Czech Technical University in Prague
Faculty of Electrical Engineering

Masterer Thesis

Progressive Computation of Global Illumination

Prague, 2012

Author: Bc. Zdeněk Glazer

Aknowledgements

I would like to thank to Hanka Pokorná, who has always supported me. And also to Ing. Jaroslav Sloup for supervision of this thesis.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.

Prague, 10. 11. 2012 _____ _____

Signature

Abstract

Global illumination always played a key role in realistic computer generated image synthesis. Many algorithms have been developed to compute realistic images in reasonable time, though rendering volumetric phenomenas such as smoke including volumetric caustics and complex glossy reflection and refraction paths is still consuming task. Recently some new light caching approaches and progressive rendering techniques were discovered. This thesis aims to test their accuracy and feasibility on various scenes containing heterogenous participating media.

Abstrakt

Globální osvětlení vždy hrálo klíčovou roli při realistické syntéze obrazu. Existuje mnoho algoritmů které dokáží realistický obraz vypočítat v rozumném čase, ale výpočet globálního osvětlení u scén obsahujících opticky aktivní prostředí, jako je například kouř je stále časově velice náročný. V nedávné době se objevily nové postupy předvýpočtu světelných cest ve scéně a nové progresivní metody výpočtu obrazu.V této diplomové práci chceme prověřit jejich přesnost a rychlost na různých scénách obsahujicích nehomogenní opticky aktivní prostřědí.

Vložit zadani prace

Contents

List of Figures

List of Tables

Chapter 1

Introduction

People have always been curious about new forms of visualization and illusion of reality. In the last decade computer generated imagery has reached a state, when majority of the public can't distinguish real photographs from entirely synthesized images. Thanks to this advancements we can enjoy otherwise impossible film shoots, imaginary worlds, visualizations of non existing buildings and much more. All this become possible, because ways how to represent objects and simulate light transport using computer algorithms have been found. The other important factor is ever growing computation power of these devices, which enables us visualize highly detailed datasets with complex materials and challenging lighting scenarios.

Even though many things are possible to simulate and visualize efficiently these days volumetric phenomenons such as clouds, smoke and other optically active medias are still real challenge even for high-end computers. This is all caused by the fact that the data we try to visualized are volumetric and that light passing through can be absorbed, scattered or even emitted in different wavelength (fig 1.1). This means that we can no longer assume that visibility of an object can be determined simply by determining if it is hidden behind other object or not.

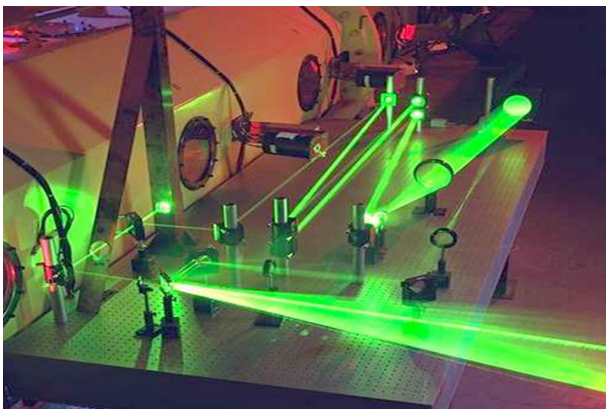

Figure 1.3: Laser beam being refracted and reflected multiple times. The beam is visible thanks to interaction with tiny particles floating in the air, which diffract part of the light energy to the camera chip.Source [NAS].

Figure 1.1: Aurora borealis one of the most fascinating volumetric phenomenons on earth. This photo also exhibits atmospheric light scattering and city glow at night. Source [SAL].

Figure 1.2: Crepuscular rays also known as good rays are nice example of volumetric shadowing effect, which can be quite believably approximated using only single light scattering. Autor [PN].

In terms of feasible render times only direct illumination component can be computed. This approach can give us satisfying results such as those on the figure (fig 1.2). Unfortunately many cool looking effects such as volumetric caustics, which can be seen on the figure (fig 1.3) are impossible to get using this approach.

Figure 1.4: Image after first progressive iteration on the left. After ten iterations in the middle and on the right final quality image after twenty iterations.

This thesis focuses on methods, which can render anything with physical accuracy and without restrictions on the rendered scene. The other very important bonus of these methods is their progressive nature. In the first passes the corse illumination of the scene is computed and it's progressively refined (fig 1.4). This for example means, that artist defining lighting mood of the visual effect shots can iterate faster. They have got something to show to the director, which results in quick turnarounds and substantial cost and time savings.

Our aim is to implement these progressive Monte Carlo methods in the presence of heterogenous, optically active media and test them on scenes with varying complexity.

1.1 Thesis structure

This Thesis consists of eight main chapters. In the first chapter we have introduced you to the problem. In the second chapter the nifty theoretical details behind realistic image synthesis are exposed and fundamental terms such as BRDF, phase function and rendering equation defined and explained.

The third chapter is an overview of current strategies and algorithms used to visualize the volumetric phenomenons and their pros and cons listed. The fourth chapter leads us to the core of this thesis, the progressive methods of computation the global illumination in the presence of volumetrically active phenomenons.

In the fifth chapter we choose some of the methods mentioned in preceding chapter and further analyze them and propose a way how to and why to use them. In the sixth chapter we present our solution and describe it's internal structure. In the seventh chapter our results and test are shown. And finally eight chapter concludes the information gained and results obtained during this thesis creation.

Chapter 2

Fundamentals of realistic image synthesis

In this chapter we briefly revise fundamental theory behind realistic computer generated imagery. Our aim is to synthesize (render) image of a virtual scene. To do this properly, with physical accuracy in mind, we have to find a suitable physical model of light.

If we would like to be absolutely correct, we would have to use quantum optics model. This would lead us to tracing individual photons and interaction with matter on an atomic level, which is completely computationally unfeasible. Nevertheless when it comes to computer graphics even a simplified ray optics model can handle majority of the phenomena we are interested in. This model assumes that light travels in straight lines, at infinite speed and the only things that might happen to it are emission, absorption, reflection, and transmission. Using this model we will be unable to simulate phenomena such as diffraction (for example chromatic aberration), polarization and other similar effects depending on light wavelengths and polarity.

When the suitable light model is chosen, we should define some numerical quantities we can compute with. In the next section we will start with radiometry quantities and inspect problem of light interaction with surfaces. This gives us enough fundaments to formulate rendering equation. We continue with light interaction with participating media, which leads us to extended rendering equation. Algorithms mentioned in the next chapters are based on this theory and they present various methods solving this complex equation.

2.1 Radiometry

Radiometry is set of techniques for measuring different kinds of electromagnetic radiation such as light. We could have used a photometry, which on the other hand measures the response of a human eye to visible light. This basically corresponds to measurements retrieved using radiometry and weighted with an eye response function. For the purposes of this thesis we will use radiometric quantities, because corresponding photometric quantities can be easily derived.

2.1.1 Radiometric quantities

Radiant flux , also known as radiant power, denoted as ϕ. This quantity corresponds to the light power mentioned on the light bulbs in Watts. The definition is:

$$\phi = \frac{dQ}{dt} \quad [W = J.s^{-1}] \tag{2.1}$$

Where Q [J] denotes radiant energy, which describes how much energy (amount of photons times their energy) is presented in a given area at a point in time. This leads us to flux definition as amount of radiant energy going through a location over time , alternatively speaking how fast is the change of the amount of the photons in a specified location.

Irradiance , also known as flux density, denoted as *E(x).*

$$E(x) = \frac{d\phi(x)}{dA} \quad [W.m^{-2}] \tag{2.2}$$

The irradiance at a point x on the surface S, can be expressed as an amount of incident flux to differential area at a given time.

Radiance is probably the most important quantity. Denoted as $L(x, \omega)$, where x is a point of interest and ω is a differential angle coming in a given direction, also known as solid angle.

$$L(x, \omega) = \frac{d\phi(x)}{cos\theta dA d\omega} \quad [W.m^{-2}.sr^{-1}] \tag{2.3}$$

Radiance expresses how much flux is coming from a differential direction $d\omega$ onto a differential surface dA. The $cos\theta$ factor compensates the shortening amount of irradiance on the surface with growing θ[1], while the level of illumination is maintained. This is the quantity, which has to be gathered for every pixel in the rendered image.

2.1.2 Relationships between quantities

In order to render our image we have to compute radiance coming from the scene through every pixel of the virtual camera sensor. From the table 2.1 you can see that radiance units are Watts per square meter per steradian[2]. According to the units of the remaining radiometry quantities we should be able to derive them using integration.

Radiant energy	Q	[J]
Radiant flux	ϕ	[W]
Irradiance	E	$[W.m^{-2}]$
Radiance	L	$[W.m^{-2}.sr^{-1}]$

Table 2.1: Table summary of the radiometric quantities.

If we integrate incoming radiance over a hemisphere denoted as Ω in a given point x we get equation 2.4. Notice a $-\omega$ in the equation, this is caused by the fact that Ω usually refers to hemisphere consisting of outgoing solid angles. But when computing irradiance we are interested in the incoming radiance.

$$E(x) = \int_{\Omega} L(x, -\vec{\omega}) * cos(\theta) d\vec{\omega} \tag{2.4}$$

To get flux from radiance, we will have to perform two integration steps. We will have to integrate radiance over a hemisphere in every differential surface, which belongs to the surface we are interested in. This is precisely written in the equation 2.5.

$$\phi = \int_{A} \int_{\Omega} L(x, \vec{\omega}) * cos(\theta) d\vec{\omega} dA \tag{2.5}$$

[1]The meaning of θ is $cos(\theta) = (\vec{n}.\vec{\omega})$, where $\vec{n}$ is surface normal at a point of interest (x) and $\vec{\omega}$ is a direction vector we want to compute radiance for.

[2]Steradian is unit of a solid angle.

Figure 2.1: This is an example of the same material (aluminum metal) with different surface texture. From left to right brushed, polished and surface after abrasive blasting. Below each image you can see corresponding BRDF function

2.2 Surface interaction

Using ray optics model, light can interact with object in only four ways. It can be created by the object, which is referred as an emission. On the other way it can be partially or fully absorbed, reflected or refracted in the same time.

As was mentioned in the beginning of this chapter, it's almost impossible to simulate these light interactions in large scenes using light interaction on a molecular level. Currently the most used scene representation is probably an object boundary representation[3]. Unfortunately the microscopic structure of the object plays a huge role in the light interaction as can be seen on the fig 2.1 and would again result in an immense complexity if represented using triangles. To cope with this problem we usually model the corse object shapes using the boundary representation and we call the microscopic behavior "the material behavior". To model the material behavior we use the BRDF function.

[3]Consisting of a finite set of triangles, quadrangles and other n-gons.

2.2.1 BRDF

The *bidirectional reflectance distribution function* describes what is the probability that an incoming light from a given incoming direction will reflect to a given outcoming direction from the given surface. As stated before we can define the BRDF function as:

$$f(\omega_i, x, \omega_o) = \frac{dL_o(x, \omega_o)}{dE(x, \omega_i)} = \frac{L(x, \omega_o)}{L_i(x, \omega_i) * cos(\theta_i) d\omega_i} \tag{2.6}$$

To formulate the BRDF function more precisely we should define it's properties:

1. **Function domain:**
   For a given surface point x the BRDF is four dimensional function. Two dimensions for the incoming direction and two for the outgoing direction.

2. **Value range:**
   BRDF can gain any positive value. For example using constant BRDF functions we can model diffuse materials.

3. **Reciprocity:**
   The BRDF value stays the same if we interchange incoming and outgoing direction. This is a the very fundamental property many global illumination algorithms rely on. Thanks to this behavior we can gather or shoot light energy using the same BRDF functions and all light interactions remain the same.

4. **Linearity:**
   It's a linear function, thus it is not dependent on irradiance from other directions. To get total reflected radiance in the given direction ω_o and given point we get an equation 2.7.
   $$L_o(\vec{\omega}_o) = \int_\Omega L_i(\vec{\omega}_i) * f_r(\vec{\omega}_i, \vec{\omega}_O) * cos(\theta) d\vec{\omega} \tag{2.7}$$

5. **Energy conservation:**
   Total reflected energy to all directions is smaller or equal to the total irradiance in any given point. This ensures that an object can't reflect more light than it receives.

2.2.2 Rendering equation

We have successfully defined the quantities we want to compute and models we want to use for the light surface interaction. What we want to achieve is to distribute the light energy in the scene and compute the energy fraction which reaches our image pixels.

To describe the energetic equilibrial state in the scene[4] we use rendering equation 2.8. As you can see it closely resembles the equation 2.7. The main difference between these two equations is the fact that the unknown radiance [5] is on both sides of the equation and that instead of the local radiance $L(x, \omega_o)$ we want to trace a ray in a direction ω_i to the scene and compute the radiance in closest ray surface intersection $L(r(x, \omega_i), -\omega_i)$.

$$L(x, \omega_o) = \int_\Omega L(r(x, \omega_i), -\omega_i) * f(\omega_i, x, \omega_o) * cos(\theta_i) d\omega_i \tag{2.8}$$

$$\overbrace{L(x, \omega_o)}^{\text{outgoing}} = \underbrace{L_e(x, \omega_o)}_{\text{emited}} + \underbrace{\int_\Omega L(r(x, \omega_i), -\omega_i) * f(\omega_i, x, \omega_o) * cos(\theta_i) d\omega_i}_{\text{reflected}} \tag{2.9}$$

The equation 2.8 can deal with light reflection, refraction and absorption. To consider the light emitting objects in our scene we have to introduce an emitted radiance [6] into the equation. Which result in eq 2.9. To get an actual image of our scene $L(x, \omega_o)$ (outgoing radiance) has to be computed for every image pixel.

2.3 Volume interaction

?? By now we have assumed that radiance is conserved along its ray path between surfaces. It's time to consider participating media interaction in our computation. To do this we should define some properties, which can be used to model and describe different types of volumetric phonomenons.

κ_a [m^-1] - Absorption coefficient

κ_s [m^-1] - Scattering coefficient

[4] This refers to the light distribution in the scene.
[5] Both $L(r(x, \omega_i), -\omega_i)$ and $L(x, \omega_o)$ are unknown.
[6] $L_e(x, \omega_o)$

κ_t [m^-1] - Extinction coefficient is composed from absorption and scattering coefficients as follows: $\kappa_t = \kappa_a + \kappa_s$.

We distinguish four different types of interaction events:

1. **Volume emission:**
   To simulate any light emitting volumetric phenomenons like fire or aurora, we have to add some energy to the ray going through this media.

2. **Absorption:**

$$dL(x,\omega) = -\kappa_a(x) * L(x,\omega)dx \tag{2.10}$$

3. **In-scattering:**

4. **Out-scattering:**

$$dL(x,\omega) = -\kappa_s(x) * L(x,\omega)dx \tag{2.11}$$

Transmitance $dL(x,\omega) = -\kappa_a(x) * L(x,\omega)dx - \kappa_s(x) * L(x,\omega)dx = -\kappa_t(x) * L(x,\omega)dx$
Integration over ray path: $L(x,\omega) = L(x_0,\omega) * e^{-\int_{x_0}^{x} \kappa_t(u)du} = L(x_0,\omega)\tau(x_0,x)$
For better understanding take a look on the picture fig(emis abs out in).

2.3.1 Phase functions

To model a light interaction on a molecular level, we use phase functions[7], which can be used to model different volumetric media types. It describes the probability of scattering light from incoming direction ω_i to the outgoing direction ω_o.

Phase function has got these useful properties:

1. **Function domain:**
   Is the same as in the case of the BRDF function. For the given point in the space it is four dimensional function. Two dimensions for incoming and to for the outgoing direction.

[7] Analogically to the BRDF functions, which model light surface interaction on microscopical level and simulate different surface materials.

2. **Value range:**
   Greater than zero and it's very useful to normalize phase function values so that:

$$\int_{\Omega 4\pi} p(\omega_i, x, \omega_i) = 1 \tag{2.12}$$

   This way we can be sure that the energy will be conserved.

3. **Reciprocity:**
   The phase function value stays the same if we interchange incoming and outgoing direction.

4. **Conservation:**
   The incoming radiance is always greater or equal to the sum of the distributed energy into all directions.

2.3.1.1 Reyleigh Scattering

$$I = I_0 \frac{8\pi^4 N \alpha^2}{\lambda^4 R^2} (1 + cos^2(\theta)) \tag{2.13}$$

2.3.1.2 Heney-Greenstein

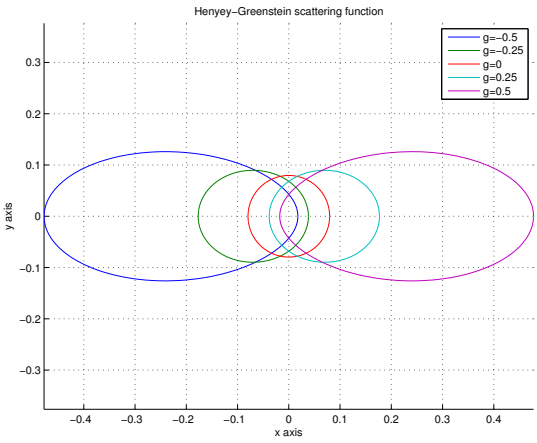

Figure 2.2: Plot of Heney-Greenstein phase function with different g coefficients. For g<0 the function represents backscattering media, for g=0 isometric scattering and for g>0 growing forward scattering tendencies.

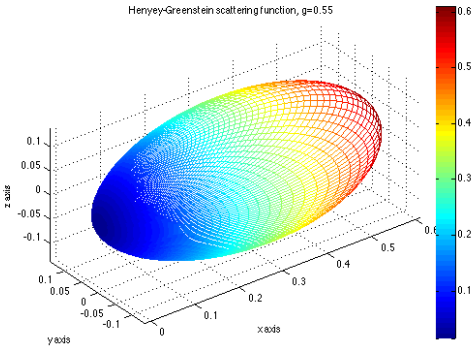

Figure 2.3: 3D plot of Heney-Greenstein phase function with scattering coefficient set to 0.55, which results in moderate forward scattering.

-isotropic -anisotropic name hg,shlick aprox, reylight (atmosphere) -jak dochazi k rozptylu svetla -pouzit matlabacky obrazky heney-g funkce ruzne koeficienty

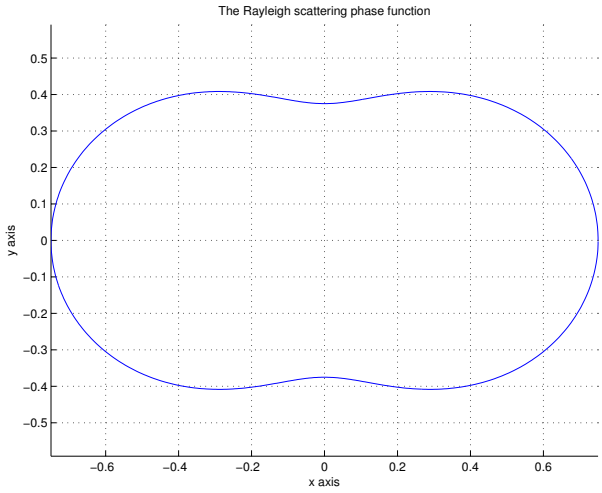

Figure 2.4: 2D polar plot of function Reyleigh scattering function.

2.3.2 Rendering equation including participating media

hezke odvozeni je v -advanced illum techniques

What steps have to be taken to get rendering equation to the next level.

2.4 Volume and surface interaction

In order to properly simulate light transport in the scenes containing both surfaces and volumetrically active media.

Different light transport paths surface surface, surface media, media surface, media media.

2.4.1 Extended rendering equation

Put together all the integrals pokracovat v odvozeni z -advanced illum techniques

2.5 Volume representation

We have got many options for volume representation in computer graphics. Rather than representing individual particles[8] we usually use larger volumes of space using quantization or some kind of bounding volume representation.

- **Particle representation** This representation is widely used in games and movies, for phenomenons like sand, dust. The main advantage is that this representation fits nicely into rasterization pipeline, where each point can be rendered out as a semitransparent point or patch.

  Though it is not very suitable representation for raytracing, because it is really hard to ray-trace something infinitely small without aliasing artifacts. For ray-tracing one can estimate a volume density based on the neighboring particles. To perform efficiently neighborhood queries an acceleration structure would have to be build upon these points.

- **Bounding volume representation** We can use any type of boundary representations and turn them into volumetrically active objects. The easiest way is to use parametric primitives such as boxes, spheres or cylinders. But efficient algorithms for mesh geometry exist too. This way we can represent just the boundary of the volume quite efficiently and fill it with any type of 3D texture[9] or value to alter it's optical characteristics[10] to get desired results. See fig [**?**] for examples.

- **Grid representation** To have an absolute control over each voxel[13] values we can use some kind of grid representation. For example fluid simulations or MRI[14] data come in a form of regularly spaced 3D grids. To over come a cubic space complexity of this representation more evolved structures, such as octree can be used[15].

[8] Even if we want to render large amounts of "particles" we usually might choose to render them as a point cloud where every particle has got one pixel size, this representation is not based on any physical prerequisites, though might serve well in many scenarios.

[9] Be it parametric or classical image stack texture.

[10] Characteristics like absorption, density etc. Discussed in section [**?**]

[13] Also known as *volumetric pixel* is a volumetric analogy of pixel in 2D. We can imagine it as an element of regular 3D grid.

[14] MRI stands for *magnetic resonance imagery* very often used in radiology for noninvasive interior medial examination.

[15] Lately a new industry standard format [Ani] for efficient storage and access of sparse volumetric datasets has been established

Figure 2.5: Left image is glass with juice[11], the volume is bounded by triangle mesh, source [NNDJ12b]. On the right we can se a simple procedural box filled with heterogenous fog[12]

Figure 2.6: Example of a volumetric MRI data set consisting of an image stack. Rendered using Voreen [RDRS10].

Figure 2.7: Smoke simulation with a simulation grid overlay, each grid cell represents one voxel with different density[1].

[1]For the smooth looking results linear interpolation of neighboring values is used.

Chapter 3

Common solutions

In this chapter we will try to summarize the most common technics for solving the rendering equation in the presence of participating media.

3.1 Rasterization

Films -

krakatoa rendering engine

nvidia smoke particles demo pixar deep shadow maps [LV00]

Still most used in films and games - fast approximation. Particle rasterization + rendering slices of 3D volumes can solve only direct illumination.

Figure 3.1: Examples of Krakatoa renderings in feature film productions [Thi]

.

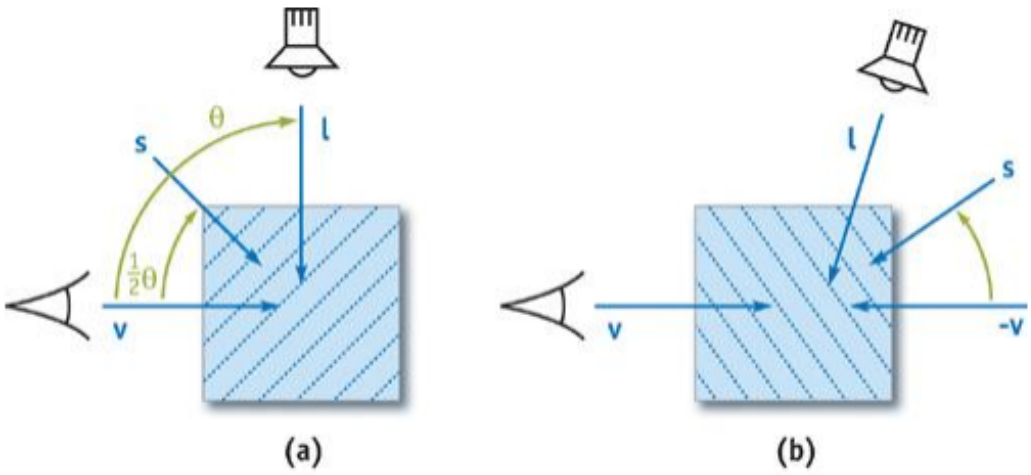

Figure 3.2: Ilustration of the half axis angle determination for particle for single particle sorting pass used for both light view (shadow map) and camera view (shaded image) rasterization passes. Source [NVI].

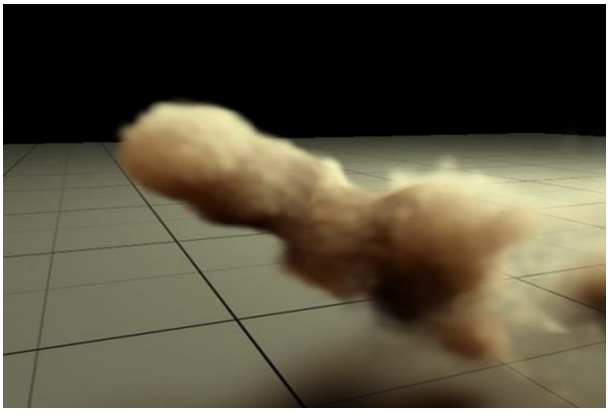

Figure 3.3: This image has been rendered using multiple shadow maps for particle direct illumination on gpu. Source [NVI].

3.2 Raytracing

Quasi Monte Carlo equation

$$\int_{I^S} f(x)dx \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)),$$

3.2.1 Raymarching volumes

More recently, another technique has become popular for sampling distances along a ray in an inhomogeneous medium. The idea comes from the neutron transport community in the 60s and has various names (delta tracking, pseudo-scattering); we'll call it Woodcock tracking in effort to promote original paper.

pbrt single scattering can support procedural volumes and any type of lighting

3.2.2 Unbiased methods

Plus and cons - very slow no caching recomputation of visibility factor

3.2.3 Biased methods

[Jar08] irradiance caching, Photon tracing, final gather ... Plus and cons

Chapter 4

Consistant progressive methods

what does it mean to be progressive

4.1 Stochastic progressive photon mapping

[]

4.2 Beam mapping

[JNT+11] Hybrid solution cpu gpu rasterization..

4.3 Virtual point lights

[Kel97] [HKWB09]

4.4 Virtual ray lights

[NNDJ12b]

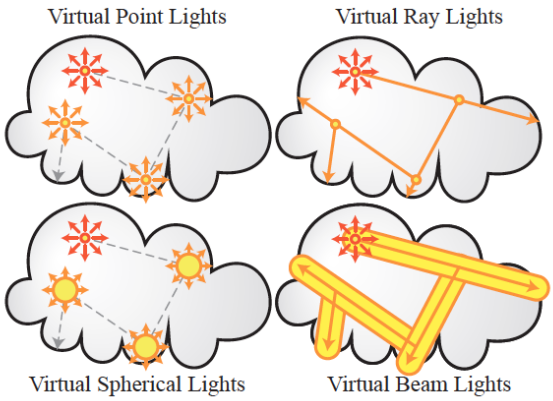

Figure 4.1: During the preprocessing stage photon paths are shoot into the participating media. This image compares the methods we can store and evaluate cached radiance in the form of virtual lights. This figure demonstrates the analogy of virtual spherical lights to virtual point lights with virtual beam lights to virtual ray lights. Source [NNDJ12a].

4.5 Virtual beam lights

As you can se on the image fig 4.1 [NNDJ12a]

Chapter 5

Proposed solution

I have chosen these methods and why... How to do it scheme of the progressive raytracing renderer.

co tu ma byt pouzil jsem tuhle a tuhle metodu proc odkazat se na predchozi kapitolu a zminit zmeny ktere jsem provedl a proc.

Co jsem orezal co rozsiril.

A dojit k blokovemu schematu co budu muset imlementovat. Mozna zminit i PBRT.

Chapter 6

Implementation

6.1 Volume representation

both unified grid and procedural volume (multiple octaves of perlin noise)

I have decided to implement it using pbrt. what is pbrt what structures I will use, which I have to create.

6.2 Ground truth

light tracer - dual algorithm to path tracer. It is able to capture caustic effects. Maybe path tracer too??

V tehle kapitole muzu i ukazat ty veci z matlabu ze funguji a jak. Klidne popsat, i postup implemntace pres jednoduzsi az po slozitejsi metody.

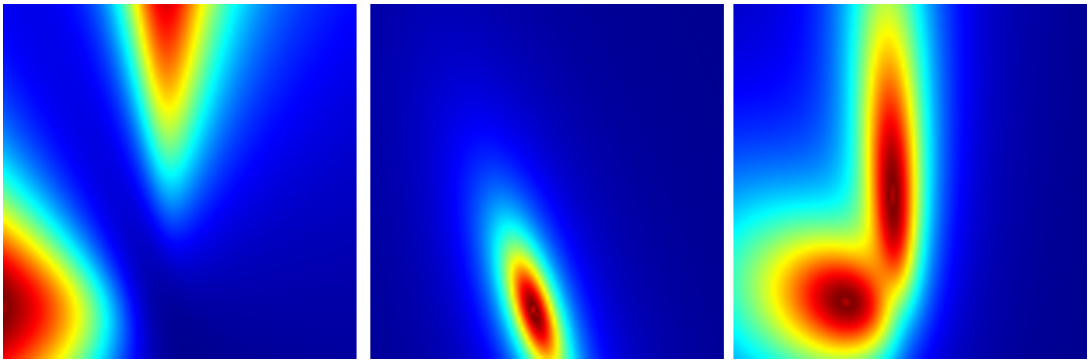

Figure 6.1: Different functions have been evaluated in discreet point pairs (from 0 to 1 parametric range) on two randomly placed rays.
On the left image only contribution using ray moderately forward scattering (g=0.75) Heney-Greenstein phase functions has been evaluated. In the middle inverse squared contribution and on the right contribution using both phase function and inverse sqared distance.

Chapter 7

Results

Mely by tu byt tabulky a srovnani metod a ruzne parametry.

Chapter 8

Conclusion

this method was implemented and results are great

8.0.1 Future work

Bibliography

[Ani] DreamWorks Animation. Opensource format for efficient storage of sparse volumetric data.@ONLINE. Available from: `http://www.openvdb.org/`.

[HKWB09] Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Trans. Graph.*, 28(5):143:1–143:6, December 2009. Available from: `http://doi.acm.org/10.1145/1618452.1618489`, `doi:10.1145/1618452.1618489`.

[Jar08] Wojciech Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, September 2008.

[JNT+11] Wojciech Jarosz, Derek Nowrouzezahrai, Robert Thomas, Peter-Pike Sloan, and Matthias Zwicker. Progressive photon beams. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2011)*, 30(6), December 2011.

[Kel97] Alexander Keller. Instant radiosity, 1997.

[LV00] Tom Lokovic and Eric Veach. Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 385–392, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. Available from: `http://dx.doi.org/10.1145/344779.344958`, `doi:10.1145/344779.344958`.

[NAS] NASA. Laser optics. Available from: `http://gimp-savvy.com/`.

[NNDJ12a] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. Progressive virtual beam lights. *Computer Graphics Forum (Proceedings of EGSR 2012)*, 31(4), June 2012.

[NNDJ12b] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)*, 31(4), July 2012.

[NVI] Simon Green NVIDIA. Nvidia cuda sdk example, smoke particle demo. Available from: `http://docs.nvidia.com/cuda/cuda-samples/`.

[PN] Petra Glazerova Petr Nejtek. Crepuscular rays.

[RDRS10] Timo Ropinski, Christian Döring, and Christof Rezk Salama. Advanced Volume Illumination with Unconstrained Light Source Positioning. *IEEE Computer Graphics and Applications*, 2010.

[SAL] Image Science and NASA Johnson Space Center Analysis Laboratory. Aurora borealis from iss. Available from: `eol.jsc.nasa.gov`.

[Thi] Thinkbox. Thinkbox software's production-proven volumetric particle rendering@ONLINE. Available from: `http://www.thinkboxsoftware.com/krakatoa/`.

Appendinx A

Appendix

abreviations + test scene images

Appendinx B

DVD Content

galerie obrazku test scen