering equation (Equation 3.1) and its limitations.

### 3.1.3 Limitations

While modelling many real-world visual phenomena, such as color bleeding, caustics, penumbra effects, etc., the rendering equation is nevertheless not devoid of limitations. James T. Kajiya was well aware of this fact as he mentions some of them in his article. In its general form as given above, it is, and I quote, "As an approximation to Maxwell's equation for electromagnetics [the rendering equation] does not attempt to model all interesting optical phenomena." Indeed, no mention is made of time, photons' wavelengths, radiance is considered invariant along a light beam, etc.

Thus, this equation does not allow us to render phenomena that are both common and decisive in achieving the goal set for photorealism. For example:

- participating media and therefore the impact of such media on radiance along light beams;
- the differences regarding the reflectivity of a surface depending on the spectrum of the light that hits it;
- fluorescence, where light absorbed at a certain wavelength is re-emitted at a higher wavelength;
- phosphorescence, where light is absorbed at a time t, before being re-emitted after a certain period of time (about 10 nanoseconds), sometimes at a different wavelength;
- etc.

This is why algorithms that attempt to solve the rendering equation have been the focus of many research activities over the past thirty years. The equation itself has been extended and improved several times to address these limitations, with success in light of the renderings that are now possible!

## 3.2 Ray tracing

Now that we have presented the rendering equation (Equation 3.1) we will discuss a class of algorithms used to solve it (or at least approximate its solution): ray tracing algorithms.

### 3.2.1 Concept

Ray tracing algorithms are based on geometrical optics, which is used to model the behaviour of light. It is an approximation of the true nature of light[1], but it nonetheless makes it possible to account for an important part of the optical phenomena (reflection, refraction,...) that can be observed in everyday life.

The fundamental principle (initially introduced in [App68]) consists in throwing rays (*ray casting*) into a 3D scene. Throwing a ray makes it possible to retrieve information on the first object intersected by it, information that will be used to determine the color of the pixels of the image to be rendered. The origin of the rays is the eye of the observer, or a camera placed in the scene. An image plane is defined between the observer and the scene, and rays are thrown through the pixels of the image. Each pixel will be colored according to the color of the first object intersected by the ray that passes through it. With this method, only visible surfaces are taken into account in the computation of the color (we use the term direct lighting).

#### Mathematical definition of objects

The surfaces of objects that compose the scene are defined mathematically. To check if a ray intersects an object in the scene, it is necessary to browse all the objects and solve the equations of their surfaces for this ray. It is a complex mathematical task and it is this

[1] Dual, wave and particle, nature

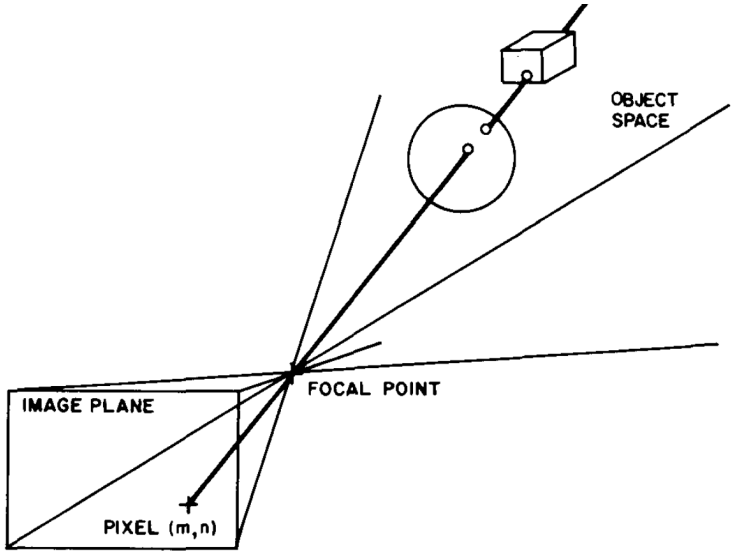



Figure 3.2: Figure issue de [Whi80] montrant le principe du tracé de rayons

step that will monopolize our computing time during the rendering of an image. This is why optimizing it is very important.

#### Extension of the algorithm

This simple algorithm will then be extended by Turner Whitted to take into account reflections, refractions and shadows (Figure 3.2). The calculations no longer stop at the first surface encountered by the ray but are reflected or transmitted (recursive call to the ray tracing function) according to the laws of geometrical optics (Snell-Descartes).

Here, the choice is made not to trace rays from the light sources, as light would actually do in the real world. Indeed, hoping that they will bounce back until they reach the sensor of the device capturing the image is very unlikely and could require an infinite number of bounce.

The shading model used is strongly inspired by the Blinn-Phong model [Bli77]. Although it was at the origin of the most realistic computer-generated images of that time (late 1970s, early 1980s), this model remains very limited, and therefore lacks the realism we aim for.

It was in 1986, at the same time as he presented the rendering equation, that Kajiya proposed an algorithm that computes an unbiased numerical solution to the latter. It is the path tracing. [Kaj86].

### 3.2.2 Path tracing

#### Monte Carlo method

Using of path tracing provides an approximate solution to the rendering equation based on the Monte Carlo method, a stochastic method for solving numerical integration problems. It tends towards the exact solution of the problem.

The key principle is to take a large number of samples for our random variable, evaluate the integral for these values and average the results to obtain an approximation to the integral. The error made is a function of the number of samples taken, but convergence is however slow: let $N$ be the number of samples, increasing $N$ will reduce the error by$\sqrt{N}$.

Let $I$ be the integral with respect to $x$ of a real-valued function $f(x)$ on $\Omega$,

$$I = \int_{\Omega} f(x)dx$$

Then, for $pdf(x)$ a probability density function defined on $\Omega$, we have,

$$I = \int_{\Omega} f(x)dx = \int_{\Omega} \frac{f(x)}{pdf(x)} pdf(x)dx$$

Therefore,

$$I = E[\frac{f(X)}{pdf(X)}] = \lim_{N \to +\infty} \frac{1}{N} \sum_{n=1}^{N} \frac{f(x_i)}{pdf(x_i)}$$

With $E[Y]$ the expected value of $Y$, given that the random variable $X$ has a probability density function $pdf$. Calculate the exact value of $I$ is not possible, only an approximation such as:

$$I \approx \frac{1}{N} \sum_{n=1}^{N} \frac{f(x_i)}{pdf(x_i)}$$

It is this method that will allow us to calculate an estimate of the solution of the rendering equation. It is a very powerful tool because it allows us to numerically solve integral problems of infinite dimension, for which it is not feasible to find a solution in closed-form.

As for the literature on the subject, I mainly relied on the course materials of Professor Mathias Paulin [Pau17], my academic supervisor for this thesis, and their references, with among others, the book by Hammersley and Handscomb [HH65].

### Sampling methods

The path tracing being an algorithm based on Monte Carlo methods, we are led to perform many probabilistic draws. Rather than using a uniform distribution, for example to determine in which direction of the hemisphere centred on the normal $\vec{N}$ at $p$ (Figure 3.3) a ray bounces, more efficient sampling methods can be used. These methods include stratified sampling or low-discrepancy sequences[2]. Such sampling methods are preferred to a conventional uniform distribution because they ensure the correct distribution of samples over the domain under consideration. This reduces the variance, which allows us to increase the accuracy of the rendering for the same number of samples.

[2]Discrepancy is the maximum deviation between samples drawn in a domain from samples that would be evenly distributed over the same domain.

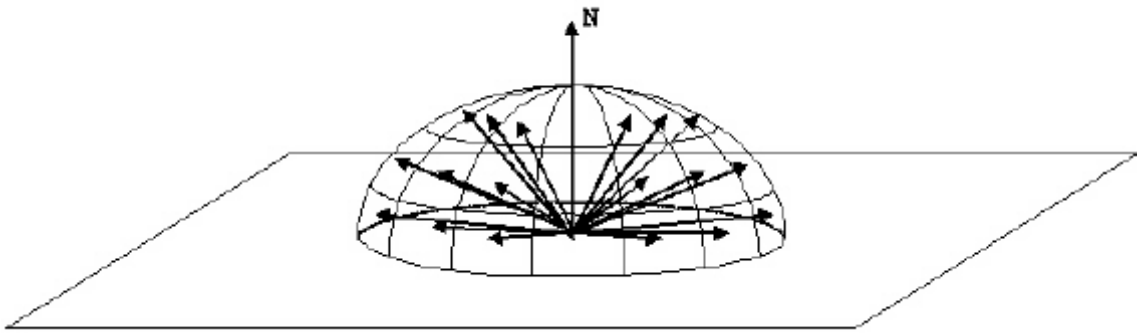



Figure 3.3: Sampling of the hemisphere centered on the normal $\vec{N}$ at a point $p$ of the surface

In the more specific case of low-discrepancy sequences (Sobol sequence, Halton sequence, etc.), the term quasi-random sequences is used. They are in fact deterministic, which ensures a certain consistency of the noise between the different renderings generated. This can be useful for creating an animation from several computer-generated images, where the consistency of high-frequency noise between the different images within the animation is important. In addition, the convergence of a so-called quasi-Monte Carlo method is higher ($O(\frac{1}{N})$ versus $O(N^{-0.5})$).

**Russian roulette**

The rendering equation being recursive, it is necessary to find a satisfactory stopping criterion. As we have just seen, the Monte Carlo method is unbiased, but stopping the reflection of rays at a given depth would introduce bias: indeed, there would be a loss of radiance in cases where a path would be stopped too early, and therefore a non-compliance with the law of the conservation of energy. The principle of Russian roulette is used to avoid this inconvenience. Russian roulette allows the stochastic termination of paths. This ensures that our Monte Carlo estimator remains unbiased, and therefore the exact result for $\infty$ samples.

The principle is simple: after a ray has bounced into the scene, the termination of the path being currently traced will be randomly decided, according to a certain probability. This probability, if it is fixed (for example, 90% chance of continuing the path), also results in a bias. It is the same as stopping the path at an arbitrary depth. This probability of pursuing a path must therefore vary. In order to reduce the variance of our resulting pixels, it is possible to choose it proportionally to the value of the reflectivity. This will favour paths that contribute a lot to the final color and, conversely, paths with a low contribution because little light is reflected will have a higher probability of stopping. This ensures that the computing time is not wasted for paths that are known to contribute very little to the final result.

Finally, the radiance is multiplied by the Russian roulette factor (1 over the chosen probability of continuing the path) to take into account the energy loss that occurs when a path is terminated prematurely.

**BRDF models**

Thus, the realism of the rendered image now depends mainly on the correct definition of the materials (and hence of their BRDF), an area in which much effort was put into both before the rendering equation was published and in the 1990s and 2000s. However, this is not directly related to the subject of this thesis and we will therefore limit ourselves here to giving only a reference reviewing the most relevant BRDFs [MU12]. To be noted is the arrival of BRDF Disney the same year, in 2012 (see the 2012 SIGGRAPH course and associated notes *Physically Based Shading at Disney* by Brent Burley).

## 3.3 Colorimetry

Thanks to the path tracing algorithm presented above, we can now obtain an approximate measurement of the radiance that is arriving at each pixel of our image. It is this radiance that in the real world gives information to our brain about the brightness of an object and its color.

What is called radiance here is in fact the result of the integration of spectral radiance (radiance of the surface considered in relation to the wavelength) over the different wavelengths of the visible light spectrum. It is therefore the distribution of this spectral radiance that is partly responsible for our perception of color.

### 3.3.1 Spectral Power Distribution

A Spectral Power Distribution corresponds to the radiant flux per unit of frequency or wavelength. This means it is a power per frequency or per wavelength. Illuminants have different Spectral Power Distributions (SPDs), and are of course of great importance in the final result of a rendering.

The color of a surface can be defined by means of a SPD, thus representing the power at each wavelength of the visible spectrum. A spectrum is continuous in reality, but we can only have a discrete sampling here (at 1nm, 5nm, 10nm,...). The accuracy of this discretization has an impact on our perception of the color. Usually, an observer is not able to discern any difference with a step less than or equal to 5 nanometers.

As it currently stands, no mention of the wavelength of light is made in the rendering equation (Equation 3.1). The equation must therefore be extended to take into account light as a spectrum, and no longer be limited to arbitrarily defined RGB colors without any physical reality.

$$L_o(p, \omega_o, \lambda) = L_e(p, \omega_o, \lambda) + \int_\Omega f_r(p, \omega_o, \omega_i, \lambda) L_i(p, \omega_i, \lambda)(\omega_i \cdot \vec{n}) d\omega_i \tag{3.3}$$

With $\lambda$ a fixed wavelength. This implies that to obtain a spectral representation of our color, we will also have to integrate over wavelengths of the visible spectrum.

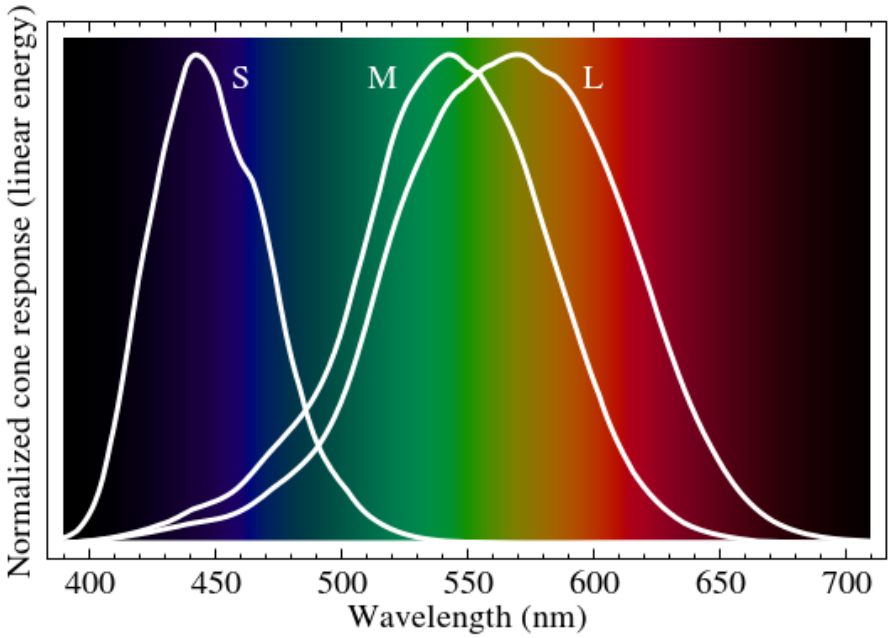



Figure 3.4: Standard response of human cones to monochromatic stimuli in the visible spectrum

### 3.3.2 Human perception of color

Our retina is coated with, among other things, three types of cells whose different spectral sensitivities are responsible for our tri-chromatic day vision: cones S (for small), M (for medium) and L (for long). Each type is sensitive to a certain part of the visible spectrum (see Figure 3.4).

### 3.3.3 From spectrum to pixel color

The International Commission on Illumination (CIE, for its French name *Commission Internationale de l'Éclairage*) has defined over the years different color spaces, models that allow a description of colors in accordance with the tri-chromatic human vision. In 1931, it defined the CIE XYZ space, which makes it possible to obtain three values X, Y and Z from a spectrum. These values, the tri-stimulus XYZ, are similar to the responses of the S, M and L cones. From the Smith and Guild tables, color matching functions have been defined by the CIE allowing the conversion of a spectrum to a XYZ tri-stimulus. It is then possible to convert the tri-stimulus into RGB color space and thus obtain a color that can be displayed by a monitor. For a more detailed explanation see chapters 5.2.1 and 5.2.2 of PBRT [PH10].

The obtained RGB color can still undergo various processes serving different purposes. In the case we are currently interested in, only tone mapping and gamma correction will be discussed.

#### Tone mapping

This is a necessary step because it is sometimes difficult, if not impossible, to reproduce the real brightness of an environment on an image. The contrasts that are perceptible by a human eye go far beyond the capabilities of our monitors. This is where tone mapping comes in. Its purpose is to reduce the large variations in light intensity in the image to preserve its details. The article by Erik Reinhard et al. addresses this problem [Rei+02].

#### Gamma correction

Human vision does not perceive brightness in a linear way. It is easier for the human eye to perceive differences between two close dark tones than between two close light tones. An image encoded without gamma correction could therefore be problematic since without correction, as much importance is given to dark areas as to bright ones. Gamma correction is therefore a way of encoding the colors of an image according to a power function close to how we perceive luminance (and hence contrasts).

## 3.4 Fluorescence

In this section we present the theoretical concepts related to fluorescence that are essential to a proper understanding of the problem. It is not intended to be exhaustive and will not detail the physics on which the fluorescence mechanisms are based.

Fluorescence is a physical phenomenon that characterizes materials and substances that absorb light at a given wavelength and re-emit it at a (almost always) higher wavelength. The first observation of this phenomenon was made in 1560 by Bernardino de Sahagún, but the term fluorescence only appears much later, in 1852, introduced in a book by George Gabriel Stokes [Sto52].

### 3.4.1 Stokes shift

Stokes shift, applied to fluorescence, represents the difference between the positions of the two peaks of a material's absorption and emission spectra (Figure 3.5). These two spectra are most of the time using wavelengths, hence the Stokes shift being a value in nanometers. When a system absorbs a photon of a wavelength of $\lambda_a$, it enters an excited state and will seek to return to its lower-energy state (ground state). Several options are possible, such as the emission of a photon, the dissipation of energy in the form of heat, etc. It takes a fluorophore[3] a time of about $10^{-8}$ seconds to return to the ground state after excitation. It will therefore emit a photon of lower energy at the end of this time (of a wavelength $\lambda_e > \lambda_a$[4]). The energy of the emitted photon is lower because part of it will have propagated to nearby molecules by vibrational relaxation or dissipated as heat. It is this phenomenon that is at the origin of the Stokes shift.

[3] Fluorescent chemical compound. It is generally composed of several molecules and is able to re-emit light after excitation.

[4] Because $\lambda = c/v$, with $c$ the speed of light and $v$ the frequency of the wave.

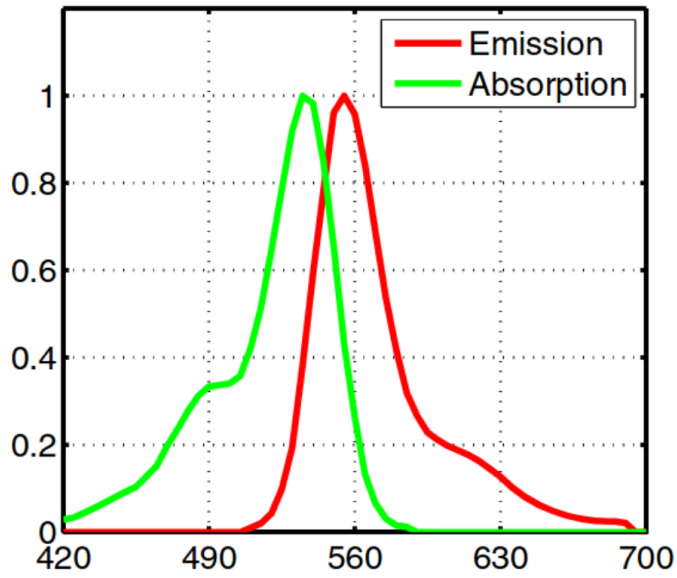



Figure 3.5: Stokes shift between the absorption and emission spectra of a fluorophore (y-axis: intensity; x-axis: wavelength)

### 3.4.2 The Glassner model

The problem of rendering fluorescence is not a recent one. It was theorized and modelled nearly a quarter of a century ago by Glassner [S G94] but was then too complex to be integrated into the rendering engines of the time. Indeed, taking this effect into account when generating a synthetic image is highly resource-intensive: this requires the use of a model that takes into account the wavelength of the light, and therefore to integrate over the wavelengths to solve the rendering equation, thus considerably increasing rendering time and the complexity of the whole system.

We can therefore extend the spectral rendering equation (Equation 3.3) to take into account the fluorescence of a surface as follows:

$$L_o(p,\omega_o,\lambda) = L_e(p,\omega_o,\lambda) + \int_\Omega f_r(p,\omega_o,\omega_i,\lambda) \int_\Lambda f_f(p,\lambda' \to \lambda) L_i(p,\omega_i,\lambda) d\lambda' (\omega_i \cdot \vec{n}) d\omega_i \tag{3.4}$$

With:

- $\Lambda$ the domain of wavelengths of the visible spectrum of light;
- $f_f(p,\lambda' \to \lambda)$, a factor that Glassner calls the fluorescence efficiency. It represents the proportion of energy transferred between the wavelengths $\lambda'$ and $\lambda$. In other words, the amount of energy absorbed by the wavelengths $\lambda'$ of the domain $\Lambda$ on which the integration is performed and re-emitted at the wavelength $\lambda$.

To be noted, an important property of fluorescent surfaces highlighted by the computer graphics community, and not the physics community, which focuses its observations on the molecular level of the fluorescence phenomenon rather than on its macroscopic (visual) aspect: the energy that is radiated from a surface by fluorescence radiates in a diffuse manner (uniformly in all directions). Thus, the specular component will never undergo a wavelength shift due to fluorescence because the light is not absorbed

and therefore not affected by the molecules responsible for the energy loss of the photons [Wil+06].

Thanks to this property, new analytical BRDFs have been modelled to render fluorescent materials ever more accurately.

# Part 4

# Niixtracer, a physically-based rendering engine

This fourth part presents the system developed during this internship, the Niixtracer rendering engine. Its most relevant functionalities are detailed, in line with the state of the art established previously.

I wrote the engine in total autonomy and am the author of all its features to this day. I used the following tool chain for the development: UNIX environment, compilation with CMake and g++, the QtCreator IDE. I used git for source code management. It is available on GitHub and of course is open to contributions.

## 4.1 Features

This section provides an overview of the features implemented in the rendering engine developed for the laboratory.

### 4.1.1 Scene files

A scene file is a way to textually represent a 3D virtual scene. It is therefore a means of describing a 3D virtual scene that a human being can easily grasp. It is this scene file that the rendering engine is responsible for browsing and parsing. It will, from the information collected in the scene file, compute the corresponding output synthetic image.

#### Qt XML

The part of the engine responsible for parsing scene files is the only one in the system that relies on an external library. The library in question is the Qt utility library, and more specifically the Qt XML module. Having already used this library to read and write scene files, it was an obvious choice.

The engine then reads the scene file in XML and, if it is well formatted, instantiates the necessary data structures to be able to generate the image.

A more detailed explanation of the format used to define Niixtracer-readable scenes is provided in the appendix (see Appendix A).

#### Geometric shapes, planes and spheres

The geometric shapes supported by Niixtracer are very simple and there are only two of them: planes and spheres. Nevertheless, the architecture as it is designed allows other shapes to be easily implemented: cylinders, parallelepipeds, triangular faces of a model, etc.

Since the purpose of the engine is not to render very diversified scenes, spheres and planes are sufficient: more shapes are not at all essential to observe the fluorescence effects that interest us in the context of this thesis.

### 4.1.2 Ray tracing

The rendering engine developed permits to render a scene using a classical recursive ray tracing algorithm, as presented by Whitted [Whi80]. It uses optical geometry to take into account reflections, refractions and shadows. For an example of the implemented effects, see figure 4.1. Other renderings are available in appendices B.1 and B.2.

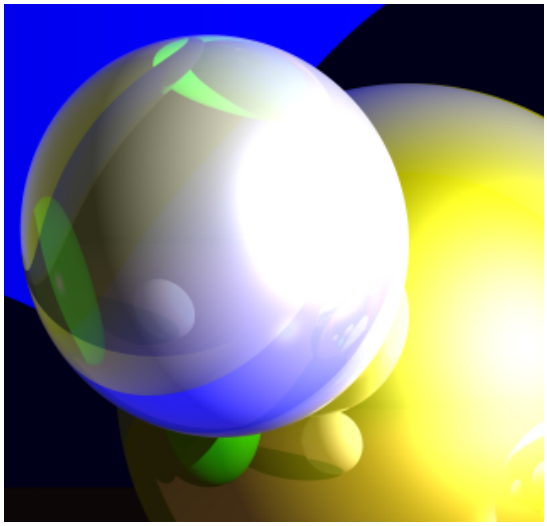
#### Reflections and refractions

To know the proportion of reflected light in relation to the proportion of transmitted light, it is necessary to solve the Fresnel equations. They provide us with information on the behavior of a wave at the interface between two media, in this case the surface of an object in our scene. For faster results to compute it is possible to use the Schlick approximation [Sch94]. The behavior of the wave at the interface depends mainly on the refractive indices of the two media traversed by it and the angle of incidence of the wave with the interface.

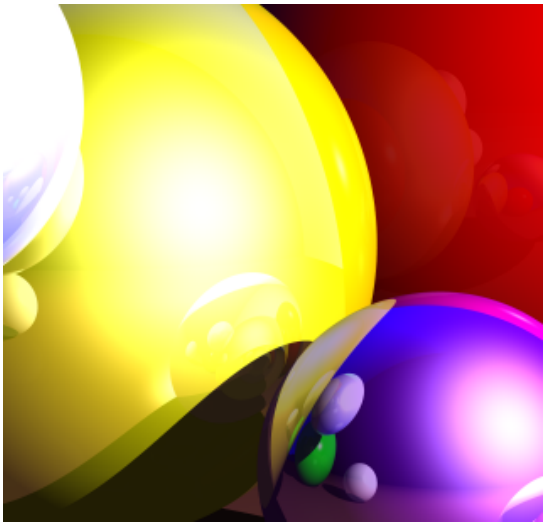
#### Shadows

At each intersection of a ray with an object, a shadow ray is launched with as its origin the point of intersection on the surface and for direction the light source to verify that it is indeed visible (and thus that the point of the intersected surface is not located in the shadows). With this algorithm the light sources used are point lights and therefore have a position in space that can be used as a target to launch our shadow rays.

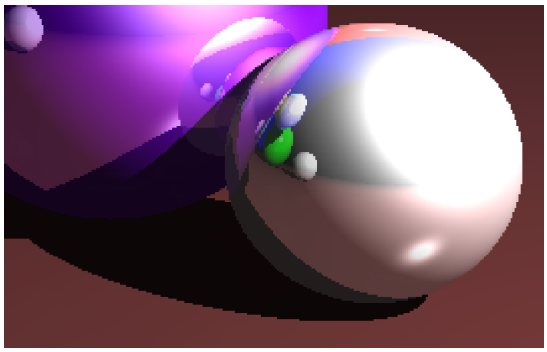
Note that the origin of the recursive rays must be shifted towards the surface to avoid problems of self-shadowing or self-intersection in the case of reflections. To accomplish this, we shift the origin of a fixed epsilon.



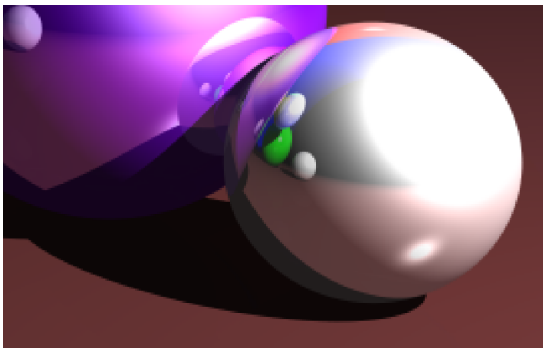
(a) Sphere that refracts light (IoR = 2.4)



(b) Multiple reflections due to max bounces > 1



(c) No antialiasing (pixelation on the contours of the spheres and shadows)



(d) Same as 4.1c but with antialiasing (pixels subdivided 4 times)

Figure 4.1: Details of a scene rendered using the recursive ray tracing algorithm. The green sphere that is reflected is out of the field, on the left. The same goes for the source which is up on the right.

#### Number of bounces of a ray

As mentioned above, since the ray tracing algorithm is recursive, it is necessary to choose when to stop the bounces of the ray in the scene. For this purpose, a maximum number of bounces is defined. This is of course not the best method if the objective is photorealism (see section 3.2.2. which addresses path tracing for this).

#### Antialiasing

This rendering technique also permits to define an antialiasing level (see Figure 4.1c and 4.1d). The method used is fairly straightforward: each pixel is subdivided into smaller sub-pixels and a ray is launched through each of them. Then their contributions are averaged. In this way the contours of the spheres appear smooth and the phenomenon of pixelation is avoided.

### 4.1.3 Path tracing

Another mode that the engine provides to render an image uses a path tracing algorithm, as described by Kajiya [Kaj86]. This time, no more geometrical optics, the different materials and their interactions with light being handled by means of Bidirectional Reflectance Distribution Functions.

#### Bidirectional Reflectance Distribution Functions

There are four BRDFs implemented at this time. They are all analytical BRDFs, derived from mathematical models rather than from experiments as it can sometimes be the case. So we have:

- Perfectly diffuse model;
- Perfectly specular model (mirror);
- Glossy model (diffuse with probability of ideal reflection);
- Ideal refractive model.

These models can be divided into two types of materials: metals and dielectrics.

Metals have a roughness parameter varying between 0 and 1, which allows to obtain more or less shiny surfaces. At zero, all rays are reflected in the ideal specular direction (special case of the mirror). Set to 1, it allows to model an ideally diffuse surface. Such a surface does not exist in reality, but some materials act in a very similar way (clay and chalk for example).

Dielectrics, on the other hand, are insulating media. They can be in different states (solid, liquid, and gas). In the case of solids, the only case of interest to us at the moment, some dielectrics have the interesting property of being translucent (especially glass). They thus have a refractive index specific to each one of them[1].

An example of the implemented materials derived from these two types is shown in figure 4.2.

#### Next Event Estimation

The engine has a feature that allows explicit light sampling (also known as Next Event Estimation, NEE). This technique considerably increases the convergence speed of our algorithm.

The principle is relatively simple: the smaller the area of a source, the less likely it is that a path will reach it. There are therefore scenes for which many paths will have a nil contribution, in other words, computing time used in vain. One solution to this problem is to sample the light explicitly as a shadow ray. For each point on a diffuse surface that would be hit by a ray, a ray is cast directly towards the source. If it is visible, the contribution of light is directly added to the path. For this to work properly and for

[1] List of refractive indices of many dielectrics: https://pixelandpoly.com/ior.html

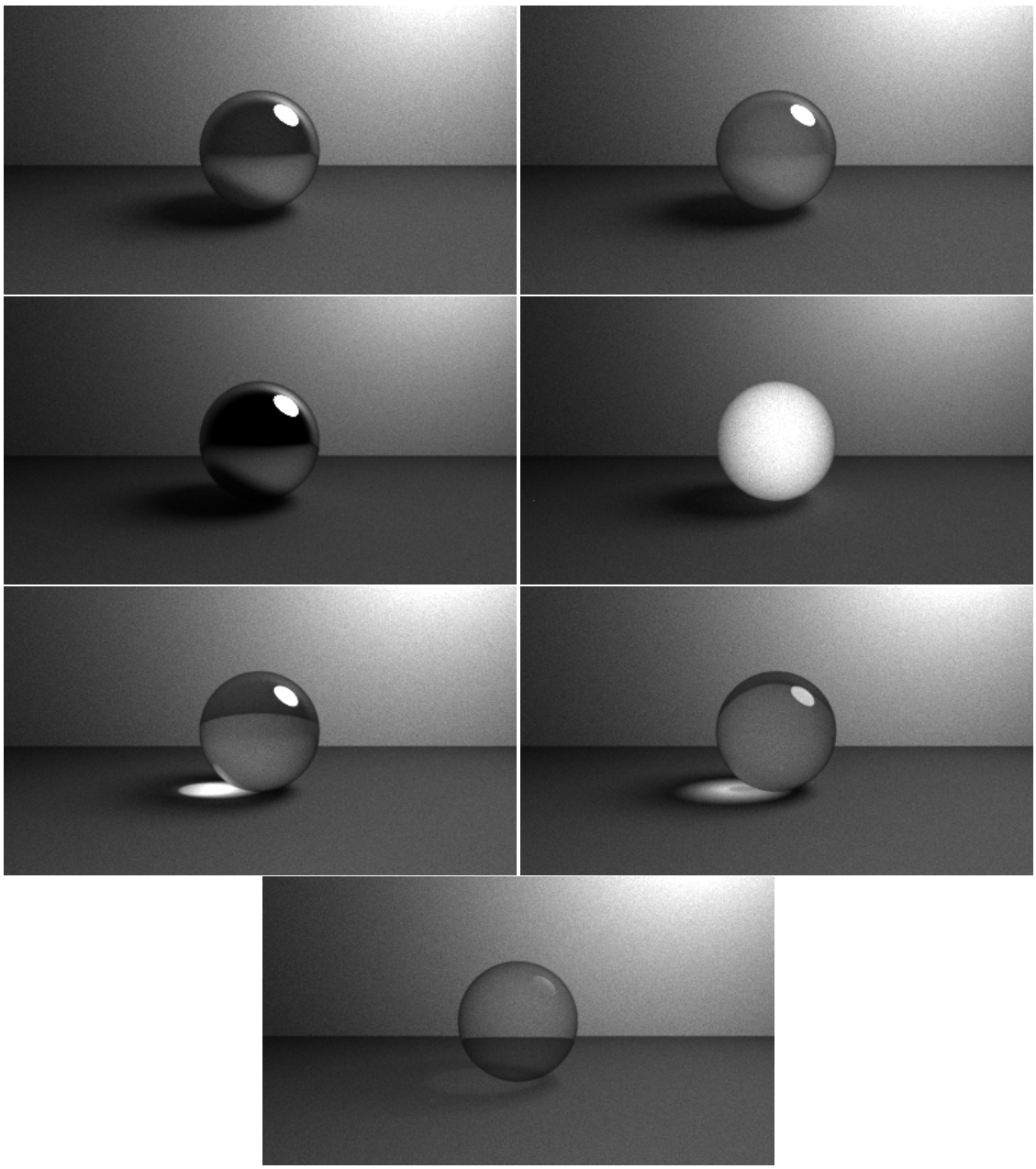


Figure 4.2: Seven different materials, line by line, from left to right: metal (roughness = 0.3), plastic (roughness = 0.7), mirror (roughness = 0), Lambertian (roughness = 1), diamond (IoR = 2.4), glass (IoR = 1.5), transparent (IoR = 1.02)

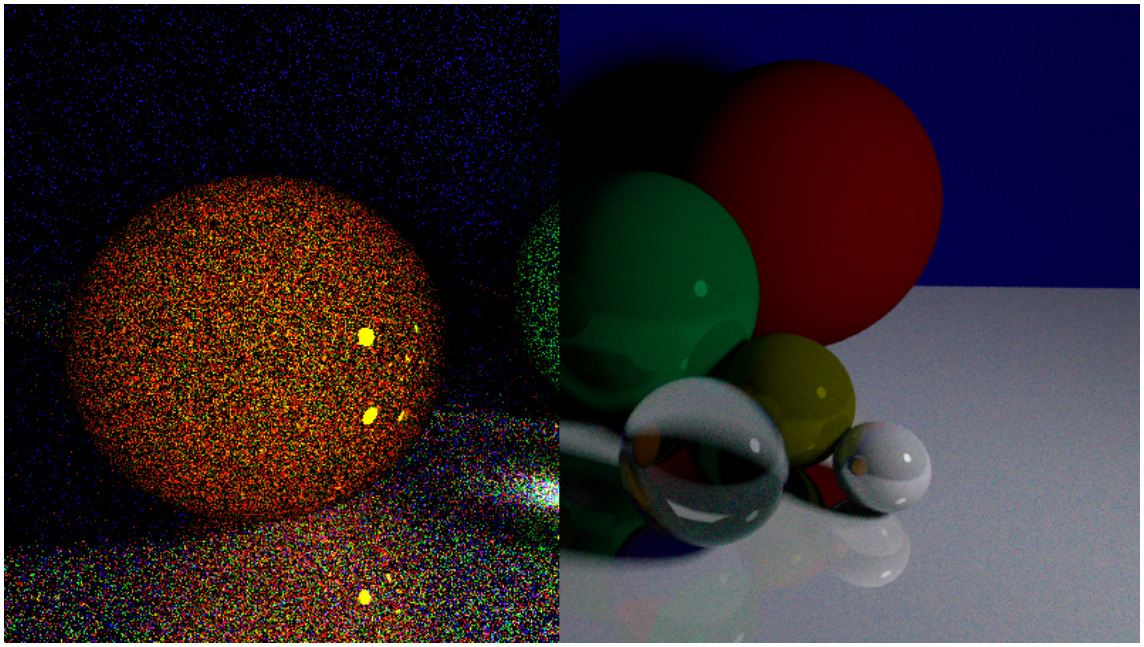


Figure 4.3: Example of the same scene, rendered without (left) and with (right) explicit sampling of sources. To be noted, the presence of the caustic on the image rendered using the standard path tracing. Number of samples per pixel: 1024

the total amount of energy distributed in the scene to remain constant (respect for the conservation of energy, once again), the contribution of the source is no longer added when a path reaches it indirectly. It is useless to add it at this time since it has already been taken into account throughout the path, at each diffuse intersection. This explicit light sampling makes it possible to separate direct and indirect illumination. This can be useful, for example to artists in the film industry, to give them greater control over the rendering of a scene, for example by emphasizing the contributions of direct illumination, or by focusing on indirect contributions from a particular source, at a given number of bounces, etc..

DIn our case, this separation between direct and indirect illumination is not essential. Nevertheless, explicit light sampling is still appreciable because it considerably reduces the variance at the same number of samples, and therefore the number of samples needed to obtain the desired rendering quality (see Figure 4.3). On the other hand, it does not allow caustics to be rendered as it stands. There is an article addressing this problem and proposing a solution to reconnect surfaces for which the source would be occulted by other refractive objects [HDF15] and the light source. To this day, this upgrade is not part of Niixtracer, and is not a priority once again.

### 4.1.4 Spectral path tracing

We now move on to the *spectral path tracing* feature, an extension of the original algorithm. Instead of using a color space such as the RGB space to represent the color of our objects, which has no real physical meaning, the spectral path tracing relies on Spectral Power Distributions to represent the colors.

### Spectral Power Distributions

As presented in the state of the art, a SPD corresponds to the radiant flux (a power) per frequency or per wavelength. This is what is used in the engine to define the proportion of the spectral radiance reflected by a surface at a given wavelength. When using spectral path tracing each object therefore has a reflection spectrum. There are two very simple types of SPDs implemented in the engine.

**Constant distribution** This distribution, as its name suggests, has an invariant energy flow according to wavelength. It can be useful to test how surfaces reflect light at each wavelength. Be careful, if a surface reflects light equally at all wavelengths of the visible spectrum, it will not appear white to us.

**Normal distribution** In this distribution energy is normally distributed. It allows to have a SPD centered on a given wavelength, and therefore to favor a certain color.

$$f(\lambda \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \, \mathrm{e}^{-\frac{1}{2}\left(\frac{\lambda-\mu}{\sigma}\right)^2}$$

With,

- $\lambda$ the considered wavelength;
- $\sigma$ the standard deviation, $\sigma^2$ the variance;
- $\mu$ the expected value.

### Illuminants

Standard illuminants, defined by the CIE, have different SPDs. These illuminants all have their own Correlated Color Temperature[2].

**Planck's law** This CCT makes it possible to obtain, thanks to Planck's law, SPDs close to those of standard illuminants defined by the CIE (see Figure 4.4). This law defines spectral radiance per wavelength emitted by a black body[3] as a function of the temperature (in Kelvin) of that body alone.

Planck's law is expressed as follows:

$$L^{\circ}_{\Omega,\lambda}(\lambda, T) = \frac{2hc^2}{\lambda^5} \frac{1}{\mathrm{e}^{hc/(\lambda k T)} - 1}$$

[2]"The correlated color temperature (CCT) is the temperature of the Planckian radiator whose perceived color most closely resembles that of a given stimulus at the same brightness and under specified viewing conditions" - CIE/IEC 17.4:1987, International Lighting Vocabulary

[3]Wikipédia: "A black body in thermal equilibrium (that is, at a constant temperature) emits electromagnetic radiation called black-body radiation. The radiation is emitted according to Planck's law, meaning that it has a spectrum that is determined by the temperature alone, not by the body's shape or composition."

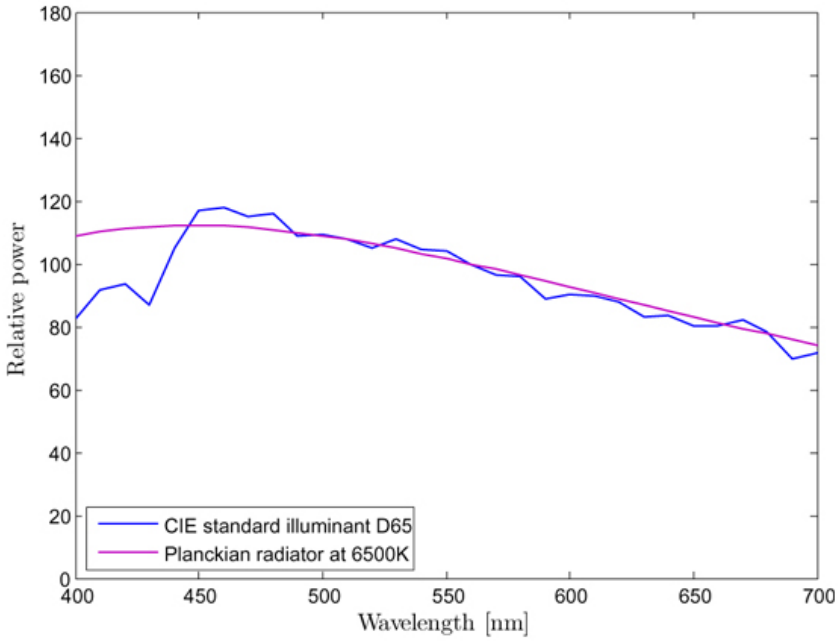



Figure 4.4: Illuminant D65 and a black body of the same temperature SPDs. Figure from www.image-engineering.de

With,

- $\lambda$ the considered wavelength;
- $h$ the Planck's constant, $(= 6.626070040(81).10^{-34} J.s^{-1})$;
- $c$ the speed of light $(= 299792458 m.s^{-1})$;
- $k$ the Boltzmann's constant $(= 1.38064852e - 23 J.K^{-1})$;
- $T$ the temperature of the surface (in Kelvin $K$).

It is clear that the only parameter varying in this formula is the temperature of the black body surface for a given wavelength.

Here are the different illuminants implemented in the engine so far:

**Illuminant A** This illuminant has been designed to correspond to indoor lighting, typically with a tungsten filament bulb. This is why it gives a yellow tint to the scenes in which it is used (see Figure 4.5).

**Illuminants D** The D series of illuminants is an attempt to characterize daylight. It consists of four illuminants:

- Illuminant D50, that represents horizon light;
- Illuminant D55, for the mid-morning or mid-afternoon daylight;
- Illuminant D65, the most common. It represents daylight at noon;

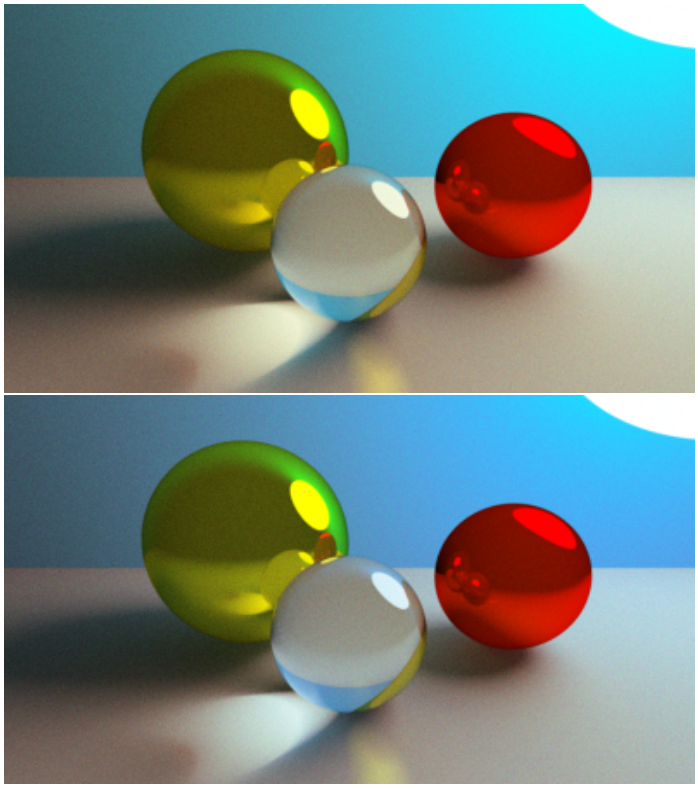


Figure 4.5: Illuminants A (top) and D65 (bottom). The color difference is most noticeable on the caustic and blue plane. Number of samples per pixel: 1024

- And finally the illuminant D75, that represents also daylight at noon, but for a sky closer to the north pole (bluer light, then).

**Black light** What is commonly referred to as black light is a light source that emits mainly ultraviolet light. The SPD of such a source has a very high peak around 370nm (in general, but this depends on the chemical compound used to make the source) and a second lower peak around 400 nanometers, which will produce radiation giving the source that violet-magenta hue characteristic of black lights.

### 4.1.5 Fluorescence

To implement the fluorescence phenomenon, it was necessary to design new materials that have emission and absorption spectra in addition to their BRDF.

#### The skew-Cauchy distribution

In order to obtain emission and absorption spectra that are faithful to the real behavior of fluorescent surfaces, a skewed Cauchy distribution is used (see Figure 4.6), following the three-parameter model proposed by Yinqiang Zheng et al. [ZSS14]. It is based on the following formula:

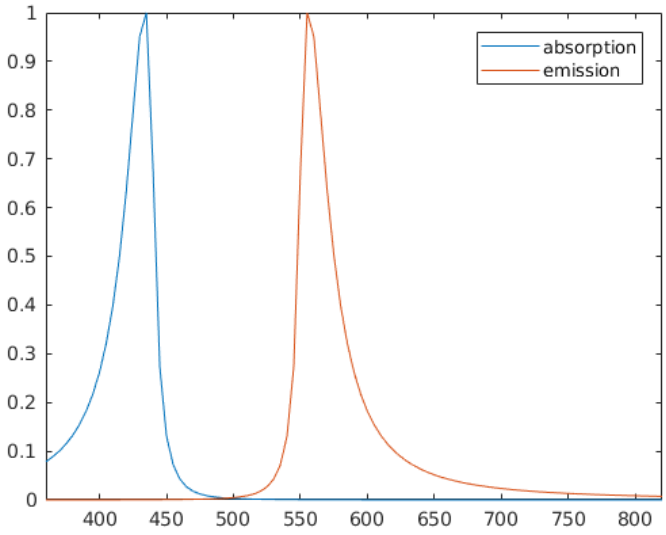



Figure 4.6: Absorption (blue) and emission (red) spectra (material number 50 in the database [McN+06])

$$f(\lambda \mid \lambda_0, \gamma, \omega) = \frac{\gamma}{\pi[\gamma^2 + (\lambda - \lambda_0)^2]}\{\frac{1}{\pi}\arctan[\frac{\omega(\lambda - \lambda_0)}{\gamma}] + \frac{1}{2}\}$$

With,

- $\lambda_0$ the peak-wavelength of the spectrum;
- $\gamma$ the width of the distribution;
- $\omega$ the skewness coefficient of the distribution.

This distribution allows to model both an emission spectrum and an absorption spectrum, as proven in the article. The skewness coefficient must be positive for the emission spectrum and negative for the absorption spectrum ($\omega_a = -\omega_e$).

A rendering of a scene without and with fluorescence is visible in figure 4.7. In this scene, the two colored spheres are fluorescent. The blue sphere on the left absorbs ultraviolet rays and re-emits around wavelengths 540 nm (green), the yellow sphere absorbs in blue wavelengths and re-emits around 680 nm (red). The fluorescent scene gains in total radiance because the absorbed ultraviolet rays are re-emitted into the visible spectrum, which can give the impression that the scene is brighter. It should be noted that, as expected, the specular component is not affected by fluorescence. More renderings and close-ups are available in appendix E.

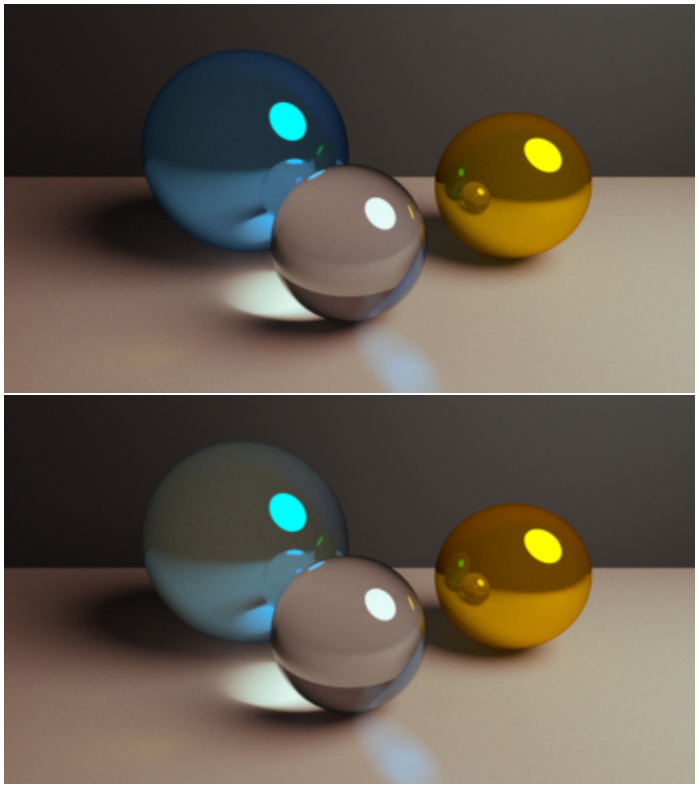


Figure 4.7: Comparison of the same scene without (top) and with (bottom) fluorescence

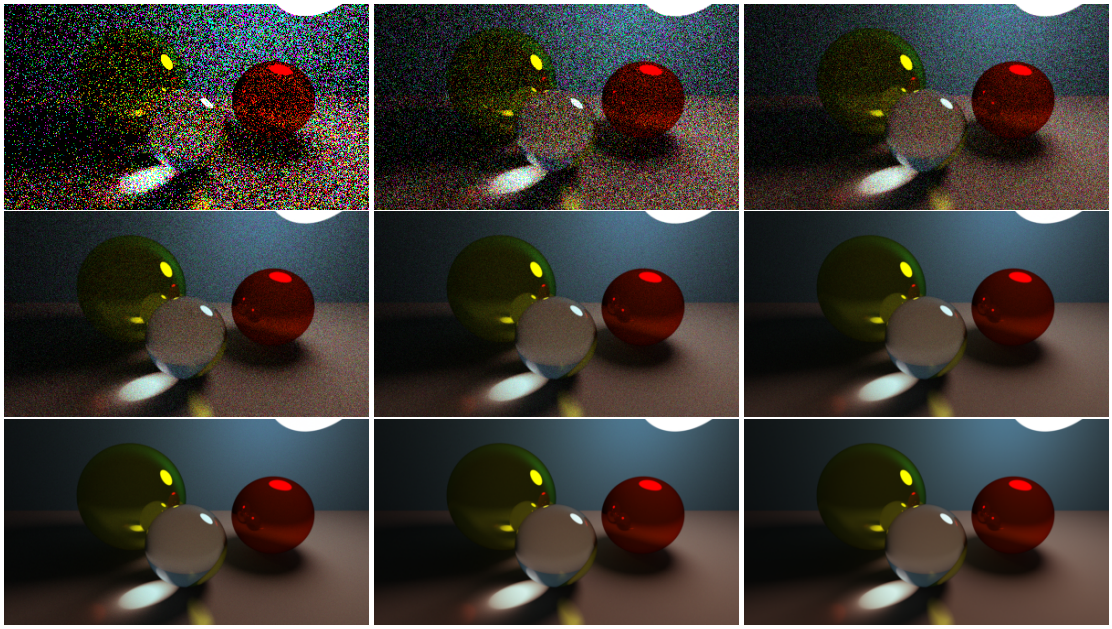
## 4.2 Results

### 4.2.1 Number of samples

The number of samples per pixel used to render an image is directly related to the variance of the image obtained compared to the expected exact result. As we have seen, the convergence rate of Monte Carlo methods is very slow ($O(1/\sqrt{n})$). To divide the error by two, the number of samples must therefore be quadrupled (see Figure 4.8). We can then set an error threshold below which we consider the rendering as correct and stop the algorithm once this threshold is reached.

### 4.2.2 Performance

Performance was not at all at the center of our concerns during development. Indeed, in the long term, the images that will be rendered with the engine by the team will be small in size and the scenes very simple to test measurements made in the laboratory using a spectral camera, for example. Nevertheless, the engine performance remains acceptable and renderings can be made within a humanly acceptable time: about an hour for a rendering with a root of the mean square error around 3. Of course, convergence depends on the complexity of the rendered scene and it is therefore never guaranteed, but the results presented and the rendering speed were considered to be acceptable considering the above mentioned usage specifications.



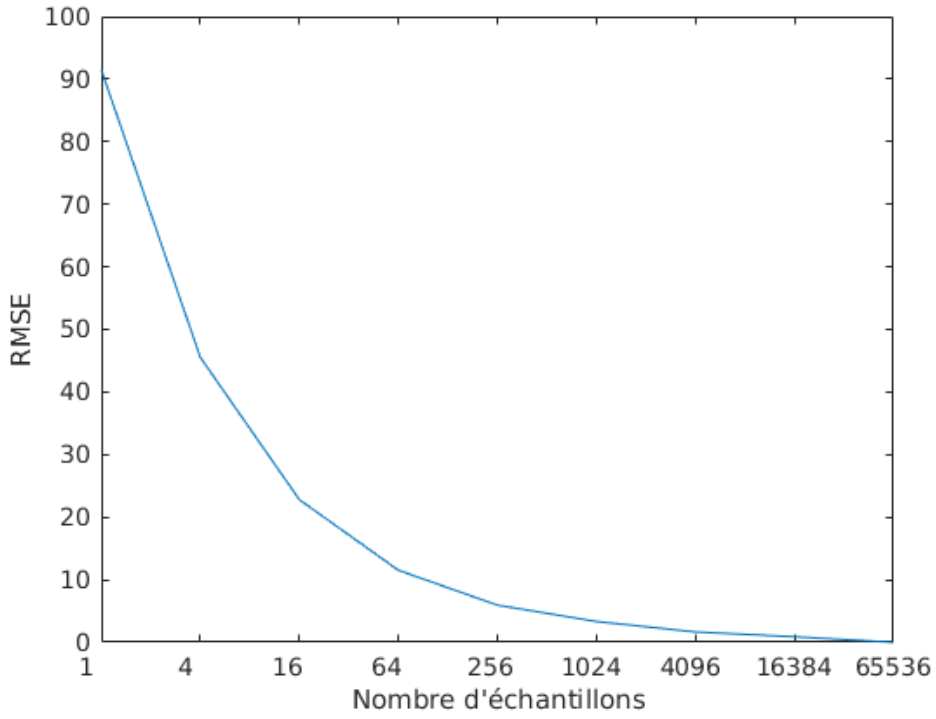



Figure 4.8: Rendering at different numbers of samples of a scene using the spectral path tracing algorithm without next event estimation and a graph of the root of the mean square error relative to the number of samples for that scene.

# Part 5

# Future work

This section addresses the final purposes of the work done during the six months spent at the NII, how it will be useful for the laboratory and how it may evolve in the future.

## 5.1 Rendering of fluorescent liquids

One of the team's current interests, and more specifically Professor Imari Sato's, is the study of fluorescent liquids. Measurements are currently underway to evaluate the BRDF of different liquids that have a fluorescent component. Among them, there are quite surprisingly common liquids, such as honey or whiskey. This is one of the reasons why working in this direction and deepening existing work on the subject is interesting. Fluorescence is a very common effect, yet it is usually ignored despite its significant contribution to the lighting of a scene.

In accordance with this direction, I have searched the literature on the subject and found an article addressing the topic [Abd+17]. This one has not yet been published in a journal (only on `arXiv.org`) but it seemed promising to us and could be implemented in Niixtracer. Similarly, the diffusion properties of many participating media are described in the article by Srinivasa G. Narasimhan et al. [Nar+06] that I have been given the opportunity to study. This database is of prime interest for this forthcoming work.

Once implemented, the engine could be used to generate physically plausible synthetic images of fluorescent participating media, which would allow the laboratory to use it as a reference to test their measurements or to provide input data necessary for solving inverse problems in computer vision (e. g. suppression of inter-reflections for shape recognition).

## 5.2 Iridescence

In the same way as fluorescence, iridescence (see Figure 5.1), or goniochromism, is a relatively common optical phenomenon in nature (butterfly wings, peacocks, bubbles, some insects,...) that has received very little coverage in the field of computer graphics.

Figure 5.1: Shell of a beetle exhibiting a gradual change in color due to the phenomenon of iridescence

To be rendered, it requires a spectral rendering engine, which takes into account the interference and phase shifts of light waves that can occur during reflection phenomena, but also multi-layered surfaces models. This phenomenon can also occur because of the diffraction of light by a surface, as a prism.

Some of the required features are already available in Niixtracer, so iridescence could be an interesting addition.

# Part 6

# Technical and personal assessments

In this sixth part, the results of the internship are presented, both on a technical level for the laboratory and on a personal level.

## 6.1 Technical side

### 6.1.1 For the laboratory

At the end of this internship, the National Institute of Informatics, and more specifically the team I joined within the Digital Content and Media Sciences division, is equipped with a new tool: the Niixtracer rendering engine.

This engine will allow the team to conduct practical experiments, but also to be extended in the future. As my supervisor can supervise up to three international interns per year in the field of computer graphics and computer vision, in addition to the doctoral students she directs and the research assistants she employs, I have no doubt that she will use this opportunity to further develop the project, particularly on the points covered in the section 5. Of course, there are many possible improvements and there is still a lot to be done, but the architecture of the system are strong enough to support it.

Also, I think that my arrival in the team made it possible to reintroduce computer graphics to its members and more particularly the rendering and its problems. Indeed, it is a field that they had abandoned in favor of computer vision and recently machine learning. I was pleased to see that my colleagues were very interested and curious, so I think I can say that this work has been beneficial to all the parties involved, and I would not be surprised to see some of them trying to forge links between the different disciplines.

### 6.1.2 Acquired skills

Thanks to this internship, I was able to acquire and consolidate many technical and social skills, knowledge acquired during my university studies in the field of computer science, and even to revisit older theoretical notions, such as the physics of light and geometric optics, for example. As previously pointed out, the engine developed is written

in C++, a language that I think it is essential to have a good knowledge of (if not to be able to master it, as the task is so challenging) to be able to apply for a position as an engineer in the field of computer graphics. I therefore acquired development skills and was able to work on a project of significant size before I started working. From my point of view, this is the whole point of the internships that are part of our university careers and in this sense, this one has totally fulfilled its mission. Also, very important to me, the many presentations of my work to the team have helped to develop my fluency and the confidence I can place in my abilities.

Being the only one to have contributed to Niixtracer, this allowed me to control all aspects of the project myself and therefore to be involved in all the different stages of the development. I had to ask myself questions about the problem to be addressed by this work, its feasibility, the design of the system itself, and also about more technical details regarding the implementation choices.

## 6.2 Personal side

This internship was a very interesting and formative experience, halfway between research and engineering work. At the conclusion of this one, I have a better understanding of the role I will have to play in a team, and the work expected of an IT engineer. Moreover, the smooth running of the internship in every aspect and the good relations I maintained with my colleagues and my supervisor reinforce my idea that I will be able to find my place in a team, whatever my future position, should it be in a company as well as in the academic world.

I am also very happy and grateful to have been able to live this very special experience in Japan. It is a goal I had set for myself a long time ago and it has affected many of my choices. These six months were far from easy: the distance from my family and friends (ten thousand kilometers as the crow flies and eight hours jet lag), the gigantic size of Tokyo, being forced to step out of my comfort zone and question myself on a daily basis, the language barrier, and so on. All these are subsidiary challenges that have helped me to progress on a personal level. I came out of it deeply marked, changed, but above all convinced that my choices were right and that this is the life I want to lead in the future: working in international environments, surrounded by people who are experts in their respective fields, from whom I have so much to learn.

## 6.3 Future prospects

My professional prospects for the future are not yet fixed. Although initially very interested, I finally declined my supervisor's offer to continue my doctoral studies at the NII, mainly for financial reasons. The salary received by a doctoral student in Japan being clearly insufficient to survive in Tokyo without having to combine small jobs, endless travel times, etc., I did not feel ready for this sacrifice. Also, we discussed my possible return to Japan for the next fiscal year (April 2019), as a research engineer.

It is an interesting compromise, but I would like to carry out a doctoral thesis in the near future, which is why I have not yet taken a decision. Perhaps an ideal solution would be a collaboration between my university and a foreign laboratory (the NII or

other) to continue my work.

Living abroad for extended periods of time teaches me a lot about myself and others, makes me more aware of the good and bad sides of the countries where I have resided, including France, but this has a significant cost: traveling too much is risky because it may no longer feel like you are at home anywhere. Impossible to know which country I will live and work in next, but it doesn't really matter: I learn a little more with each step, and this internship wasn't a faux pas for sure.

# Part 7

# Conclusion

This laboratory research internship is my first work experience beyond the usual two months of the Bachelor's internships. This longer period of time allowed me to explore the state of the art in the field of physically-based rendering and to acquire a better understanding of it.

This overview gave me the keys to define a relevant and formative internship topic for my career: the rendering of fluorescent materials using spectral path tracing. To explore this subject I have developed a physically-based rendering engine that allows spectral rendering and takes fluorescence into account.

Fluorescence is a visual effect that was somewhat neglected by the computer graphics community, yet it is not yet fully covered, as the recent work of Adrian Jarabo and Victor Arellano [JA18] and Michal Mojzík et al. [MFW18] has demonstrated.

Although my contribution is for the laboratory closer to an engineering work it is based on a methodology specific to a research work and could lead to various extensions such as the rendering of non-homogeneous fluorescent liquids.

To conclude, my willingness to pursue a doctorate in an international context is strengthened by this experience and I hope to have the opportunity to continue the work initiated here in the future.

# Appendices

# Appendix A

# Scene files

Definition of the parameters used by the engine (A.1) to render a scene in BMP format, recorded at the PATH location, with XRES*YRES resolution. The computer generated image will use the MODE algorithm, with a maximum number of bounces (except in cases where Russian roulette is used) equal to DEPTH. SAMPLES samples will be drawn per pixel to render the image. In the case of spectral rendering, the ILLUM illuminant must be specified.

Code A.1: Parameters of the algorithm used and the image rendered

```
<scene>      <!-- This tag encloses the entire file -->
  <settings>
    <xres>XRES</xres>
    <yres>YRES</yres>
    <path>PATH</path>
    <mode>MODE</mode>
    <depth>DEPTH</depth>
    <sampl>SAMPLES</sampl>
    <illum>ILLUM</illum>
  </settings>
  ...
</scene>
```

Code A.2: Definition of the camera and the point lights in the scene (ray tracing)

```
...
<camera>
  <position>CPOSX, CPOSY, CPOSZ</position>
  <look_at>LKTX, LKTY, LKTZ</look_at>
</camera>
<light>
  <position>LPOSX, LPOSY, LPOSZ</position>
  <color>LCOLR, LCOLG, LCOLB, LCOLA</color>
</light>
...
```

Then, the camera that will capture the image of the scene is defined by its position in space (CPOSX, CPOSY, CPOSZ) and the point of the scene it looks at (LKTX, LKTY, LKTZ). This camera being a pinhole camera, we can define its position by a single

point in space. For the mode *rt* (ray tracing recursive) light sources are defined as point lights (A.2) using a single position (LPOSX, LPOSY, LPOSZ) and an RGB color (LCOLR, LCOLG, LCOLB, LCOLA).

The objects in the scene (spheres and planes) are defined as in A.3. A plane by its normal (NORMX, NORMY, NORMZ) and its distance to the origin D, a sphere by the position of its center (CENTX, CENTY, CENTZ) and its radius R. The objects have a color (COLR, COLG, COLB, COLA) or a Spectral Power Distribution REFL_SPD depending on the rendering mode used.

Code A.3: Definition of a plane and a sphere

```
...
<plane>
   <normal>NORMX, NORMY, NORMZ</normal>
   <distance>D</distance>
   <color>COLR, COLG, COLB, COLA</color> <!-- ray/path tracing -->
   <reflectance>REFL_SPD</reflectance> <!-- spectral path tracing -->
   <material>MAT</material>
</plane>
<sphere>
   <center>CENTX, CENTY, CENTZ</normal>
   <radius>R</distance>
   <color>COLR, COLG, COLB, COLA</color> <!-- ray/path tracing -->
   <reflectance>REFL_SPD</reflectance> <!-- spectral path tracing -->
   <material>MAT</material>
</sphere>
```

In the case of objects that are light sources, an EMS parameter must be added for the emission of the source, which is either a floating value in the case of ray tracing and path tracing or a Spectral Power Distribution representing the emission spectrum of the source for the spectral path tracing. Example in A.4.

Code A.4: Definition of a source object

```
...
<sphere>
   ...
   <emission>EMS</emission>
</sphere>
```

Finally, for a fluorescent object, it is necessary to define the absorption spectrum ABS_SPD of the object and the S value of its Stokes shift.

Code A.5: Definition of a fluorescent object

```
...
<sphere>
   ...
   <fluorescence>
      <absorption>ABS_SPD</absorption>
      <stokes>S</stokes>
   </fluorescence>
</sphere>
```

Code A.6: A complete example of a scene presenting some of the implemented effects (spectral path tracing/various materials/illuminant D65/fluorescence/...)

```
<scene>
  <settings>
    <xres>400</xres>
    <yres>225</yres>
    <path>test_smcpt_d65_fluo_4096sp.bmp</path>
    <mode>spectral_mcpt</mode>
    <depth>10</depth>
    <sampl>4096</sampl>
    <illum>bbd65</illum>
  </settings>
  <camera>
    <position>0.000000, 5.000000, -6.000000</position>
    <look_at>0.000000, 2.0000000, 2.000000</look_at>
  </camera>
  <plane>
    <normal>0.000000, 0.000000, -1.000000</normal>
    <distance>-10</distance>
    <material>matte</material>
    <reflectance>gray</reflectance>
    <emission>0.0</emission>
  </plane>
  <plane>
    <normal>0.000000, 1.000000, 0.000000</normal>
    <distance>0</distance>
    <material>matte</material>
    <reflectance>const</reflectance>
    <emission>0.0</emission>
  </plane>
  <sphere>
    <center>0.000000, 1.500000, 1.000000</center>
    <radius>1.5</radius>
    <material>diamond</material>
    <reflectance>const</reflectance>
    <emission>0.0</emission>
  </sphere>
  <sphere>
    <center>4.000000, 1.800000, 3.500000</center>
    <radius>1.8</radius>
    <material>metal</material>
    <reflectance>orange</reflectance>
    <emission>0.0</emission>
    <fluorescence>
      <absorption>blue</absorption>
      <stokes>220</stokes>
    </fluorescence>
  </sphere>
  <sphere>
    <center>-2.500000, 2.500000, 4.000000</center>
    <radius>2.5</radius>
    <material>metal</material>
    <reflectance>blue</reflectance>
    <emission>0.0</emission>
    <fluorescence>
      <absorption>purple</absorption>
      <stokes>120</stokes>
```

```
      </fluorescence>
    </sphere>
    <sphere>
      <center>11.000000, 14.000000, -2.500000</center>
      <radius>6</radius>
      <material>matte</material>
      <emission>bbd65</emission>
    </sphere>
  </scene>
```

# Appendix B

# Ray tracing

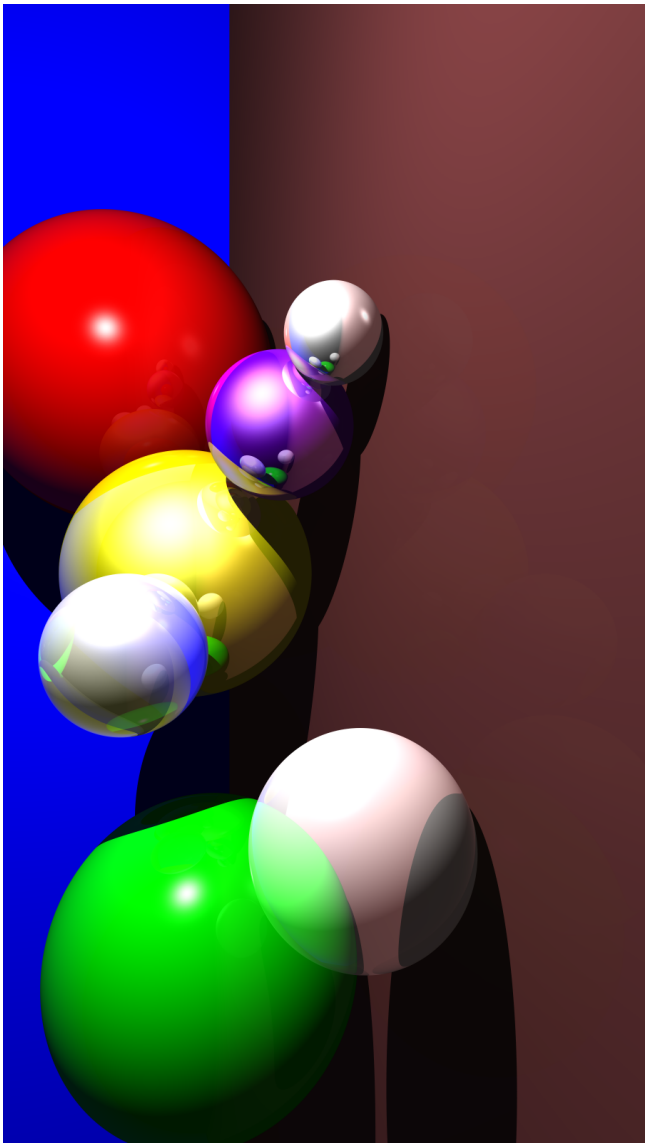


Figure B.1: Example of a scene rendered with the recursive ray tracing algorithm, with reflections, refractions with different indices, antialiasing,... Note that transparent objects cast shadows.

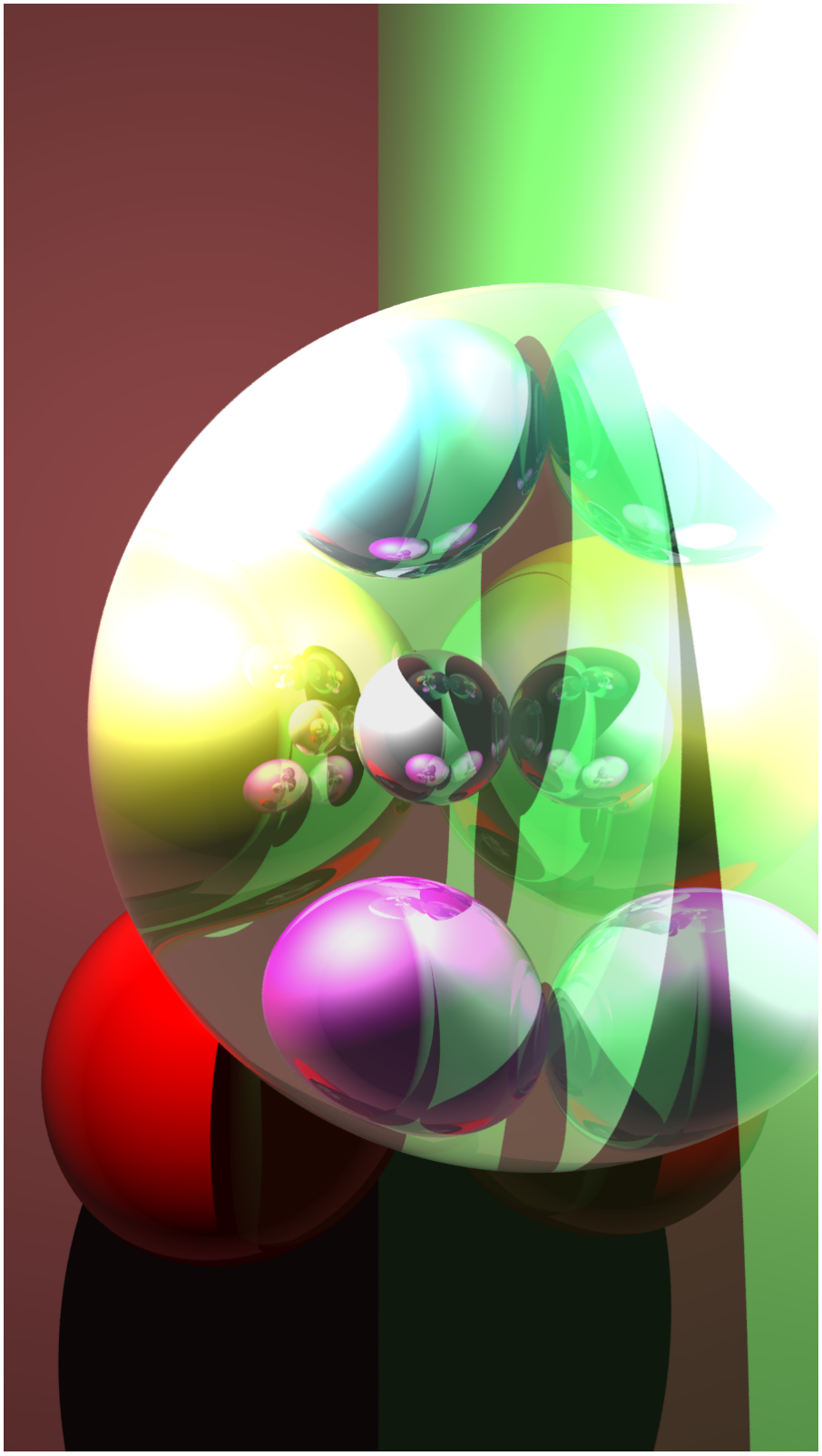


Figure B.2: Example of a scene rendered with the recursive ray tracing algorithm. Four spheres (white, cyan, yellow and magenta) are placed under a dome with a refractive index of 2.4. The yellow sphere seems much more voluminous than the others, this is not the case in reality. Also, the red sphere is not visible behind the dome.

# Appendix C

# Path tracing

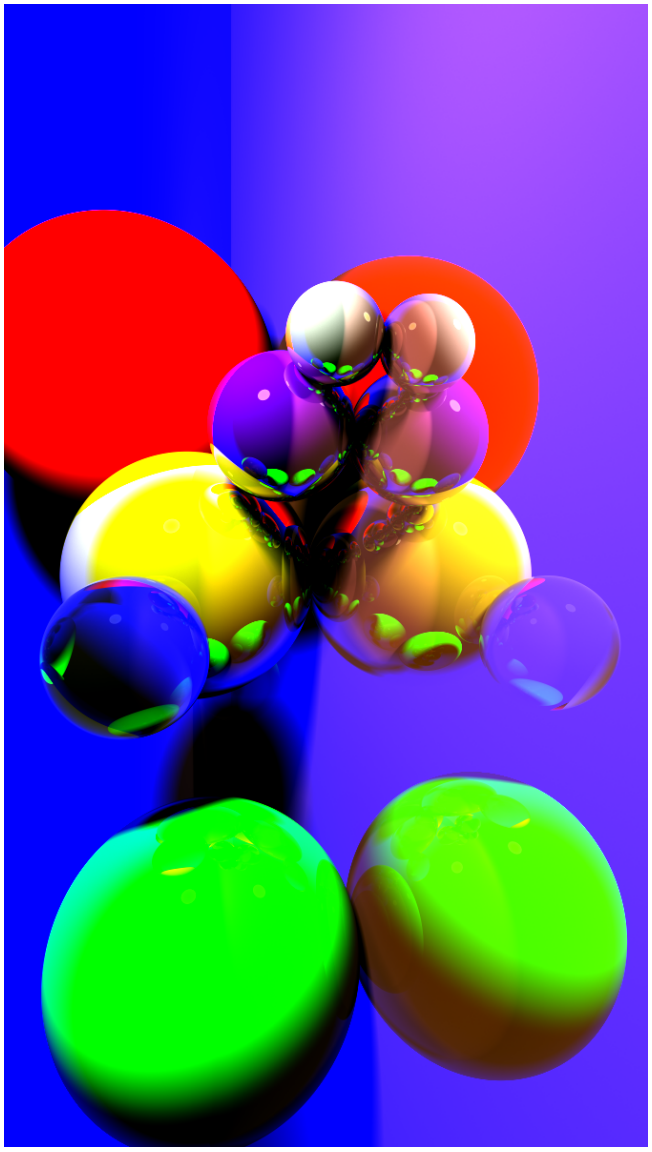


Figure C.1: Example of a scene rendered with the path tracing algorithm, with explicit lightsampling. Note the presence of color bleeding, penumbra, etc.

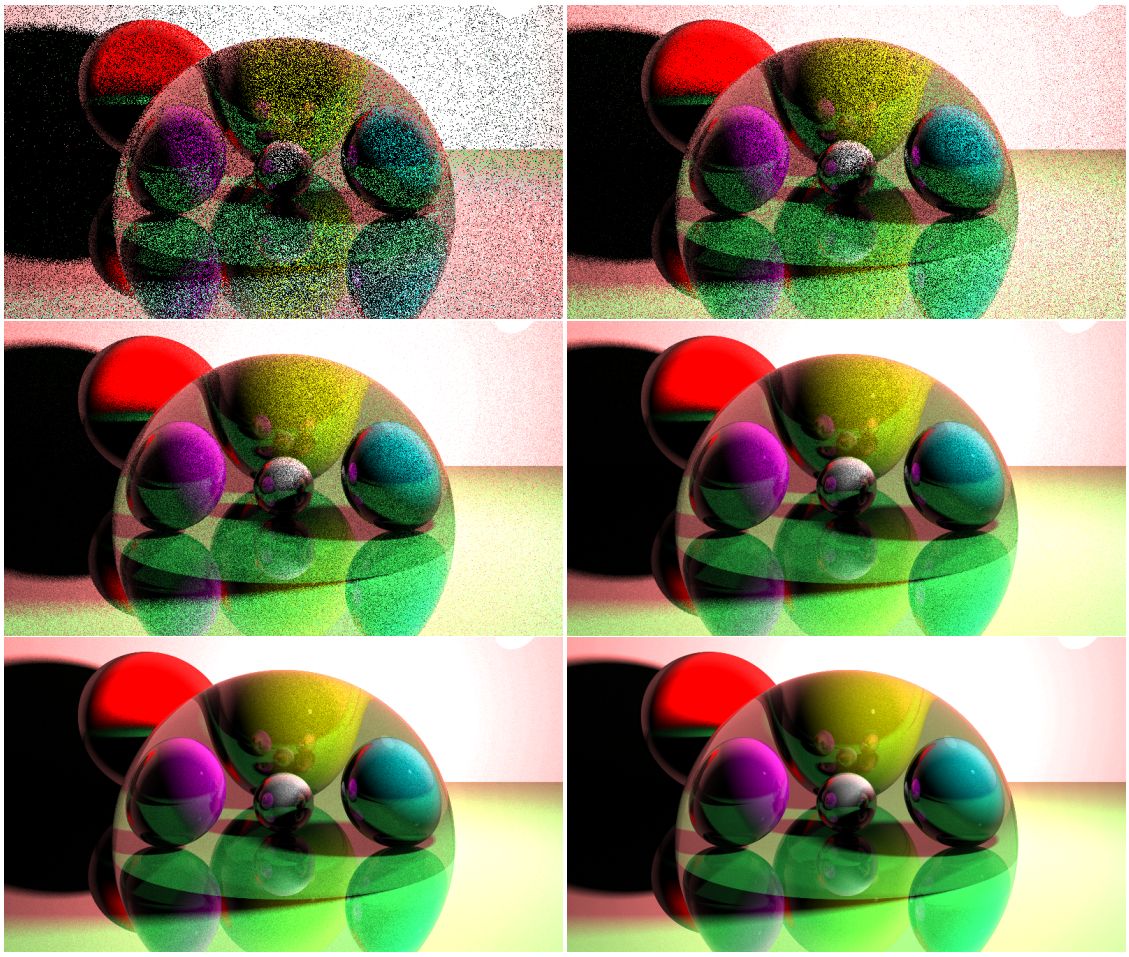


Figure C.2: Scene rendered in path tracing mode with RGB colors and different number of samples per pixel (1 4 16 64 64 256 1024). Note the differences with the ray tracing version of the same scene.

# Appendix D

# Spectral path tracing

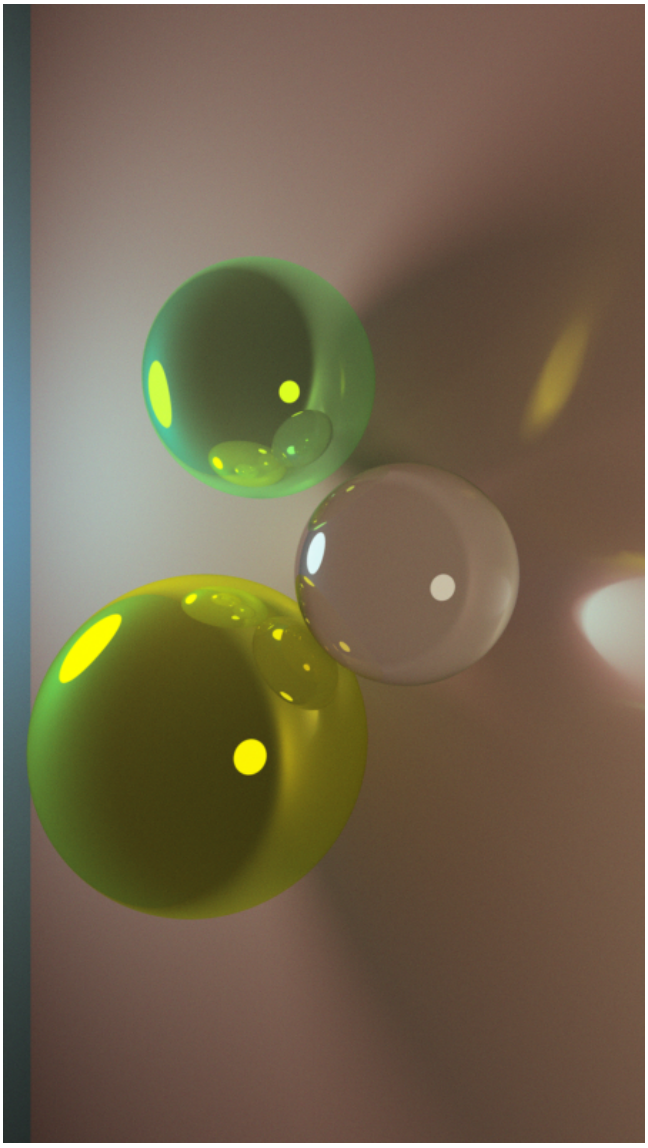


Figure D.1: Example of a scene rendered with the spectral path tracing algorithm. Two sources are present in the scene, an important one that causes shadows and caustic and a small one far behind the camera.

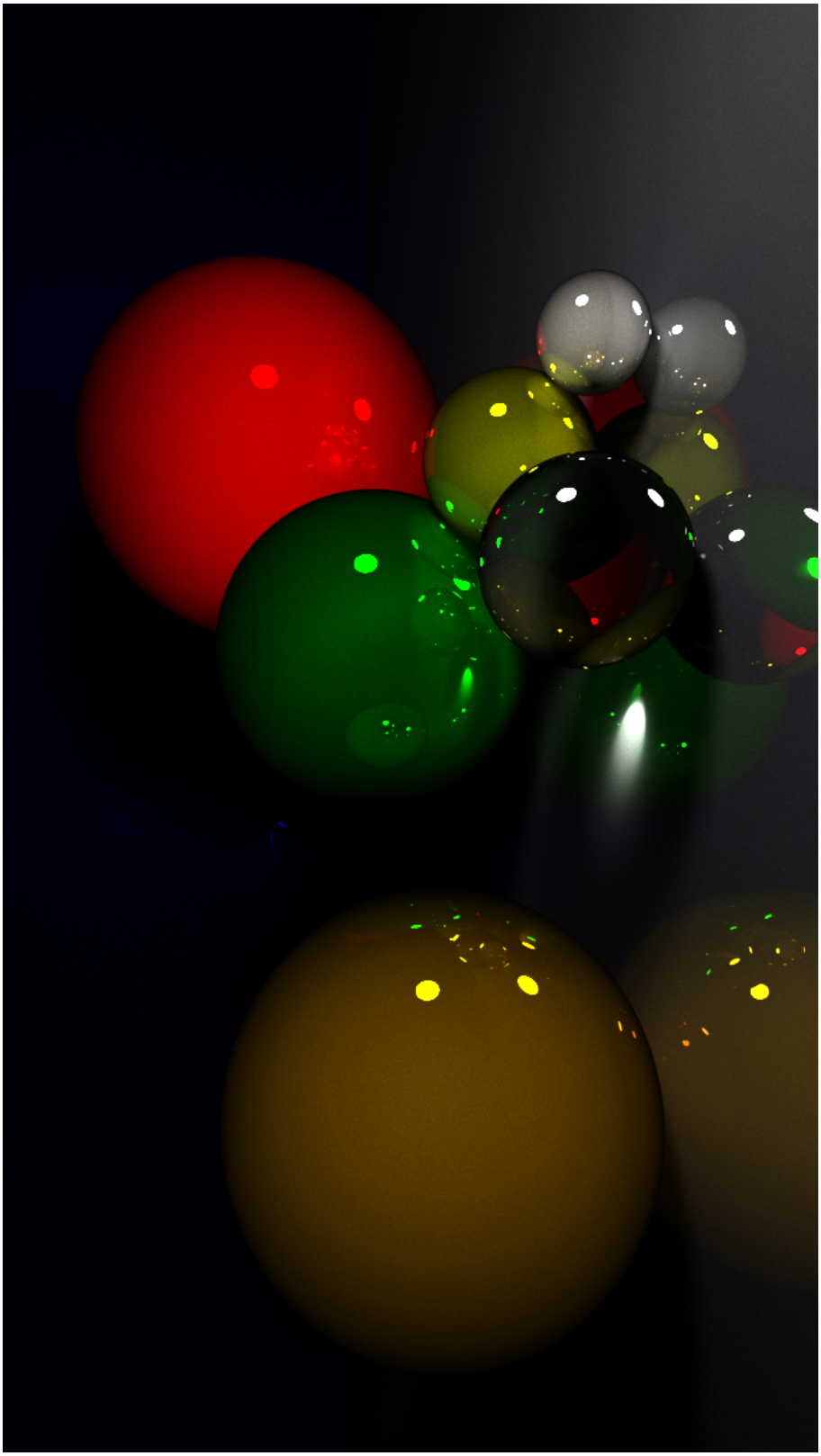


Figure D.2: Example of a scene rendered with the spectral path tracing algorithm. The source is small and distant, so it took a very large number of samples to render this scene: 262144

# Appendix E

# Fluorescence

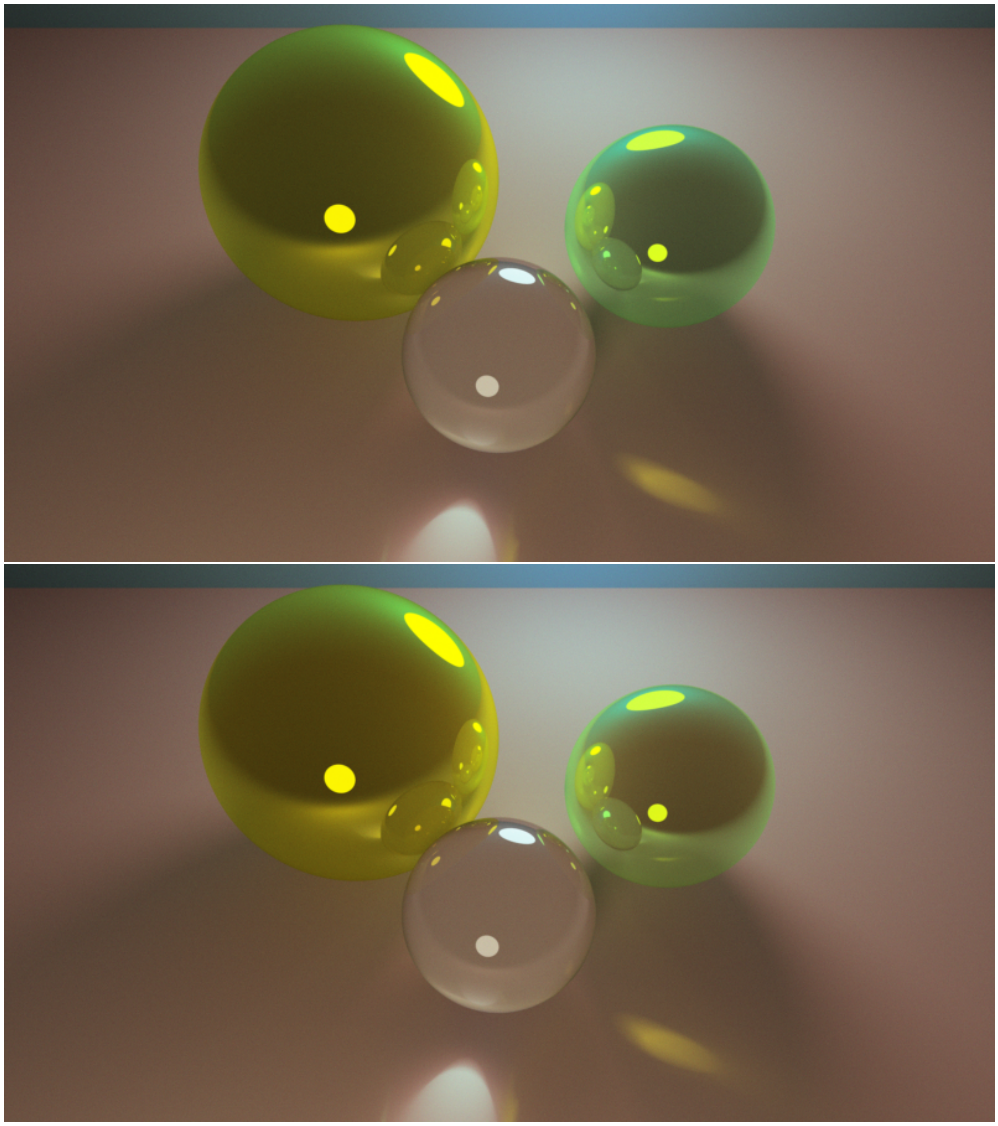


Figure E.1: Comparison between the scene presented in D.1 and its fluorescent version (the yellow and green spheres are both fluorescent, absorb in ultraviolet range and emit in red and orange respectively). Number of samples: 4096

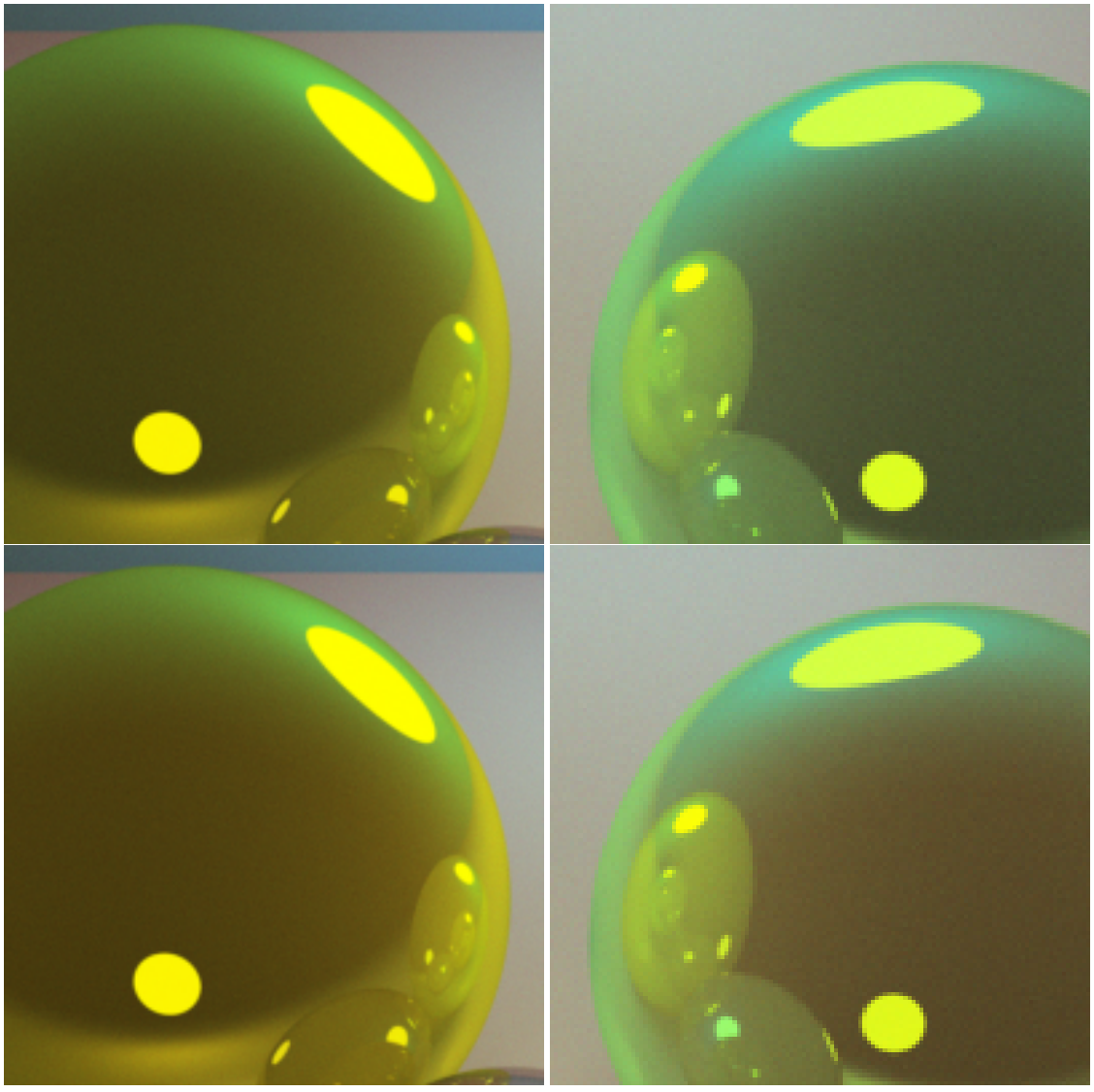


Figure E.2: Details of the scene for a clearer insight into the fluorescence effect (images on the bottom)

# Bibliography


[Abd+17] Marwan Abdellah et al. "A Physically Plausible Model for Rendering Highly Scattering Fluorescent Participating Media". In: *arXiv preprint : arXiv:1706.03024v2* (June 2017) (cit. on pp. 10, 35).

[App68] Arthur Appel. "Some Techniques for Shading Machine Renderings of Solids". In: *Spring Joint Computer Conference* (1968), pp. 37–45 (cit. on p. 14).

[Bli77] James F. Blinn. "Models of Light Reflection for Computer Synthesized Pictures". In: *SIGGRAPH Comput. Graph.* 11.2 (July 1977), pp. 192–198 (cit. on p. 15).

[HDF15] Johannes Hanika, Marc Droske, and Luca Fascione. "Manifold Next Event Estimation". In: *Computer Graphics Forum* (2015) (cit. on p. 28).

[HH65] J. M. Hammersley and D. C. Handscomb. *Monte Carlo methods*. Repr. with minor corrections. London : Chapman and Hall, 1965 (cit. on p. 16).

[ICG86] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. "A Radiosity Method for Non-diffuse Environments". In: *SIGGRAPH Comput. Graph.* 20.4 (Aug. 1986), pp. 133–142 (cit. on p. 11).

[JA18] Adrian Jarabo and Victor Arellano. "Bidirectional Rendering of Vector Light Transport". In: *Computer Graphics Forum* 37.6 (2018), pp. 96–105 (cit. on p. 40).

[Kaj86] James T. Kajiya. "The Rendering Equation". In: *SIGGRAPH Comput. Graph.* 20.4 (Aug. 1986), pp. 143–150 (cit. on pp. 11, 15, 26).

[McN+06] George McNamara et al. "Spectral imaging microscopy web sites and data". In: *Cytometry Part A* 69A.8 (2006), pp. 863–871 (cit. on p. 32).

[MFW18] Michal Mojzík, Alban Fichet, and Alexander Wilkie. "Handling Fluorescence in a Uni-directional Spectral Path Tracer". In: *Computer Graphics Forum* 37.4 (2018), pp. 77–94 (cit. on p. 40).

[MU12] Rosana Montes Soldado and Carlos Ureña Almagro. "An Overview of BRDF Models". In: *http://digibug.ugr.es/handle/10481/19751* (Mar. 2012) (cit. on p. 18).

[Nar+06] Srinivasa G. Narasimhan et al. "Acquiring Scattering Properties of Participating Media by Dilution". In: *ACM Trans. Graph.* 25.3 (July 2006), pp. 1003–1012 (cit. on p. 35).

[Nic65] Fred E. Nicodemus. "Directional Reflectance and Emissivity of an Opaque Surface". In: *Appl. Opt.* 4.7 (July 1965), pp. 767–775 (cit. on p. 12).

[Pau17] Mathias Paulin. "Modélisation et simulation de l'éclairage". In: *Aspects théoriques de l'informatique graphique - Partie Rendu* (2017) (cit. on p. 16).

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. 2nd ed. Morgan Kaufmann, 2010 (cit. on pp. 9, 19).

[Rei+02] Erik Reinhard et al. "Photographic Tone Reproduction for Digital Images". In: *ACM Trans. Graph.* 21.3 (July 2002), pp. 267–276 (cit. on p. 20).

[Rus88] Holly Rushmeier. "Realistic image synthesis for scenes with radiatively participating media". PhD thesis. Cornell University, 1988 (cit. on p. 10).

[S G94] Andrew S. Glassner. “A Model for Fluorescence and Phosphorescence”. In: *Proceedings Fifth Eurographics Workshop on Rendering*. Springer Berlin Heidelberg, June 1994, pp. 57–68 (cit. on p. 21).

[Sch94] Christophe Schlick. “An Inexpensive BRDF Model for Physically-based Rendering”. In: *Computer Graphics Forum* 13.3 (Aug. 1994), pp. 233–246 (cit. on p. 24).

[Sto52] George Gabriel Stokes. “XXX. On the change of refrangibility of light”. In: *Philosophical Transactions of the Royal Society of London* 142 (1852), pp. 463–562 (cit. on p. 20).

[Whi80] Turner Whitted. “An Improved Illumination Model for Shaded Display”. In: *Commun. ACM* 23.6 (June 1980), pp. 343–349 (cit. on pp. 15, 24).

[Wil+06] Alexander Wilkie et al. “A Reflectance Model for Diffuse Fluorescent Surfaces”. In: *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques*. GRAPHITE ’06. ACM, 2006, pp. 321–331 (cit. on p. 22).

[ZSS14] Yinqiang Zheng, Imari Sato, and Yoichi Sato. “Spectra Estimation of Fluorescent and Reflective Scenes by Using Ordinary Illuminants”. In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 188–202 (cit. on p. 31).