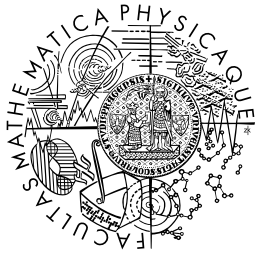


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Marcel Hruška

A Methodical Approach to the Evaluation of Appearance Computations

Department of Software and Computer Science Education

Supervisor of the master thesis: doc. Alexander Wilkie, Dr.

Study programme: Computer Science

Study branch: Computer Graphics and Game Development

Prague 2020

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: A Methodical Approach to the Evaluation of Appearance Computations

Author: Bc. Marcel Hruška

Department: Department of Software and Computer Science Education

Supervisor: doc. Alexander Wilkie, Dr., Department of Software and Computer Science Education

Abstract: Abstract.

Keywords: key words

Contents

Introduction

Goals

Thesis organization

1. Rendering and Color Science

This chapter serves as an introduction to the computer graphics and the color science. We briefly overview basic aspects of these fields, mainly to familiarize the reader with some of the fundamental processes, their backgrounds and usages. We also establish the terminology, such as *rendering* or *RGB color space*, that will be used throughout the thesis frequently. A significant part of the following sections is based on Wyszecki and Stiles [17], Wilkie [15], Nimier-David et al. [10] and Pharr et al. [12].

First, we discuss the mechanisms behind the light and colors and then we look into the process of the physically based image reproduction.

1.1 Light

According to the definition by Barbrow [2], the light is "any radiation capable of causing a visual sensation directly". In other words, the visible light is an electromagnetic radiation that is perceivable by the human eye and allows us to see the objects around us.

As all electromagnetic radiations, the light also propagates in form of waves. The oscillation direction of these waves does not change the color of the light but it may interact differently with the reflective/refractive objects as it passes through them. A representation of such wave propagation is displayed in Figure 1.1.

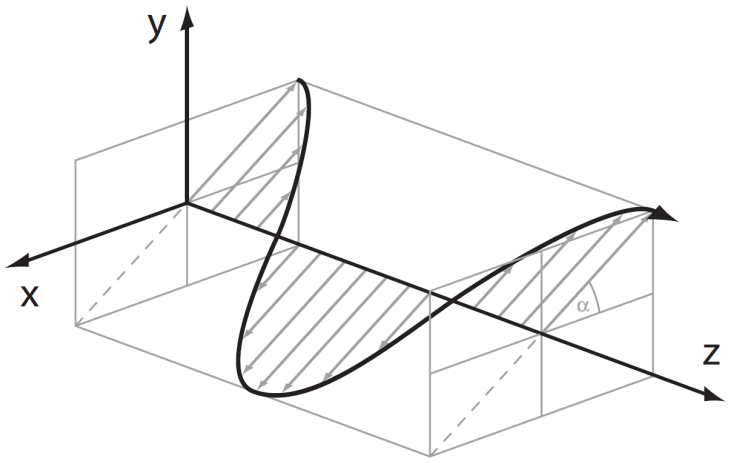


Figure 1.1: A propagation of wave [15]

Normally, by the term light we mean the *visible light* (also called the *white light*) which consists of multiple waves of unique frequencies (wavelengths). There are no exact boundaries to the visible spectrum as distinct human eyes might perceive light slightly differently but the lower boundary is estimated between 360 and 400nm and the upper boundary from 760 to 830nm [14]. Above this range there is the infrared light and below the ultraviolet. An explanatory image of the known electromagnetic wavelengths can be found in Figure 1.2.

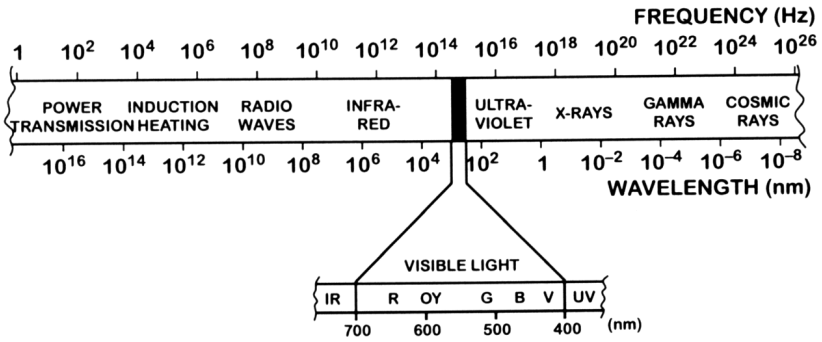


Figure 1.2: An image displaying various wavelengths [15]

1.1.1 Color

While we observe an illuminated object, three different signals are sent from the eye sensors (rods and cones) to the brain, each representing a red, a green or a blue wavelength. When put together inside the brain, they form a sensation of the final color.

To categorize the colors, we formed several reproducible representations of them called the *color spaces*. A natural decision was to create an *RGB color space* as it directly correlates with the signals sent from the human eye's rods and cones. Note that multiple variations of the RGB color space exist such as *sRGB* or *Adobe RGB*. An illustrative comparison between several color gamuts is shown in Figure 1.3.

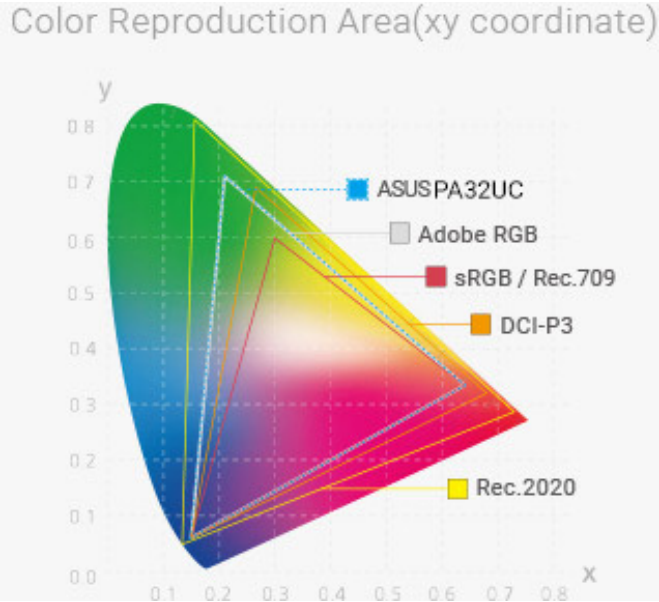


Figure 1.3: An illustrative comparison between five RGB gamuts by Asus [1]

In 1913, the Commission internationale de l'éclairage (International Commission on Illumination), shortly CIE, was formed as an authority that defines almost everything that concerns colors and their perception. In 1931, they conducted

[1] https://www.asus.com/Microsite/ProArtMonitor/experience-truecolor.html

the color matching experiments to obtain the three color matching functions that would convert the color stimuli perceived in our eyes to the *CIE RGB* color space. As these functions had a negative component, a new imaginary color space was created called *CIE XYZ*. These conversions are further described in subsection 1.3.2.

CIE also defined *CIE L*a*b** color space, standard illuminants D65 and D50 and many others.

1.1.2 Photometry and Radiometry

Two different science fields were developed to quantify the light – the *photometry* and the *radiometry*. The radiometry recognizes the light as an electromagnetic radiation while the photometry focuses more on the human perception of the light. Despite the distinct purposes, their quantities are often easily convertible.

Radiometric Quantity	**Photometric Equivalent**
Spectral radiant energy $[J]$	Luminous energy $[Lumen - second]$
Radiant flux $[W]$	Luminous flux $[Lumen]$
Irradiance $[W.m^{-2}]$	Illuminance $[Lumen.m^{-2}]$
Radiant intensity $[W.sr^{-1}]$	Luminous intensity $[candela = Lumen.sr^{-1}]$
Radiance $[W.sr^{-1}.m^{-2}]$	Luminance $[candela.m^{-2}]$

A brief description of each of them:

Spectral radiant energy Amount of the light energy at some wavelength

Radiant flux Amount of the light energy with respect to time

Irradiance Flux at a specific point (space)

Radiant intensity Flux in a direction (sr (steradian) is a unit of the solid angle — surface on a unit sphere, whole sphere has 4π steradians)

Radiance Spatial and directional flux

The relationship between these quantities is described by the *spectral efficiency function*. It states how efficiently a human eye reacts to different wavelengths, i.e. we can detect some wavelengths more easily than others. As we can see in Figure 1.4, the scotopic (night) perception peaks at around 507nm and the photopic (day) at 555nm.

1.2 Physically based rendering

One of the ultimate goals of the computer graphics is the ability to reproduce visually plausible and physically coherent images that should be indistinguishable from a photograph based on a description of a scene. Such process is called the *photo realistic rendering*. In this thesis, we abbreviate the term and call it simply the *rendering* as the non-photo realistic one does not concern us.

Depending on the implementation, the renderer simulates various phenomena commonly seen in nature such as light reflections, refractions, shadows, etc.

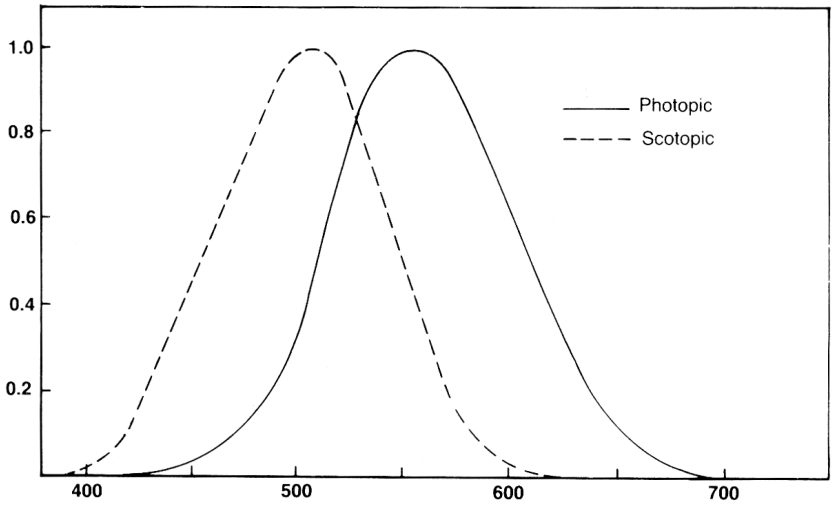


Figure 1.4: Relative luminous efficiency function [15]

Providing a powerful hardware, modern renderers adapt various physical models (or their approximations) of light transport or material properties to provide accurate photo realistic results. In reality, the renderers are so capable that the rendered images are almost identical to the real life photos. An example can be seen in Figure 1.5.

Figure 1.5: An image generated with the Corona Renderer[2]

The main idea is similar for every renderer:

1. A 3D digital scene is described by the objects it contains
2. A light simulation algorithm runs for every visible pixel from the viewer

[2] `https://corona-renderer.com/gallery`

3. Upon object interaction, the shading of the intersected point is computed
4. As the algorithm terminates, a picture ("photograph") of the scene called the *render* is created

Please see figure Figure 1.6 for a simple demonstration of this workflow.

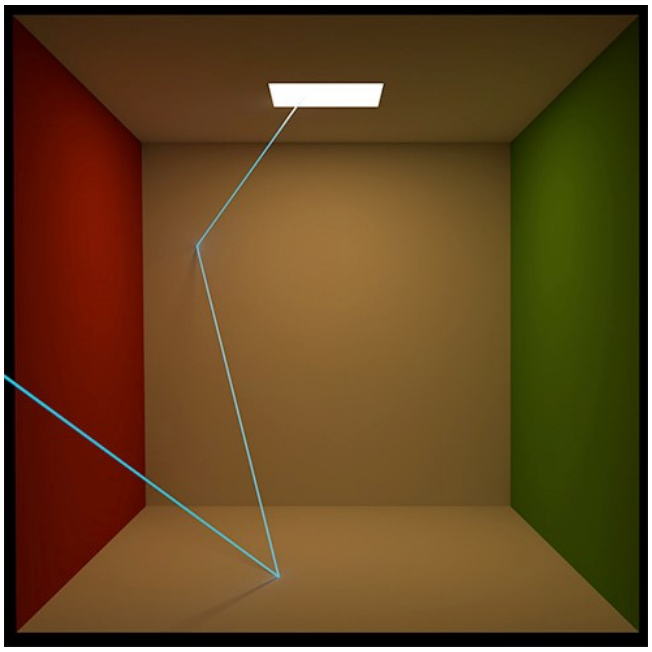


Figure 1.6: A visualization of a light transport algorithm (path tracer) [1]

1.2.1 Digital scene

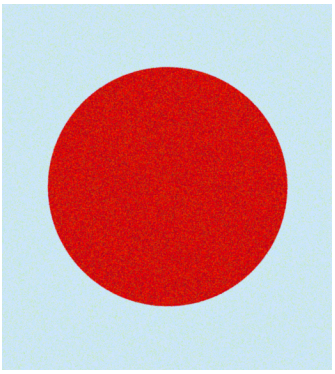
Basic elements of a digital scene are roughly the same for each renderer.

Camera A camera (or a sensor) in a digital scene works in the same manner as in real life — it records a picture. Generally, you may define the coordinate position and the viewing vectors but also the properties such as focal distance or the type of the film.

Light source The scene needs to be illuminated by one or multiple sources in order to be visible. The common kinds of lights are point light, area light, spot light or environment (constant) lighting.

Objects The actual visible contents of the scene are objects. Almost all rendering systems offer a choice to either use their precomputed basic geometry such as spheres or triangles or to include a mesh geometry described in an external file (usually created by an external modeling software). These objects must state their material properties so that the algorithm may correctly interact with them, e.g. diffuse vs. reflective material.

Unfortunately, as each renderer may have a very unique implementation details, the formats of the scenes are vastly different. For example, mitsuba uses XML but PBRT has its own specific format. An example of a simple scene for Mitsuba2 can be found in Figure 1.7.



```
<scene version="2.0.0">
 <!-- Light transport algorithm -->
 <integrator type="path"/>

 <!-- Camera looking at the sphere -->
 <sensor type="perspective">
  <transform name="to_world">
   <lookat origin="0,-6,0" target="0,0,0" up="0,0,1"/>
  </transform>
 </sensor>

 <!-- Red sphere in the middle -->
 <shape type="sphere">
  <bsdf type="diffuse">
   <rgb name="reflectance" value="1.0,0.0,0.0"/>
  </bsdf>
 </shape>

 <!-- Light blue light all around the scene-->
 <emitter type="constant">
  <rgb name="radiance" value="0.6,0.8,0.9"/>
 </emitter>
</scene>
```

Figure 1.7: A simple scene rendered with Mitsuba2 (left) along with its scene description (right)

1.2.2 BRDF

The fundamental part of the rendering process is its implementation of the light transport simulation. This and the following sections will describe the physics theory and the models behind the light transport. Then we take a look at the specific algorithms.

The materials are described by *Bidirection Distrubtion Reflectance Function*, shortly *BRDF* [9]. It looks as follows:

$$f_r(\omega_i, \omega_o) = \frac{dL_o(\omega_o)}{L_i(\omega_i) cos\theta_i d\omega_i} \tag{1.1}$$

This function is given the incoming vector ω_i, the outgoing vector ω_o and it states how much radiance is reflected from the direction ω_i ($L_i(\omega_i)$) to the direction ω_o ($L_o(\omega_o)$) for the specific material. An image interpretation of the function is in Figure 1.8. As it is a distribution function, we can also reformulate it's meaning as a probability density that a defined amount of light energy gets reflected from ω_i to ω_o.

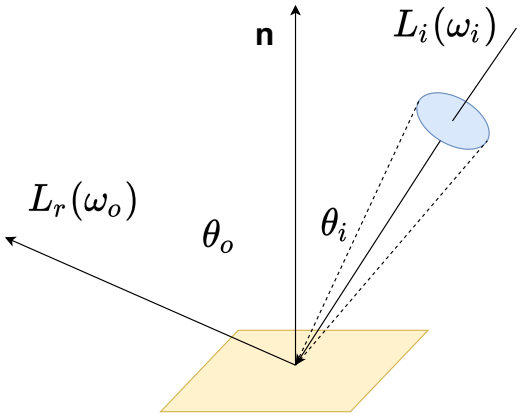


Figure 1.8: Bidirectional Distribution Reflectance Function

In the simplest cases, the BRDF states how reflective the surface of an object

is. The renders of diffuse, rough glossy and mirror materials are compared in Figure 1.9.

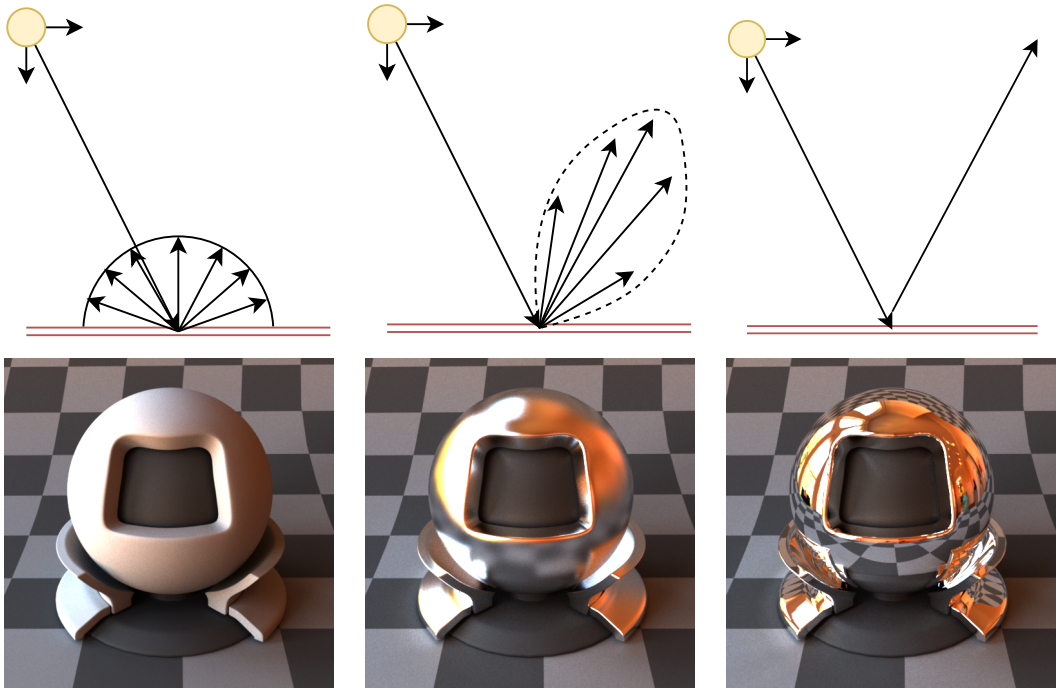


Figure 1.9: A preview of diffuse (left), glossy (middle) and mirror (right) materials rendered in Mitsuba2 along with their illustrative BRDF visualizations

Physically based BRDFs must fulfill several properties 4:

Heimholtz reciprocity The amount of reflected energy from the incoming direction to the outgoing direction is equal to the amount of energy in the reversed directions ($f_r(\omega_i, \omega_o) = f_r(\omega_o, \omega_i)$).

Energy conservation The amount of reflected energy cannot be larger than all received energy.

Positivity BRDF is always positive ($f_r(\omega_i, \omega_o) \geq 0$).

Note that BRDF concerns only opaque surfaces. There exist multiple distribution functions that describe behavior of other materials, for example:

BTDF Describes light transmission

BSDF Combination of BTDF and BRDF (e.g. glass, water)

BSSRDF Considers scattering of the light under the surface as well (skin)

1.2.3 Global Illumination

With the BRDF defined, we can now formulate an equation that evaluates the global illumination of a scene — illumination of each point from all light sources. It is generally called the *rendering equation* [8]:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + L_r(x, \omega_o) \tag{1.2}$$

Let's break it down first:

x is the currently computed point in the scene.

L_o is the outgoing radiance.

L_e is the emitted radiance of the point x as x can be on a light source.

L_r is also called the *reflectance equation* and it states the total amount of the reflected radiance for all contributions of the incident radiance. Hence, it is an integral over the upper hemisphere over x that look as follows:

$$L_r(x, \omega_0) = \int_\Omega f_r(x, \omega_o, \omega_i) L_i(x, \omega, i) cos\theta_i d\omega_i \tag{1.3}$$

, where

$f_r(x, \omega_o, \omega_i)$ is the BRDF of x as defined in Equation 1.1.

$L_i(x, \omega_i)$ is the incoming radiance from a light source.

An image interpretation of the reflectance equation can be seen in Figure 1.10.

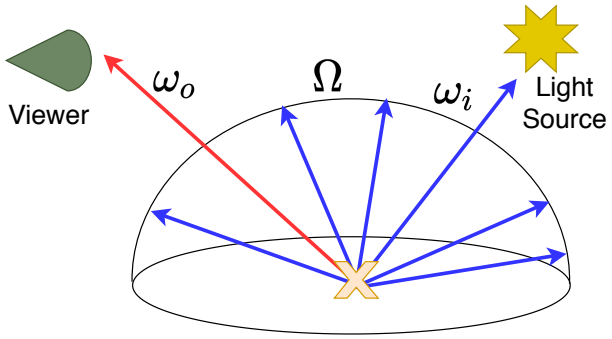


Figure 1.10: Reflectance Equation

As a matter of fact, each light transport algorithm tries to solve some of the formulations of the rendering equation.

Interestingly, the light transport is recursive in nature. As we can see from the rendering equation, to compute the outgoing radiance at a certain point x, we need to know all the contributed incoming radiances. These do not necessarily have to originate at a light source — the incoming radiance may come from another, non-emitting point y in the scene as a result of the rendering equation computed at the point y.

1.2.4 Monte Carlo integration

Before we proceed to the actual algorithms that evaluate the rendering equation, we briefly introduce a method that is used to approximate the definite integral part of the equation — *Monte Carlo integration* [3].

Formally, for a multidimensional definite integral

$$I = \int_\Omega g(x) dx \tag{1.4}$$

Monte Carlo (MC) estimates I as

$$\langle I \rangle = \frac{1}{N} \sum_{k=1}^{N} \frac{g(\xi_k)}{p(\xi_k)}; \xi_k \propto p(x) \tag{1.5}$$

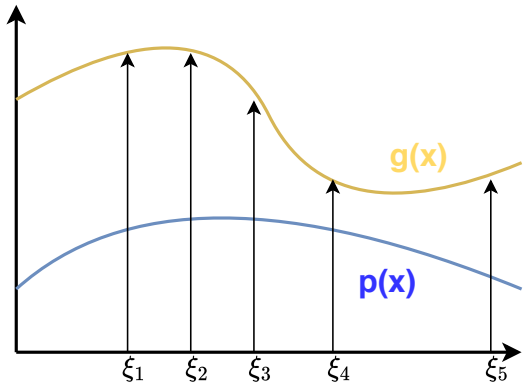


Figure 1.11: Monte Carlo method 2D visualization

In other words, Monte Carlo is a non-deterministic method that sums N randomly chosen samples ξ_k, computes their values $g(\xi_k)$ and averages them. To reduce variance, an importance sampling is introduced by drawing samples from a distribution $p(x)$ that is chosen for each specific problem to approximate the former $g(x)$ function. In reality, the importance sampling ensures that if we pick some samples twice as much, we decrease their weight to half.

There exist other methods that are used to approximate integrals such as deterministic quadrature or Markov Chain Monte Carlo (MCMC).

1.2.5 Light transport algorithms

Path tracing

Over the years, a large number of various light transport algorithms and their variations have been developed, where each has it's own benefits. The one that we will mention the most in this thesis is called the *path tracing*. It's core idea is simple:

1. For each pixel in the image plane, shoot a primary ray r from camera into the scene.
2. If r hits a non-emitting object at point x:
   - 2.1. Compute BRDF at x.
   - 2.2. Generate a new random direction ω. Ideally, the distribution of the generated direction should be proportional to the BRDF — e.g. diffuse BRDF would generate a direction uniformly over a hemisphere while glossy BRDF would prioritize samples from the reflectance lobe (look at Figure 1.9).
   - 2.3. Add the BRDF value to the final color of the pixel.

2.4. Check for a terminating condition — there exist several options, usually a combination of them is applied:

Maximum depth User specified maximum number of recursions.

Russian Roulette Randomly choose if the ray survives, with each consecutive ray the chance lowers.

BRDF-proportional Depending on the surface material, we decide whether the ray survives or not. For example, reflective or refractive surfaces need a lot more recursions as they propagate the light further to the scene than diffuse surfaces.

2.5. In case the termination was not successful, bounce – shoot a secondary ray r fromthe point x in the direction ω and continue from step 2.

3. If r hits an emitting object (light source), add it's emission L_e to the final color of the pixel

4. If no scene geometry is hit, terminate the algorithm and add the color of the surrounding light (if there is any).

The bouncing of the light in the scene nicely correlates with the recursive nature of the rendering equation. Even though the path tracing is a slow algorithm (e.g. not suitable for real-time rendering in games), it's variations can be extremely accurate, even indistinguishable from a real photograph.

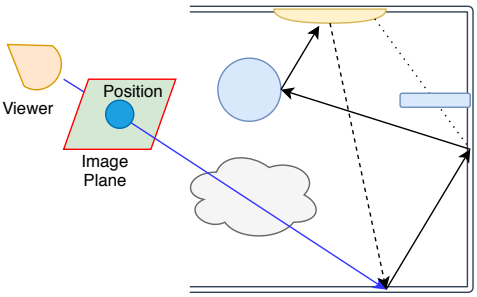
MIS In the algorithm described above, the direct illumination computation of each scene intersection is dependent only on the BRDF of the intersected surface and consequent walk to the light source. However, such scenario that the light source is hit at the end of every walk is greatly dependent on the number of samples and the maximum allowed depth of the recursion. Consequently, this creates variance which can be easily improved the integration of the *multiple importance sampling* (MIS). Generally, it involves a combination of multiple sampling techniques but in our case it is a combination of the BRDF proportional sampling and the light source sampling — in each step of the path tracing, every light source that is visible from the intersected point contributes to it's value. Both sampling methods are, of course, weighted to avoid an over-illumination.

Volumes Another aspect that needs to be accounted for in the rendering process are volumetric objects such as fogs or smokes. Normally, there are two ways that a volumetric object may effect the light passing through it. The volume is either attenuating the light by absorbing it or scattering to different directions. Or the volume can also strengthen it by emitting light (e.g. flame) or scattering light from different directions to the current one.

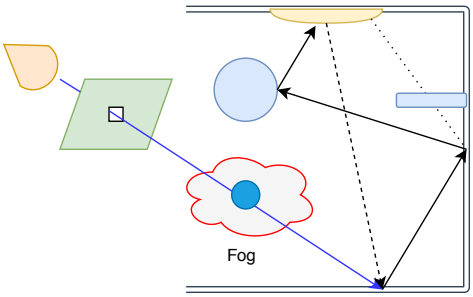
A single walk of a path tracer capable of volume tracing and MIS sampling is visualized in Figure 1.12.

Other methods

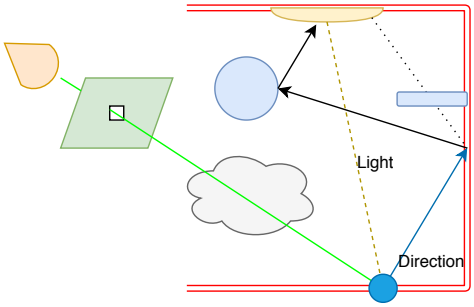
As the path tracing is of our main concern in this thesis due to its physically based and unbiased properties, some of the other global illumination techniques are described only briefly here:



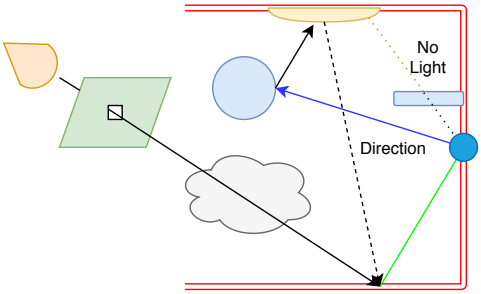
(1) Primary ray created



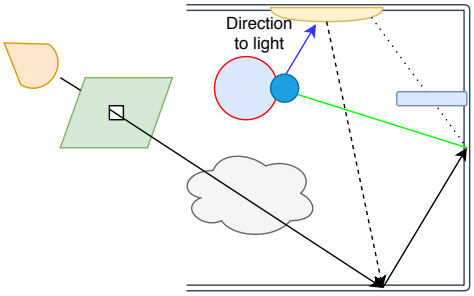
(2) Passing volume, attenuation



(3) Geometry hit, new direction, BRDF + light added



(4) Geometry hit, new direction, BRDF added (light obscured)



(5) Geometry hit, new direction points to light, add contribution and terminate

Figure 1.12: A visualization of a single walk in path tracer

Ray tracing [5] Similar to the path tracing but there are no bounces from the surfaces, simulates only reflections, refractions, scattering etc. Capable of realtime rendering these days.

Photon mapping [7] Two rays are traced independently — from the camera and from a light source until termination, then the radiance is computed based on their final positions. Faster at some scenarios but biased (does not have to converge to a correct solution).

Radiosity [13]Uses the finite element method instead of Monte Carlo. View independent, the light is traced from the source and bounced (possibly) to the viewer. Good for precomputations.

1.3 Spectral Rendering

So far, we've considered the colors to be internally represented by a tristimulus color space during the rendering process. For the explanation purposes, let's consider it to be RGB color space – the objects are defined in RGB, the path tracing step colors are RGB and the output color is RGB. In a large number of scenarios, this workflow is sufficient as we are capable of simulating a majority of the common aspects (e.g. optics) while keeping the rendering simple and robust. Unfortunately, the RGB color space is only a fraction of the visible gamut and does not contain any information about the light as an electromagnetic radiation. Consequently, we are loosing a significant amount of information, causing the colors to be at times inaccurate and some phenomena completely impossible to render.

Therefore, a new approach to the rendering has been introduced that internally represents the colors as a spectrum distribution function instead of a tristimulus color space — the *spectral rendering*. The core idea is to track and sample several wavelengths at once for each step of the path tracing and to perform the integration all over them.

Most of the phenomena evaluated in this thesis are a direct consequence of the light's nature as an electromagnetic radiation, hence the primary focus is placed on the spectral rendering. The following sections are largely based on Wilkie and Purgathofer [16].

1.3.1 Color representation

If the spectral rendering is desired, there is a need to store the spectral distribution function. While several techniques are feasible, it is often a matter of a simple trade-off between the precision and the performance. For example, sampling wavelengths uniformly each 10nm would yield significantly more accurate results than sampling only four wavelengths in total. However, such approach might become unbearable in terms of speed and memory. Moreover, the spectral functions are mostly quite smooth. Some examples of these functions can be seen in Figure 1.13.

A typical approach is to sample at larger ranges (10 or more nanometers) and combine it with the basis functions [11]. More approaches and their details are briefly explained in Wilkie and Purgathofer [16].

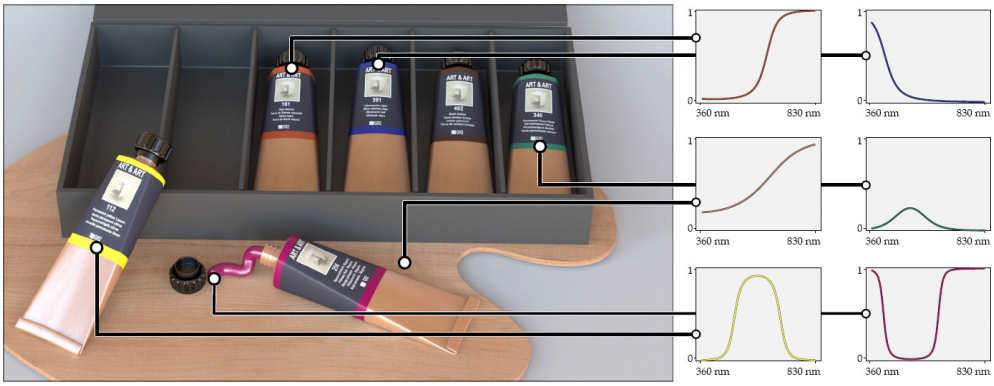


Figure 1.13: Spectral curves measured for different colors[6].

1.3.2 Conversions to tristimulus color spaces

The conversion from the spectrum to a RGB color space:

1. Compute the tristumulus value XYZ using the CIE color matching functions (shown in Figure 1.14)

$$X = \int P(\lambda)\overline{x}(\lambda)d\lambda$$
$$Y = \int P(\lambda)\overline{y}(\lambda)d\lambda$$
$$Z = \int P(\lambda)\overline{z}(\lambda)d\lambda$$

   , where $P(\lambda)$ is the spectral power distribution and $\overline{x}(\lambda)$, $\overline{y}(\lambda)$ and $\overline{z}(\lambda)$ are the color matching functions.

2. Convert the XYZ to the desired RGB color space using a transformation matrix. Depending on the specific RGB color space, the matrix differs — an example of CIE XYZ to sRGB conversion is demonstrated in item 2.

$$r = 3.240X - 0.969Y + 0.55Z$$
$$g = -1.537X + 1.875Y - 0.204Z$$
$$b = -0.498 + 0.041Y + 1.057Z$$

3. (Optional) As the resulting r,g,b values may be negative, a gamut mapping might be necessary.

1.3.3 Advantages

The spectral rendering presents the ability to reproduce the colors in a more photorealistic way. We might not see it at first, but the colors produced by a conventional RGB renderer tend to be slightly over-saturated as they do not account for the spectral characteristics which might attenuate the final color.

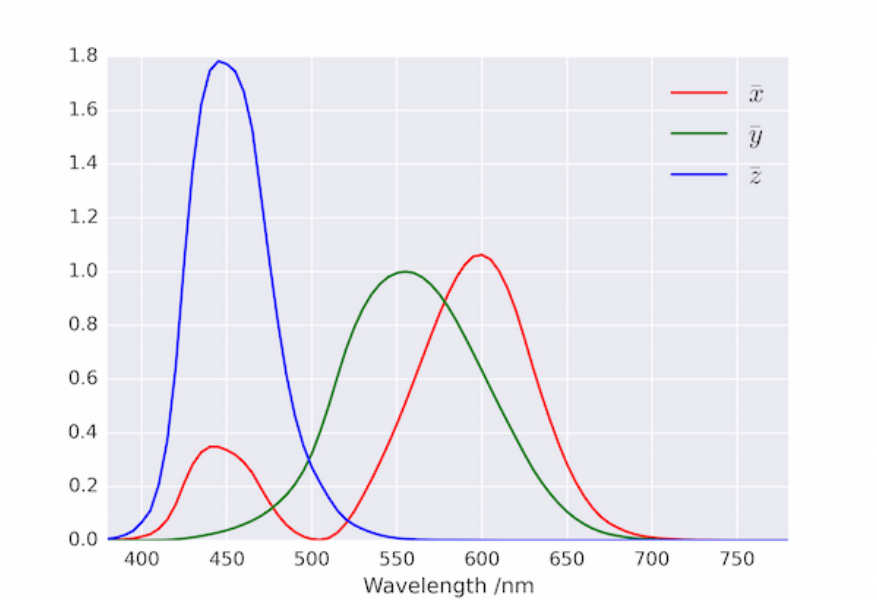


Figure 1.14: Color matching functions plotted in Python[3]

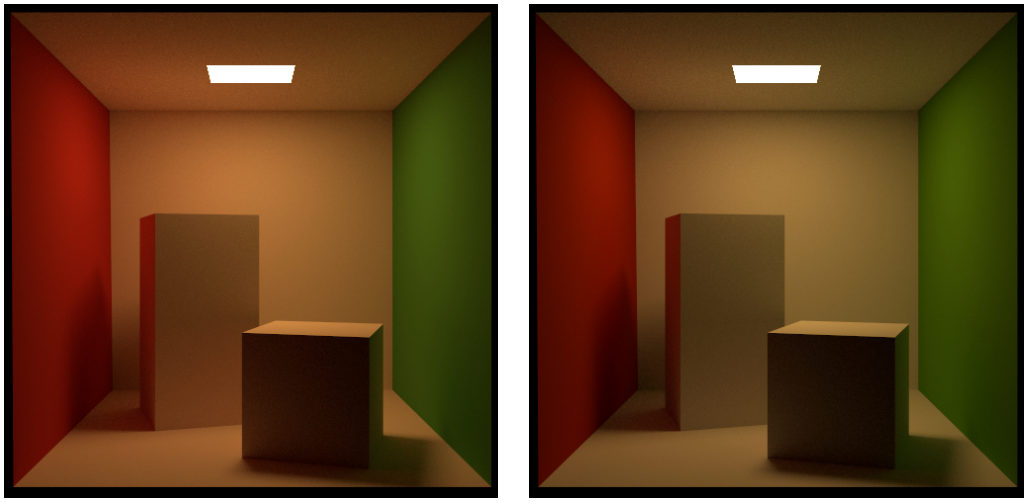


Figure 1.15: A comparison between the RGB render (left) and the Spectral render (right) of the same scene by Mitsuba2.

A comparison between the RGB and the spectral render of the same scene by Mitsuba2 is shown in Figure 1.15.

Still, the biggest improvement is the possibility to reproduce some of the natural phenomena for which the tracing of multiple wavelengths is an absolute necessity. Namely, those are:

Fluorescence Absorption and re-emission of a different color

Dispersion Splitting of the white light into its wavelength components via refraction

Polarisation Change of the oscillation direction of a light wave

Iridescence Thin layer constructive/destructive interference

They are explained in detail in chapter 2.

1.3.4 Disadvantages

On the other side, the need to numerically integrate over multiple wavelengths introduces a problem of the chromatic noise. Even though the performance is not a key factor, it obviously worsens as well.

Moreover, we are still unable to properly display spectral images on the current monitors. The only viable way is to store the distinct spectral bands as separate images. Clearly, it is quite inconvenient to have multiple results instead of one. Therefore, the rendering needs to have a well-done spectrum to final RGB conversion as the final image is still being displayed on an RGB monitor. Fortunately, the conversion is well-defined and fairly simple to implement.

Due to the fact that the RGB gamut is only a subset of the visible spectrum, the situation gets significantly more complicated for the reversed conversion. As there exist infinitely many spectra for one RGB value, several techniques were proposed to convert the tristimulus to the spectral domain — commonly called the *spectral upsampling*. For those who may be interested in the details of the current development to the spectral upsampling, refer to the article by Jakob and Hanika [6] which proposes a solution that is capable of converting full sRGB gamut with zero error.

There exist several reasons to integrate the spectral upsampling, mainly because the spectral values are a lot harder to obtain and to use. You need a specific device (spectrometer) that would measure the color values under a specific light and then use regularly distributed samples from it as an input. Because of the reproducibility of the RGB color space and its legacy usage (lots of existing textures are already defined in RGB), it is a lot more convenient to input the values of textures as an RGB value, convert them internally to the spectral domain and convert them to the desired color space for the output image.

[3] `https://scipython.com/blog/converting-a-spectrum-to-a-colour/`

2. Appearance Computations

2.1 Reflectance

2.2 Spectral accuracy

2.3 Polarization

2.4 Fluorescence

2.5 Iridescence

2.6 Dispersion

3. Benchmark

3.1 Framework

3.2 Common Data

3.3 Supported Spectral Renderers

3.3.1 Mitsuba2

3.3.2 ART

3.4 Scenes

3.5 Usage

3.6 Future Work

Conclusion

Bibliography

[1] Mitsuba2 docs. `https://mitsuba2.readthedocs.io/en/latest/`. Accessed: 2020-6-7.

[2] LE Barbrow. International lighting vocabulary. *Journal of the SMPTE*, 73 (4):331–332, 1964.

[3] Russel E Caflisch et al. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 1998:1–49, 1998.

[4] Bernardt Duvenhage, Kadi Bouatouch, and Derrick G Kourie. Numerical verification of bidirectional reflectance distribution functions for physical plausibility. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, pages 200–208, 2013.

[5] Andrew S Glassner. *An introduction to ray tracing*. Elsevier, 1989.

[6] Wenzel Jakob and Johannes Hanika. A low-dimensional function space for efficient spectral upsampling. In *Computer Graphics Forum*, volume 38, pages 147–155. Wiley Online Library, 2019.

[7] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. AK Peters/CRC Press, 2001.

[8] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[9] Fred E Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied optics*, 4(7):767–775, 1965.

[10] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6):1–17, 2019.

[11] Mark S Peercy. Linear color representations for full speed spectral rendering. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 191–198, 1993.

[12] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[13] Francois X Sillion and Claude Peuch. Radiosity & global illumination. 1994.

[14] DH Sliney. What is light? the visible spectrum and beyond. *Eye*, 30(2): 222–229, 2016.

[15] Wilkie. Cgg course: Introduction to the color science. `https://cgg.mff.cuni.cz/~wilkie/Website/NPGR025.html`. Accessed: 2020-11-7.

[16] Kate Devlin1 Alan Chalmers1 Alexander Wilkie and Werner Purgathofer. Tone reproduction and physically based spectral rendering. Eurographics, 2002.

[17] Gunter Wyszecki and Walter Stanley Stiles. *Color science*, volume 8. Wiley New York, 1982.