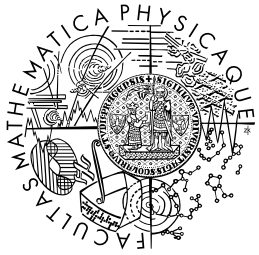


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

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 [8], Nimier-David et al. [6] and Pharr et al. [7].

1.1 Color perception

1.2 Physically based rendering

One of the ultimate goals of the computer graphics is to be capable of reproducing visually plausible and physically coherent images based on a description of a scene that should be indistinguishable from a photograph of the same 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. 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 indistinguishable from the real life photos. An example can be seen in 1.1.

Figure 1.1: An image generated with the Corona Renderer

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
3. Upon object interaction, the shading of the intersected point is computed
4. When it's done, results in form of a picture ("photograph") called *render* are created

Please see figure 1.2 for a simple demonstration of this workflow.

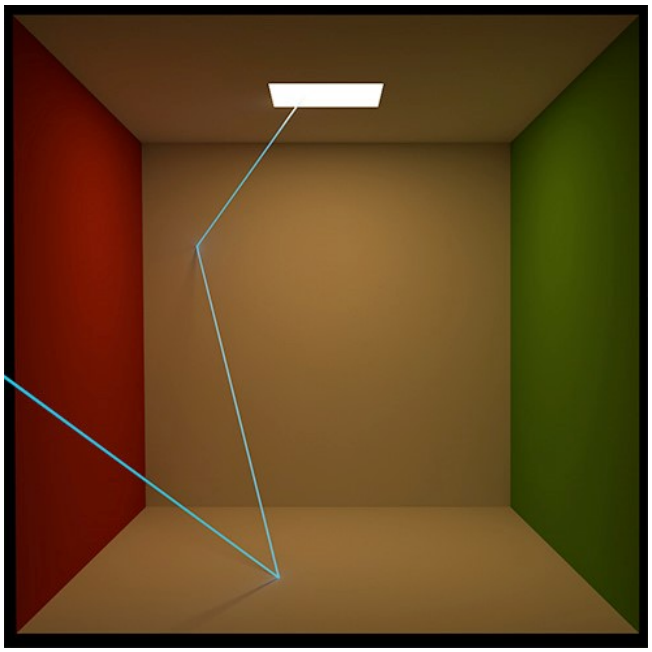


Figure 1.2: A visualization of a light transport algorithm (path tracer) [1]

1.2.1 Digital scene

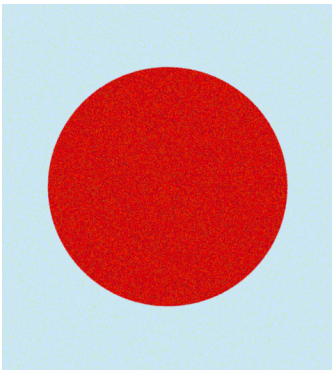
The basic elements of a digital scene are roughly the same for each renderer.

Camera A camera 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 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 content 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 modeling software). They also have to 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 1.3.



```
<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 1.3: 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 of objects inside a scene are described by *Bidirection Distrubtion Reflectance Function*, shortly *BRDF* [5]. 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 tells us how much radiance is reflected from the direction ω_i ($L_i(\omega_i)$) to the direction ω_o ($L_o(\omega_o)$). An image interpretation of the function is in 1.4. As it is a distribution function, we can rather describe it as a probability density that a defined amount of light energy gets reflected from ω_i to ω_o.

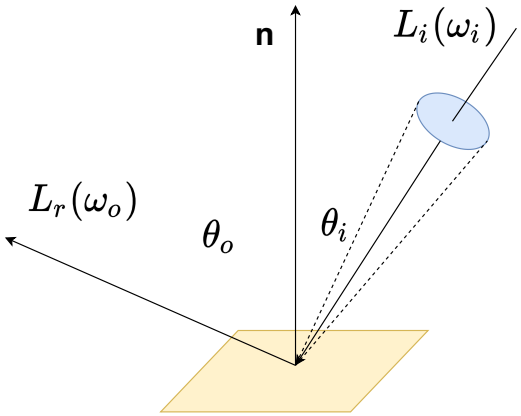


Figure 1.4: Bidirectional Distribution Reflectance Function

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 1.5.

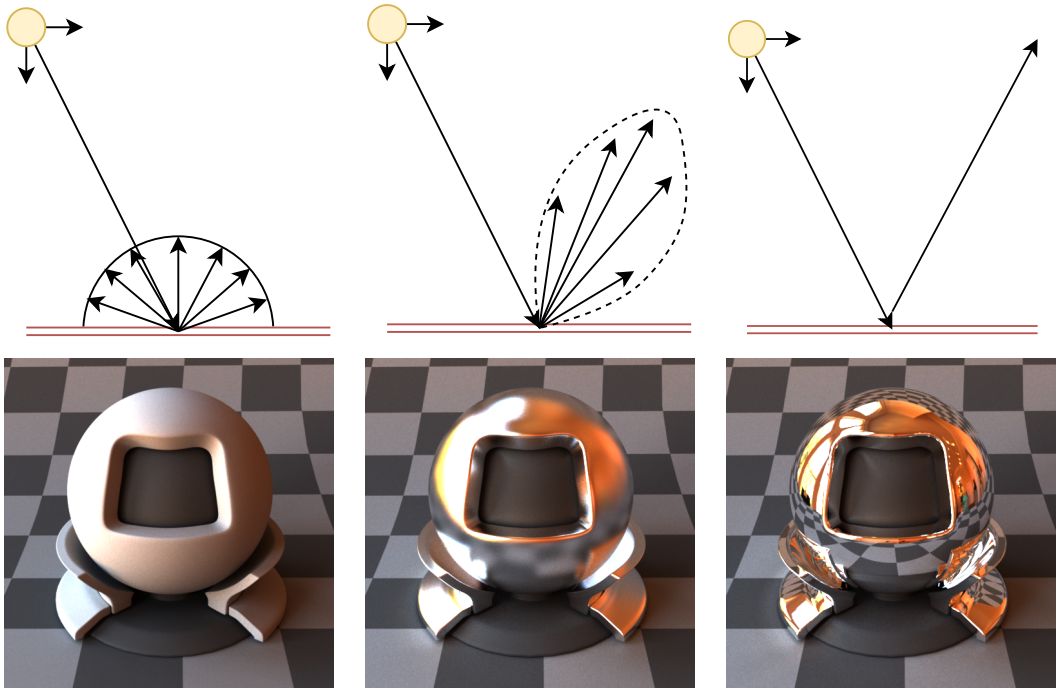


Figure 1.5: A preview of diffuse (left), glossy (middle) and mirror (right) materials rendered in Mitsuba2 along with their illustrative BRDF visualizations

Physically based BRDFs must fulfill several properties 3:

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 received

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

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 describes the global illumination of the scene — illumination of each point from all light sources. It is generally called the *rendering equation* [4]:

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

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 the 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 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 1.6.

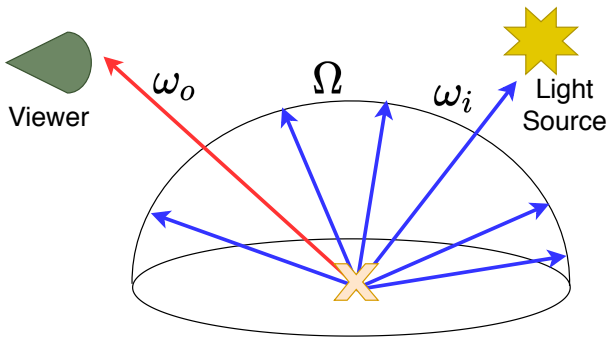


Figure 1.6: Reflectance Equation

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 for 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* [2].

Formally, for a multidimensional definite integral

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

Monte Carlo (MC) estimates I as

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

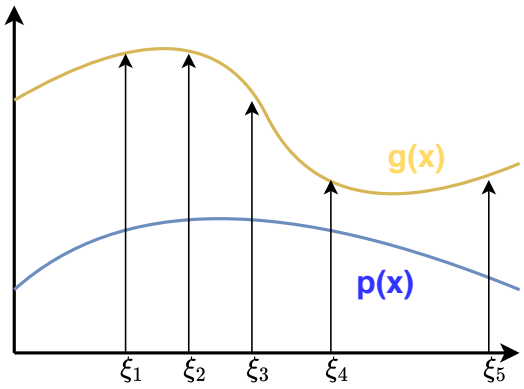


Figure 1.7: Monte Carlo method 2D visualization

In other words, Monte Carlo is a non-deterministic method that sums N randomly chosen samples, computes their values 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. This states 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

There exist various algorithms for the light transport simulation 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. Shoot a primary ray r from camera for each pixel in the image plane 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 1.5).

   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:

   Maximum depth User specified maximum number of recursion.

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

   BRDF-proportional Depending on the surface material, we decide whether the ray survives or not. For example, reflective or refractive surface need a lot more samples than diffuse surfaces.

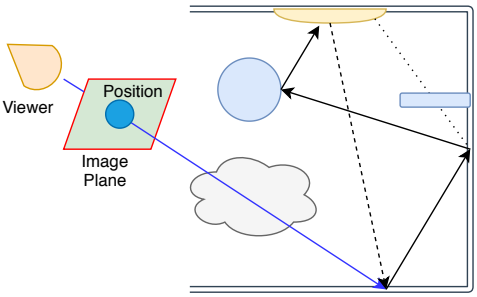
   2.5. In case the termination was not successful, bounce – shot a secondary ray r from point x in the direction ω and continue from step 2.

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

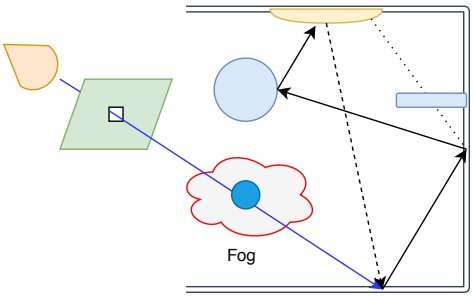
4. If it no scene geometry is hit, terminate the algorithm. In case this is the primary ray, the pixel is the color of a surrounding light.

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.

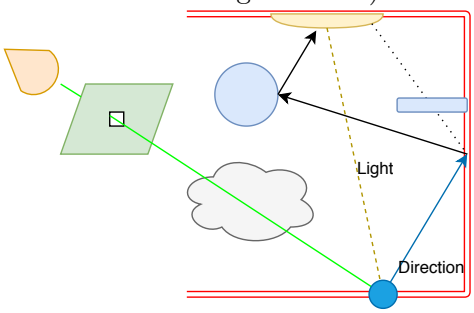
MIS In the algorithm described above, the direct illumination computation is dependent only on the BRDF of each intersected point and consequent walk to the light source. However, such scenario that the light source is hit at the end of every walk is greatly dependent on the number of samples and the maximum allowed depth of the recursion. Consequently, this creates variance which can be easily improved the integration of the *multiple importance sampling* (MIS). This involves 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.



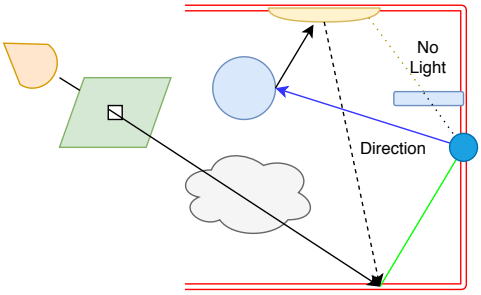
(1) Ray at pixel on image plane created (from blue dot along blue line)



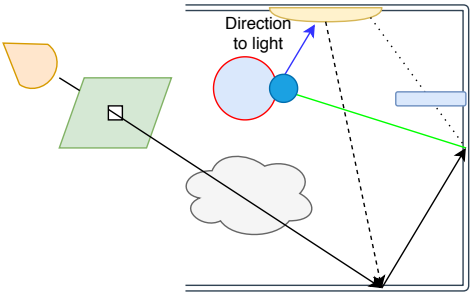
(2) Passing volume, attenuation



(3) Geometry hit, generate new direction, compute BRDF + light contribution



(4) Geometry hit, generate new direction, compute BRDF (no light - obscured)



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

Figure 1.8: A visualization of a single walk in path tracer

Volumes Another aspect that needs to be accounted for in the rendering process generally are volumetric objects such as fogs or smokes. Generally, there are two ways that a volumetric object may effect the light passing through. Is is either attenuating the light by absorbing it or scattering to different directions. Or it 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 1.8.

2. Spectral Rendering

2.1 Color representation

2.2 Rendering process

2.3 Conversions to RGB

2.4 Advantages

2.5 Disadvantages

3. Appearance Computations

3.1 Reflectance

3.2 Spectral accuracy

3.3 Polarization

3.4 Fluorescence

3.5 Iridescence

4. Supported Spectral Renderers

4.1 Mitsuba2

4.2 ART

5. Benchmark

5.1 Usage

5.2 Framework

5.3 Common Data

5.4 Scenes

5.5 Future Work

Conclusion

Bibliography

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

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

[3] 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.

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

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

[6] 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.

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

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