Otto-von-Guericke-University Magdeburg
Faculty of Computer Science
ISG - Computational Visualistics

Master Thesis

Hierarchical Global Illumination by Using Cone Casts

by

Johannes Jendersie

September 22, 2014

Supervisors: Jun.-Prof. Dr. Thorsten Grosch
Dr. Dirk Lehmann

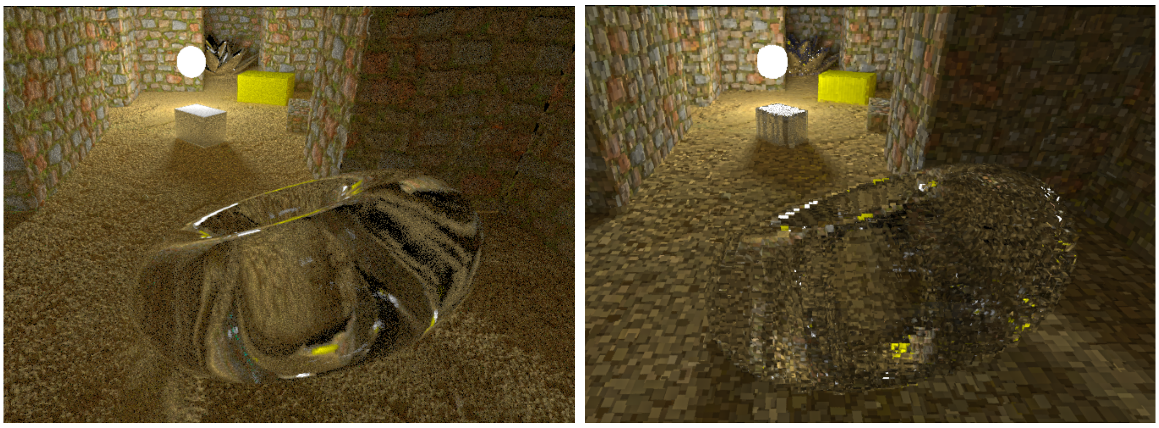
Abstract

Casting rays is a very common elementary operation in light simulation methods. We replace rays by cones to estimate a level of detail in a hierarchically represented scene. This enables us to use only a part of the scene hierarchy to compute a full global illumination. Hence the performance and memory footprint become proportional to the image resolution and the method can be used to render large out-of-core scenes.

In contrast to former cone casting methods we do not try to find an intersection between a cone and the scene. Instead we only trace the central ray and sample the cone interior stochastically. This simplifies the intersection problem to rays where we use the cone radius for the level of detail estimation only. Whereas, reducing the radius towards zero leads to a standard ray tracing. Therefore it is possible to trade off correctness against performance.

Further we exploit the discretization of the scene to store intermediate results of diffuse illumination to increase the profit from recursive ray tracings. A novel light distribution pass also stores illumination directly, but generates artifacts yet.

Unsolved problems are a reliable reflection distribution for clustered scene geometry and the generation of caustics in the view-importance reduced scene.



Direct rendering on a coarse hierarchy level
Left: path traced reference; *Right*: direct hierarchical illumination with only the top 1.8% of the hierarchy

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht. Alle Bilder und Skizzen wurden selbst erstellt und nicht aus anderen Quellen entnommen.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Magdeburg, ...
Johannes Jendersie

Contents

Symbols

ω	**Solid angle** ($d\omega, \Delta\omega$), or two angles (θ, ϕ) in a 3D-space (ω_i, ω_o)
θ	The angle between ray and surface normal $\mathbb{n}$, or first component of ω
α	**Half opening angle of a cone**
Φ	**Luminous flux**, used to describe the light transport between objects
I	**Luminous intensity**, good size for light source descriptions
E	**Luminous illuminance** (irradiance), incoming flux per area
B	**Radiosity**, outgoing flux per area
L	**Luminance** (radiance), flux per visible area and solid angle
$L(\omega)$	Luminance from/into a direction
L_e	(Self) **Emitted luminance**
L_D	**Stored diffuse luminance**
$\mathcal{E}(x)$	**Expected value** of x, x can be a variable or function
$p(x)$	**Probability density function**, x can be any input parameter(s)
N	**Number of samples** in a random experiment
ξ	**Standard random number** sample $\in [0, 1]$
$(x)_+$	Short for $\max(0, x)$
$f(x), g(x)$	General unnamed functions
f_r	**Bi-directional reflection distribution function**
F_{rs}	**Light form factor**, for the transport of light between surfaces
ρ	**Diffuse reflectance**, ratio between incoming and outgoing flux
s	**Shininess**, exponent in the Phong BRDF
d	**Euclidean distance**
A	**Area**
$\mathbb{n}$	Surface **normal**
$\mathbb{x}$	**Location** in 3D, a 3 component vector
$\mathbb{d}$	**Direction vector** with length 1
$\mathbb{r}$	**Reflection vector** with length 1
$\mathbb{o}$	**Origin** of a ray/cone
t	**Ray parameter** from parametric form $\mathbb{o} + t \cdot \mathbb{d}$
r	**Radius** of a circle or cone
$\mathcal{O}$	Landau notation for worst case asymptotic run time

1 Introduction

Global illumination (GI) is a term used for a number of effects which appear when simulating light such as direct illumination, indirect illumination, reflections and caustics. Algorithms to simulate light came up in the 80s and 90s. Three approaches, path tracing [Kaj86], photon mapping [Jen96] and radiosity [GTGB84], are basic techniques, also used in many recent methods. We also use a combination of path tracing and photon mapping and store diffuse illumination similar to the radiosity method.

The mentioned algorithms are often too slow for their application fields like the movie industry and augmented realities. When mixing virtual characters and scenes with the reality in a movie the correctness is very important. On the other hand a video consists of many frames in high resolution which make it takes hours [Chr08] to compute a few seconds, while artists must be able to change the scene and want be able to see immediate results. Similar needs for quality hold for other augmented realities where a real scene is extended by virtual objects. The long-term goal is to be able to add those virtual objects in real time even on weak mobile devices.

1.1 Requirements and Goals

The target of this work is to find an approach which is as correct as possible while being interactive for huge scenes. Consequently, it must be able to support out-of-core scenarios. It should be possible to chose between quality and performance by a simple control factor. Further, it should be flexible in the number and type of supported materials and the resulting visual effects. It must cover direct and indirect illumination, reflections, refractions and their resulting caustics.

To achieve these targets a method is designed which can compute all kinds of illumination directly on a hierarchy without need of the underlying geometry. This enables out-of-core scenarios where only relevant parts are loaded. Additionally, it increases the performance due to hierarchical sampling. The out-of-core loading process is not part of this work, rather it is only evaluated how cone casts perform on a hierarchy and how that hierarchy can be utilized to increase the convergence speed.

1.2 Implementation in Brief

The developed technique is based on a point based representation of a scene (section 4.1). The surface elements (surfels, see [PZVBG00]) contain a compressed description of a bidirectional reflection distribution function (BRDF) and the current diffuse illumination. In contrast to other approaches, a surfel is not pre-illuminated by direct light. Upon it a hierarchy is created which approximates both the geometry and material properties of the clustered geometry.

Then we introduce a cone cast operation which selects different level of details (LODs) from the hierarchy based on the cone radii. Contrary to earlier cone tracing [Ama84] we do not try to get a smooth antialiased result for a cone. Instead we use a stochastic variant to simplify the intersection problem and make cone casts faster. Therefore a normal ray is traced and the radius of the cones is used to select nodes which fit into the cone as intersection targets. To sample the full interior of the cone unbiased the central ray is resampled randomly before the cast.

To simulate the illumination we use the cone casts in two ways. The first method is a unidirectional path tracer where intermediate results are stored in the nodes of

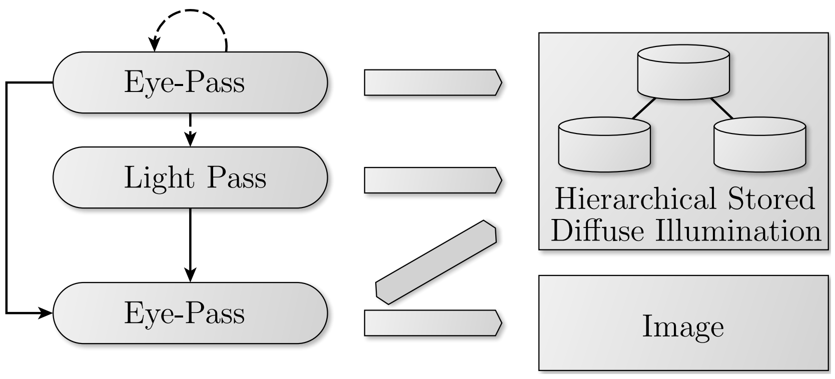


Figure 1 Light simulation algorithm. The first eye-pass determines the node impotencies and stores results in the hierarchy. Then an optional light distribution pass can be done before the image is rendered using and upgrading the stored information

the path to increase the convergence speed. In that tracer the cone radii is primary determined by the pixel's solid angle. It is possible to narrow the cone to interpolate towards the correct solution. In the following one iteration of the path tracer for all pixels will be called eye-pass.

As a second method we propose a light distribution pass which works similar to photon mapping. Again cones are used to generate light paths for small particles. Instead of storing these photons in a map they are directly applied to the stored information which is used by the path tracer pass. Therefore the global probability to hit the nodes must be estimated which introduces several artifacts in the current implementation. The advantage of the light distribution pass is that it can be switched off after a short time when the caustics are converged. Further gathering iterations can be computed faster afterwards.

In an experiment we evaluate if the light distribution pass can be computed with the detail level chosen from the eye-pass to keep a small memory footprint. The results are very noisy but still relative meaningful. It could be possible to distribute photons without loading further scene information using another method for the photon gathering.

1.3 Structure

Chapter 2 introduces the physical basics, algorithms and light simulation methods which are necessary to understand the new method. Then chapter 3 shows up existing extensions and further techniques related to this work.

Chapter 4 explains the cone casting and simulation steps in detail. First of all, the representation of the scene and hierarchy is discussed where several approximations for material properties and illumination are made in the nodes (sections 4.2, 4.3, 4.4). We use a bounding volume hierarchy built with a slightly modified kd-tree. Each node in this tree has an approximated BRDF and illumination information. Our model is designed to be as small as possible which leads to a loss of generality. The errors and memory requirements are evaluated later in chapter 5. We show that a scene can be rendered with a constant number of nodes which is proportional to the number of pixels (5.2). Because of the size of the proportionality factor ($\approx 200\frac{\text{nodes}}{\text{pixel}}$) and the made errors the approach is not as performant as expected and real out of core scenarios still do not fit in memory.

After the setup of the scene representation, the two methods consisting of path tracing and light tracing are applied to the scene. Section 4.5 describes the eye-pass in detail. The focus lies on the use of hierarchical information wherever possible by choosing appropriate LODs. In this context the problems of surface curvature and meaningful light-path generation are discussed.

Taking advantage of the discretization of the scene by choosing LODs, light can be stored fuzzily. This is used to compute the diffuse illumination in all nodes instead of computing the average sampling results on the image only, as in conventional path tracing. This increases convergence speed and reuses all rays (sections 4.6, 5.8). Specular indirections are not stored because a node might be observed from any direction by reflected rays and thus the view dependency of specular reflections would need to be treated.

Section 4.7 introduces the independent light distribution pass which also exploits the discretization to store illumination directly instead of photons. This way caustics can be generated and a bidirectional component is added to the total lighting progress. This pass involves the computation of a global probability which is also discussed in this section.

Section 4.8 describes how the hierarchy is updated after both passes. In case of the eye-pass this is not necessary but improves the illumination information stored in lower detail levels. The light distribution needs to push and pull this information to be correct, otherwise light would be lost. This step is very similar to the push-pull in the hierarchical radiosity method [HSA91] but differs a bit to make the two passes in our approach independent.

Finally the approach is evaluated in chapter 5. It is possible to create results, using only a fraction of the available geometry. With a decreasing set of used scene information the performance increases (section 5.3). Still, like other stochastic methods, the results suffer from noise and some of the steps are making too large errors, so further improvements need to be done. The current implementation is using only the CPU and thus does not achieve interactive frame rates.

2 Fundamentals

For the hierarchical illumination we tried to mix multiple techniques which are known to solve certain problems well which are path tracing as general solution, photon mapping for caustics and radiosity for diffuse illumination. This section will explain the most fundamental ideas and how their relevance to this work.

2.1 Photometric Values

The transport and perception of light can be described by a list of physical quantities. They are all applicable for different cases, but still are related to one another. The following formulas are the base for most global illumination approaches.

In reality, light is an energy in form of discrete photons which have both wave and particle characteristics. All materials absorb, reflect or emit photons which interact again with other surfaces. The energy of a photon determines its frequency, resulting in a color, and their number defines the brightness. Radiometry is describing the full spectrum of all frequencies and photometry the visible portion ($\frac{c}{f} = \lambda \in [380, 780]nm$) only. For the conversion from radiometric to photometric values the integral over the full visible spectrum must be taken weighted by the perceived brightness of the respective frequency. The factor for monochromatic light of $555nm$ is $683\frac{lm}{W}$ defined during the standardization process of photometric units. The following units are that of photometry where global illumination is set.

$$\text{Solid angle} \qquad \omega \qquad [sr] \tag{2.1}$$

$$\text{Luminous energy} \qquad Q \qquad [lm \cdot s] \tag{2.2}$$

$$\text{Luminous flux} \qquad \Phi = \frac{dQ}{dt} \qquad [lm] \tag{2.3}$$

$$\text{Luminous intensity} \qquad I = \frac{d\Phi}{d\omega} \qquad [cd] \tag{2.4}$$

$$\text{Luminous illuminance} \qquad E = \frac{d\Phi}{dA_{\text{receiver}}} \qquad [lx] \tag{2.5}$$

$$\text{Radiosity} \qquad B = \frac{d\Phi}{dA_{\text{sender}}} \qquad \left[\frac{lm}{m^2}\right] \tag{2.6}$$

$$\text{Luminance} \qquad L = \frac{d\Phi}{dA \cdot \cos\theta \cdot d\omega} \qquad \left[\frac{cd}{m^2}\right] \tag{2.7}$$

A solid angle ω is the 3D equivalent of an angle in 2D. Instead of measuring the arc length on a unit circle (radian) the surface of the volumetric angle on a unit sphere is taken (steradian). The size of the full angle is 4π (surface of a unit sphere). The solid angle describes the fraction of space covered by the angle, e.g. a cone in the unit sphere.

The sizes Q and ϕ are the amount of light and the amount of light per time unit. In photometry this is the result of the integration over the spectrum taking the perception into account. Otherwise these two and all other quantities must be written as a function of the frequency h.

Luminous intensity (referred to as 'intensity' later on) is the energy per time and steradian which can be used to describe the outgoing light distribution from non uniform light sources or the light radiated by some object.

For incoming light the quantity is luminous illuminance E whereas for outgoing light the radiosity B is used. Both have the same basic units $\frac{lm}{m^2}$ and describe the energy received or sent per time unit and area. Both of them do not account for the angle from/to which the light is coming/sent.

However, the luminance L takes account of both the area and the angle. Since the area is changing dependently on the angle the term $\cos\theta$ transforms the area dA to a projected area, which considers the visible amount of surface from the specific direction. The angle θ lies between the incident/excident direction and the surface normal.

2.1.1 Derived Dependencies

From the upper equations some useful connections can be derived. Most of the time luminance L, intensity I or radiosity B are given and luminance L or illuminance E are searched. In the following, incident values are indexed with i and excident ones with o (outgoing).

The illumination as integral over the incoming luminance per solid angle over a hemisphere can be derived from equations 2.5 and 2.7:

$$E = \int_{2\pi sr} L(\omega_i) \cdot \cos\theta_i \cdot d\omega_i \tag{2.8}$$

The integral boundary must be increased to $4\pi sr$ if light can come from inside the object itself e.g. for volumetric effects such as subsurface scattering.

To compute the light transport between two surfaces the photometric law of distance can be used. Figure 2 shows how the values are related.

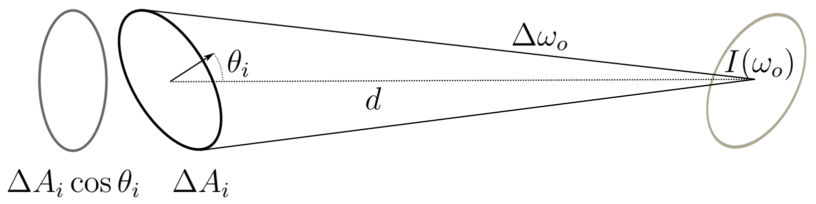


Figure 2 Light transport between surfaces; The surface on the left receives a flux of $I(\omega_o)$ times $\Delta\omega_o$ which is approximated by the projected area $\Delta A_i \cos\theta_i$ and the distance d.

Therefore the illumination on the target side can be approximated as:

$$E \approx \frac{I(\omega_o) \cdot \Delta\omega_o}{\Delta A_i}$$

by inserting equation 2.4, transformed to $d\phi$, into 2.5. The solid angle $\Delta\omega_o$ is approximated by the quotient of projected area and the squared distance (see figure 2):

$$\begin{aligned} \Delta\omega_o &\approx \frac{\Delta A_i \cdot \cos\theta_i}{d^2} \\ E &\approx \frac{I(\omega_o) \cdot \cos\theta_i}{d^2} \end{aligned} \tag{2.9}$$

2.2 Bidirectional Reflection Distribution Function

Once the illumination E is determined if must be turned into a luminance L by interacting with the material. Commonly the reflection characteristic of light at surfaces is modeled by the BRDF, which is in this context a factor of how much from the illumination is reflected from the material into a specific direction:

$$f_r(\omega_o, \omega_i) = \frac{dL(\omega_o)}{dE(\omega_i)} \tag{2.10}$$

$$= \frac{dL(\omega_o)}{dL(\omega_i) \cdot \cos\theta_i \cdot d\omega_i} \tag{2.11}$$

The second form can be derived from the first by using equation 2.8.

BRDFs of real materials can be determined by measuring $dL(\omega_o)$ in an environment of known illumination $dE(\omega_i)$ or luminance $dL(\omega_i)$. The measured values create a huge amount of data. For rendering purposes there are lots of analytical models which represent common observable reflection patterns.

2.2.1 Lambert Emitter and Lambertian Materials

Many material models use a combination of independently modeled BRDFs. A Lambert emitter is a light source for which the radiated luminance is independent of the outgoing direction.

$$L(\omega) = const \tag{2.12}$$

$$\Rightarrow \; I(\omega) = I_0 \cos\theta \tag{2.13}$$

Lambertian materials are such which become a Lambert emitter under illumination. They reflect the incoming light uniformly in all directions which causes a factor of $\frac{1}{\pi}$. Additionally fractions of the light are absorbed depending on the frequency which is expressed by the reflectance $\rho \in [0,1] = \frac{\Phi_o}{\Phi_i} = \frac{B}{E}$.

$$f_r(\omega_o, \omega_i) = \frac{\rho}{\pi} \tag{2.14}$$

A material with this property is called diffuse. There are no existing materials which perfectly satisfy this condition. However, the term can be used as a basis for approximations, especially in combination with other physical laws.

2.3 Path Tracing

Path tracing is the a stochastic solution to simulate light which was introduced together with the **rendering equation** by Kajiya [Kaj86]:

$$L_o(\mathbb{x}, \omega_o) = L_e(\mathbb{x}, \omega_o) + \int\limits_{2\pi sr} f_r(\mathbb{x}, \omega_o, \omega_i) \cdot L_i(\mathbb{x}, \omega_i) \cdot \cos\theta_i \cdot d\omega_i \tag{2.15}$$

The general idea behind path tracing is to compute samples L_i by ray casts to the scene. Starting at a location $\mathbb{x}$, rays are casted and intersections are searched. When the ray hits an object it is redirected according to its material and traversed recursively to a maximal depth. The luminance at the hit location is returned to the ray sender and transformed by f_r (2.11) to a new outgoing luminance recursively. Finally each ray returns the luminance for a random path in a certain direction. Instead of generating infinite many rays per hit point only one path is traced and the integral is determined stochastically via Monte Carlo integration.

2.3.1 Monte Carlo Integration

Monte Carlo integration is the stochastic estimation of a definite integral using N random numbers. Due to the law of large numbers the expected value converges to the correct (unbiased) value if N goes to infinity. Per definition the real expectation value $\mathcal{E}$ can be obtained by an integral over the target function $(g(x))$ times the probability density function $p(x)$

$$\mathcal{E}(g(x)) = \int_a^b g(x) \cdot p(x) \cdot dx \tag{2.16}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} g(\chi_i) \tag{2.17}$$

In equation 2.17 the argument χ_i is a sample of the random variable $\chi(\mathbb{x}, \omega_i)$. With $f(x) = g(x) \cdot p(x)$ the equation can be rewritten to

$$\int_a^b f(x) \cdot dx \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(\chi_i)}{p(\chi_i)} \tag{2.18}$$

This sum can be applied directly to solve the rendering equation 2.15.

2.3.2 Importance Sampling

It is possible to use a uniform density function $p = \frac{1}{2\pi}$ to generate samples for the integration during path tracing. However, it can be shown that a density function $p(x) \sim f(x)$ converges faster.

Importance sampling is trying to estimate the unknown $f(x)$ as weight for the sample generation $p(x)$. Often the BRDF f_r or a related function is chosen because it is the best locally known measure for weighting the light integral assuming a uniform luminance over all solid angles. Another good choice is to use the already known samples to estimate the importance of certain solid angles thus getting $p(\chi_i) \sim L(\chi_i)$ or even $p(\chi_i) \sim f_r \cdot L(\chi_i)$.

2.4 Space Partitioning Trees

For all ray casting operations, e.g. path tracing or shadow test, intersections of rays with the scene surface must be found. There are several different kinds of spatial search structures which can be used to faster find these intersections. Among the best known ones are octrees and kd-trees.

Quadtrees, octrees and higher order variants subdivide a space into 2^k cells where k is the number of dimensions. This subdivision can be done adaptively until the number of primitives in each cell is below a certain threshold. Leaving the empty cells as large as possible results in a sparse octree while subdividing all cells creates a full tree. By marching from cell to cell octrees can be used for efficient ray casts e.g. as applied in Gigavoxels [Cra11].

Contrary a kd-tree is a binary tree which divides the k dimensions interleaved. The cut of the geometry is commonly placed at the median, leading to a perfectly balanced tree. Instead of rotating through all dimensions regularly one can also always split

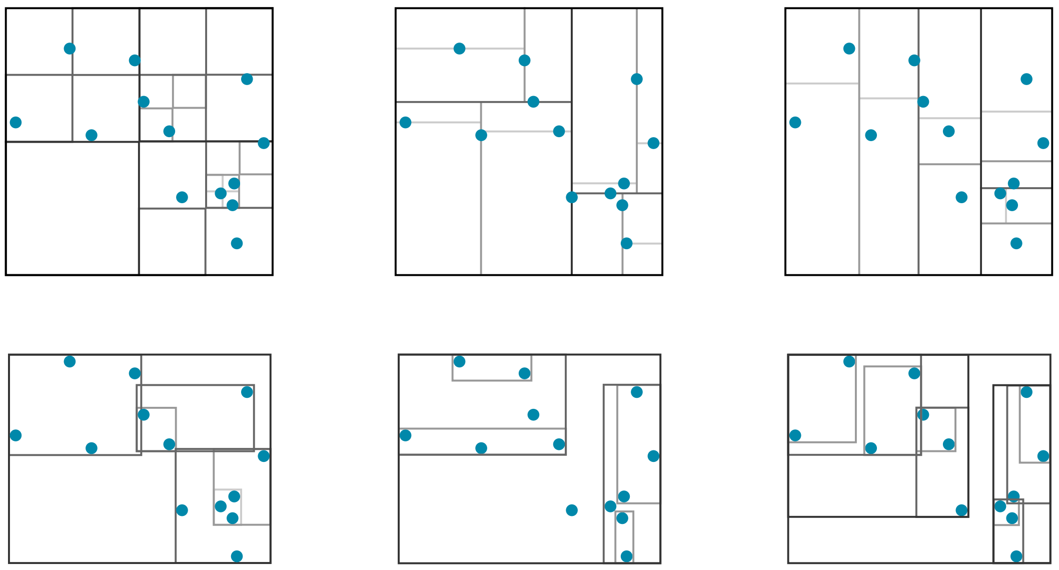


Figure 3 *Left*: 2D-Octree (Quadtree), *Middle*: kd-tree, *Right*: Leave oriented kd-tree with the modification to split along the dimension of largest expansion. *Second row*: the bounding boxes for the nodes' content respectively. The modification forces more regular side lengths.

along the largest remaining extension in the assigned data. This leads to a more regular subdivision in terms of the bounding volumes for the contained data as is visible in figure 3.

A problem in all space partitioning trees is to store non-point geometry which resides inside multiple nodes. In this case it can either be referenced form all such nodes or stored in the smallest node containing the full object.

Even if balanced it is not possible to guarantee a $\mathcal{O}(\log n)$ query time, where n is the number of primitives in the scene. In both described data structures scenes can be constructed where certain queries take $\mathcal{O}(n)$ time. E.g. imagine a ribbon parallel to the view direction. A ray which passes very close must test against all primitives of the ribbon without hitting any.

We used a kd-tree because it adapts much better to the local density of scene geometry while using as little space as possible. The primitives are stored leaf-oriented and each node above contains a clustered lighting information for the two children. To build the tree the geometry is treated as dimensionless (points without extension). Later the bounding boxes are computed per node which leads to a bounding volume hierarchy (BVH). So we use the kd-tree to construct a BVH rather than using the kd-tree itself.

2.5 Photon Mapping

A different solution for the lighting problem is photon mapping where particles of light which are emitted by the light sources are distributed among the scene. It was invented in 1996 by H. W. Jensen [Jen96]. In contrast to path tracing photons are distributed from the light source, instead of searching the lights from the point of view. Finally photons are locally collected for illumination.

First the luminous flux of the source is divided into N equal parts. Each is sent to the scene by ray casting, again using a space partitioning tree. For the ray creation importance sampling is necessary for non-uniform light sources. In directions where the light is brighter more photons need to be sent, because they all have the same flux. Then, if a photon hits a surface it is stored in the photon map or reflected depending on the BRDF. If not absorbed by the material a new path is generated and the photon path is traced further. Photons are only stored at (partial) diffuse surfaces. In cases of reflections or refractions the photon does not contribute to the local illumination.

Usually the photons are stored in a data structure which allows fast range queries of points, for example in kd-trees. The number of photons which can be stored is limited by the memory and invariant to the complexity of the scene.

The second pass, called radiance estimation, uses ray casts, but instead of a Monte Carlo integration gathers the photons from the map at the hit points. To do so there are two possible options: Count photons in the local area via a range query or find the k nearest neighbors and determine the distance r to the farthest. All gathered photons are used to illuminate the target, where ω_p is the angle towards the photon and $\Delta\Phi_p$ is its flux.

$$L(\mathbb{x}, \omega_o) \approx L_e(\mathbb{x}, \omega_o) + \frac{1}{\pi r^2} \sum_{p=1}^{N} f_r(\mathbb{x}, \omega_o, \omega_p) \Delta\Phi_p \tag{2.19}$$

The collection of photons in the local area introduces several artifacts. It is possible that light bleeds through walls and convex corners take photons into account which should not contribute to the local illumination leading to an overestimation. Similar, the illumination of concave surfaces is underestimated. Some of the problems can be reduced by a modified gathering pass (see section 2.5.1). Additionally the collection of photons can be improved in several ways making the query itself more complex.

Another popular modification of the photon mapping is the introduction of a dedicated caustic-map. Photons which were reflected specular at least once (LSS*D paths) are stored in the caustic map. For the caustic map generation many small photons are used while for the standard map fewer larger ones are taken. To generate the two maps either rejection sampling must be performed or an importance sampling must directly generate rays into directions of specular and diffuse objects respectively.

2.5.1 Final Gathering

Final gathering is using a higher number of rays is the gathering pass to reduce the artifacts and the blurring introduced by the collection radius r. One ray is casted for each pixel where the following steps are done at the hit point:

- Trace reflections recursively if primary hit point is specular
- Compute the direct illumination explicitly (using many samples for area light sources)
- Query photons from the caustic map locally
- Estimate the indirect illumination with many ray casts and do the radiance estimate at the hit points
- Sum up all the results

The resulting image contains the illumination of both photon maps, has a nearly noise free direct illumination and overcomes the problem of light bleeding through walls most times, because the radiance estimate is done only for caustics and indirect light where those artifacts are less visible by far.

2.6 Micro Rendering

Micro rendering [REG$^+$09] is a point based global illumination technique which can be used as a replacement for final gathering. For each point hit by a ray cast from the eye the scene is rendered from that point's view. The result is weighted with the BRDF and integrated. During rendering the projection is warped according to the BRDF to increase the quality of important samples.

The rendering is done using a hierarchy of oriented discs (splatting). These contain the radiance estimate results of a photon mapper or other illumination information. Thus a cut through the hierarchy which has approximating pixel resolution in the micro-render-targets can be used to significantly increase the performance because only a rough approximation of the geometry is used.

Refractions and multiple (glossy) specular indirections are not supported by micro rendering. Diffuse and glossy materials are captured well and the overall performance is good.

2.7 Hierarchical Radiosity

A completely different approach to compute global illumination of diffuse materials are the radiosity methods [GTGB84, GCT86]. The idea is that each illuminated point becomes a light source (Lambert emitter) itself. Each has an area A an isotropic radiosity B which is stored and used to illuminate the other elements in the scene. For the light transport between all primitives the light form factors F_{rs} are computed from the visibility, the distance and the size of the sender s and receiver r.

$$F_{rs} = \frac{1}{A_r} \int\limits_{A_e} \int\limits_{A_s} V(s,r) \cdot \frac{\cos\theta_r \cdot \cos\theta_s}{\pi \cdot d^2} dA_s dA_r \tag{2.20}$$

$$B_r = E_r + \rho_r \sum_{s=1}^{n} B_s \cdot F_{rs} \tag{2.21}$$

Once all F_{rs} factors are computed there are n linear equations in the form of 2.21. This linear equation system can be solved directly but is extremely large and potentially dense.

Since solving this huge equation system is not performant, iterative methods like progressive radiosity [CCWG88] were invented. In each iteration the surface with the highest not distributed light is chosen and illuminates all other elements.

Distributing light in a hierarchy can solve the problem of too large equation systems by effectively reducing the number of senders and receivers. Hierarchical radiosity [HSA91] is motivated by the N-body problem. They subdivide sender and receivers to decrease the number of interactions. Therefore an oracle function is introduced to decide if the chosen level of light transport is sufficient. It is used to refine the scene before each iteration. The oracle function is the most complex part to implement because it must detect any local difference to subdivide and areas of similar behavior to summarize.

Using the oracle function each relevant node of the hierarchy can be illuminated by computing the sum over the chosen related surfaces. To avoid some of the used nodes being illuminated during one iteration while others are not, the illumination must be stored double buffered. After an iteration only a fraction of the nodes in the hierarchy is illuminated. Now the new results must be applied and pushed/pulled through the hierarchy such that all nodes have an equal amount of illumination, because not all nodes were chosen to be a receiver before. With the resulting illumination the oracle function can again refine the hierarchy.

Since refining the scene sufficiently for high frequency features as shadows leads to too many nodes again, a Final Gathering on a less precise illumination can be done too. Like before, the artifacts in detailed regions are hidden by the indirect illumination and the direct illumination is correct due to sampling.

In this thesis the diffuse lighting is also stored within a hierarchy. Consequently, steps like pushing and pulling the results are applied. Contrary to radiosity, the illumination per node itself is determined by ray casts from light and view position.

2.8 Summary

Each of the mentioned methods has its strengths and weaknesses. Finally our approach mixes ideas from all three to achieve qualitative results fast:

Path Tracing is unbiased and universally applicable for all scenes and materials. On the other hand the results are very noisy and without optimization it can take many hours to generate the final image.

We use path tracing as the basic underlying technique. It is extended by cone casts and the stored intermediate results.

Photon Mapping can be substantially faster than path tracing and is strong in generation of caustics.

Similar to casting photons, light is distributed to the scene and stored in the nodes as diffuse illumination in our method.

Hierarchical Radiosity creates noise free diffuse images but has several artifacts in high frequency areas and does not cover all materials and light paths.

Both photon illumination and intermediate ray casting results are stored in the hierarchy. With its push-pull steps and double buffering this is similar to hierarchical radiosity. Instead of computing the form factors F_{rs} and equation 2.21 the illumination is stochastically integrated via cones.

3 Related Work

Our hierarchical illumination approach is inspired by many ideas which were introduced in the last chapter. In the following other related techniques are shown in brief which also influenced this thesis.

First improvements and alternatives for the light simulation methods introduced in chapter 2 are presented (section 3.1). Afterwards further elementary tools for illumination are introduced. Among those the BRDFs section 3.2) which are describing the materials are most important. Section 3.3 references the spherical harmonics as potential representations for angular dependent functions. Finally, section 3.4 references the pseudo-random number generator used for stochastic processes.

3.1 Light Simulation Algorithms

There are many approaches which try to solve or approximate the rendering problem. Among these are modifications for path tracing which can cover more complex scenes faster (section 3.1.1). As alternative to ray casts, cone casts are introduced (section 3.1.2) together with other methods which make use of simplified geometry as point based global illumination (section 3.1.3). Then photon mapping is reviewed in section 3.1.4.

3.1.1 Path Tracing Derivatives

Path tracing is an unbiased global illumination approach introduced by Kajiya as a numerical solution for the rendering equation [Kaj86] which he derived. Lafortune gives a good summary of the topic in [Laf96].

While it is known to render photometrically correct results, it is very slow in its original form. To halve the noise in the final picture the number of rays must be quadruplicated. Many variations of Kajiya's original method were developed to improve the performance. Bidirectional path tracing, invented independently by Lafortune et al. [LW93] and Veach et al. [VG95], explicitly increases the importance of light paths by computing two paths simultaneously. One is casted from the light source and the second from the point of view (eye). Then the light transport between each sample of the light pass and the eye pass is calculated. This way caustics converge much better because they are generated by very likely paths from the view of light.

Similarly, the Metropolis Light Transport approach [VG97] searches for the most relevant paths. Starting with a single path originated at the light source, random mutations are made to the path e.g. adding, removing or translating a point in the path. New mutated paths are randomly accepted and the sample is applied to the image or the mutation is rejected. The rejection function tries to estimate the probability distribution according to the resulting light distribution to converge towards the searched light transport function. Metropolis is an unbiased flexible approach which can handle difficult lighting scenarios fast.

Other approaches try to reuse the recursive path-results like in [BSH02] where rays are shared between neighboring pixels. The result of a recursion is applied to an adjacent path too, such that additional paths are created at the cost of single ray casts to check the visibility of the path connection. This is similar to a blur in the path space so the parameters must be chosen carefully to reduce the resulting artifacts. Another method which reuses path is Eye-Path Reprojection [HBGM11] where intermediate results are always reprojected to the image plane. For low additional costs the number of paths per pixel is increased. Therefore a ray cast as done to check the visibility of the current path point and to find the pixel in which this point lies. The reprojection even allows to skip pixels when generating primary rays to increase rendering performance, because pixels are randomly filled through other passes. The technique is implemented on GPU and it was shown that a depth buffer can be used to determine the visibility of a point before reprojection.

In this thesis the illumination is also based on path tracing using cones where samples are reused by storing intermediate results. Nevertheless the other mentioned optimization methods are independent and could be included for further improvements. In our approach bidirectionality is introduced by a hybrid of unidirectional path tracing and a light distribution pass which is more similar to photon mapping.

3.1.2 Cone Tracing

To compute a ray tracing on a hierarchical level we generalized rays to cones. This idea is also very old. J. Amanatides came up with this idea in [Ama84], where he used it to increase the rendering performance and improve the anti-aliasing in the image. His main problem was to find intersections and intersection areas between cones and the scene. We avoid these problems by relying on the central ray only. In Amanatides approach the light inside a cone is approximated by a set of intersecting objects for which the areas and the area of overlap are computed. From the surfaces among these objects a representing point is chosen to proceed with the recursion. He mentions but does not describes that the cone radii must be changed during reflection depending on the surface curvature, which we did with a heuristic.

An other related technique is the tracing of ray differentials [Ige99] by H. Igehy. He calculates a differential of the ray-path parametrization without the need of adjacent rays to estimate an anisotropic area (footprint) at the hit point. This information is used to perform an improved texture sampling using mipmapping and other techniques. In contrast to a stochastic sampling this reduces the noise due to the texture LOD. The introduced ray differential is a differential of positions and directions at all hit points on the path. The costs of computation are relatively small and do not require the tracking of adjacent rays. This could be used in our approach to compute better cone radii when tracking rays at curved surfaces than with the heuristic.

There is also a real time technique based on cones called voxel cone tracing [CNS+11]. In a first pass the radiance and incoming direction of light is stored in a sparse octree near the surfaces in the scene. During rendering from the camera cones are generated dependent on the BRDF. Then indirect illumination is sampled from the light volume by using those voxels which fit into the cones.

Similar the Irradiance Atlas [CB04] stores lighting information from a photon mapping in a sparse voxel octree. In a final gathering pass the radiance estimate is then replaced by a lookup in the tree. To determine the required LOD for the lookup they use ray differentials.

3.1.3 Point Based Global Illumination

Surface elements (surfels) are small points representing atomic parts of the scene, often in form of oriented discs. They do not contain explicit connectivity as other surface discretizations [PZVBG00]. It is possible to use them for hierarchical approximations of large models as in [RL00]. Rusinkiewicz et al. achieve an interactive frame-rate rendering large models from scanned data by cutting through the hierarchy where each surfel with a size close to one pixel is rendered. The most common form of rendering surfels is splatting, where small ellipses (splats) are rasterized like usual triangles.

For global illumination surfels are used since [Bun05]. He uses them to approximate ambient occlusion and single bounce indirect lighting. The micro-rendering approach from Ritschel et al. [REG+09] uses splat representations to compute a final gathering pass, e.g. for photon mapping, with diffuse and glossy reflections. Points are easy to handle because they are dimensionless which makes them practical for out-of-core approaches like in [Chr08] and [KTO11]. In [Chr08] Christensen focuses on the illumination model itself which works similar to hierarchical radiosity [HSA91]. [KTO11] describes how octrees can be maintained efficiently for the purpose of out-of-core light simulation. They rasterize the scene from the view of each relevant surfel to generate a single bounce indirect illumination.

3.1.4 Photon Mapping Derivatives

Since Jensen's first approach [Jen96] many improvements where made. The speed could be increased by [PPI98] where an importance map is created in a third (pre-)pass to stir the creation of the photon map towards the visible scene. In [CJ02] photon mapping is optimized for animated scenes and motion blur. In this method photon mapping is generalized to distribute photons over space and time.

With progressive photon mapping by Hachisuka et al. [HOJ08] a multi-pass method was introduced which can guarantee an unbiased solution due to an unlimited amount of photons. Before that, photon mapping could not guarantee to find the correct solution of the rendering equation, because the number of photons was limited by the memory. In progressive photon mapping the photons are discarded after the gathering step of each pass. With each pass the gathering radius for the collection of photons is reduced to guarantee the convergence.

In contrast to a path tracer photon mapping usually converges faster to a result with less noise and is very good at generating caustics. Caustics are difficult to obtain otherwise, because finding a path from a point to the light source is much less probable than finding many paths from the source which are focused by reflections or refractions. Using photon maps to generate caustics were shown in [WUS03] to be suitable for real-time rendering. As many other hybrid techniques we applied a technique similar to photon mapping to obtain caustics faster. In contrast, photons are not stored as usual but are directly applied to the scene's representation.

3.2 Bidirectional Reflectance Distribution Functions

The concept of BRDFs was introduced 1977 by Nicodemus et al. [NRH+77]. For any material a BRDF describes how much light is reflected of an incoming direction with respect to an outgoing direction. Therefore the basic form depends on four parameters for the two angles in 3D. Basically a BRDF can be represented by a huge lookup table which is often the case for physical measured materials. One such table defines, according to the light, all necessary reflection information for a pure material sample. Often the BRDF is extended to further parameters as e.g. the location (spatial BRDFs).

There are many analytical models to describe certain materials with much less memory requirements like the most used basic Phong model [Pho75]. Phong used a cosine lobe ($\cos^s$) and the law of reflectance to compute the intensity of reflected light in real time applications. His model is not energy preserving but can be normalized to be plausible [Lew94]. The simple model allows the interpolation of its parameters which is necessary for our approach. Later, J. Blinn [Bli77] changed the parametrization of the phong model to increase the performance. He introduced the halfway-vector, the normalized average between incident and excident ray, to avoid the computation of the reflection vector.

The specular Cook-Torrance [CT82] model for rough surfaces makes use of micro-facets distributions to generate many different materials including metals and plastics. It is more physical related than the Phong model and pays attention to the change in the reflected color depending on the angle through the Fresnel term. For the reflection shape is uses Beckmann or Gaussian distributions.

Ward's empirical anisotropic model [War92] is based on micro-facets and gaussian distributions instead of cosine lobes. It is designed to be fit to measured materials. Like Blinn-Phong it is parametrized with the half angle between incident and excident direction.

The two layered anisotropic Ashikhmin-Shirley [AS00] BRDF is similar to Ward's model. They also introduce a non-Lambertian diffuse term to achieve energy conservation to fit measured materials better.

Many more methods can be found in a survey by Schlick [Sch94] or the more recent evaluation [MU12]. The enumerated BRDFs were considered for our approach, but none of them seemed suitable for hierarchical illumination because the reflectance characteristic of a cluster becomes unfeasible for these models fast. Further alternatives are approaches to find parametrization and compressions for arbitrary measured BRDFs by factorizing them into two or three parts [KM99, LRR04]. Kautz and McCool came to a solution which requires at least a 64x64 texture to be adequate and the factorization model from Lawrence et al. [LRR04] has a high quality while providing a fast correct sampling method, but it still has a too large and dynamical parametrization space. The generalized Phong model by Lafortune et al. [LFTG97] makes use of multiple cosine lobes. It is likely to be most suitable for hierarchies because it is able to fit any reflection pattern approximatively while having a controllable memory footprint. Unfortunately, fitting the lobes is a non-linear process which takes too much time to be done for million nodes.

3.3 Spherical Harmonics

Spherical harmonics (SHs) [ABR01] represent arbitrary functions on a sphere on a polynomial base. Nowadays they are used for example for precomputed lighting as in [Gre03] and [Slo08] which both give a practical introduction to the topic. For their evaluation Sloan [Slo13] proposed a fast method including an implementation. Spherical harmonics could be used to represent BRDF factorizations and many other direction dependent data in the scene as done in [Chr08] for diffuse lighting. Due to their high smoothness when using only a few base functions we discarded them for the use of BRDFs and for performance reasons elsewhere. The current approach does not use SHs anymore but we will discusses their use later on.

3.4 Random Number Sources

Many assumptions of the correctness of a numerical approaches depend on the quality of the (pseudo-)random number source. They must create very uniformly distributed samples while showing no patterns and having a very long period. We used a Xorshift-128 generator [Mar03] and never noticed any patterns. For parallelization it is necessary to have one generator per thread where the small size of 128 Bit is a great benefit in contrast to other generators as the Mersenne Twister [MN98] which is larger, slower and statistically similar to the Xorshift generators.

4 Hierarchical Illumination

In the following the previously explained concepts are applied to a hierarchy. The main question is: Does an illumination based directly on the hierarchy create plausible images? Thereby the main problem is to find out how to reflect light in a cluster node without accessing the underlying geometry itself. A correct solution would be possible if the reflectance of a node could be represented accurately via a BRDF. Unfortunately, this requires far too much memory and computation time (see section 4.3). For this reason several heuristics are introduced in this chapter. They are evaluated later in section 5.4.

First of all, the used elementary and hierarchical geometric representations are introduced in sections 4.1 and 4.2. Then a BRDF model is developed which is designed to require only little memory and to allow interpolation for arbitrary material combinations (section 4.3). Since the angle to the normal $\cos\theta$ used in the model is undefined for a cluster with an unknown normal, a heuristic replacement is made in section 4.4. Afterwards the reflection and sampling of cones is explained. In section 4.5 the cone cast is introduced. The tree intersection which is determined with the central ray is handled in 4.5.1. To calculate the correct integral of the incoming light in a cone the rays are re-sampled after the reflection. For computing the opening angle of the reflected cones, heuristics based on the average curvature are introduced in section 4.5.2.

Using the basic cone cast operation, the two illumination passes, ray tracing (section 4.6) and light distribution (section 4.7), are described. The first one is a modified path tracer which stores the intermediate results in the scene. Therefore the Monte Carlo integration is done per node which requires to store the sum of samples and their number N. A second modification is that caustic paths are excluded to reduce the noise. The light distribution working similarly to photon mapping adds those paths again. In contrast to photon mapping particles are neither stored in nor gathered from a map. Instead the light is stored in the hierarchy the same way as the intermediate results. This requires the estimation of a probability depending on global features. Push and pull operations are applied in both passes to fill all hierarchy levels with the lighting information (section 4.8).

4.1 Primitive Geometric Representation

The most common approach to represent the elementary geometry of arbitrary scenes is to use triangles. They can express explicit surfaces well and are the output of most modeling software. An alternative is the usage of point clouds. They are easier to handle than triangles and are used in level of detail representations such as large mesh rendering [RL00], micro rendering [REG$^+$09] and in point-based color bleeding algorithms [KTO11, Chr08, Bun05]. Some ray tracers directly support spheres or cubes as primitives, which is fast for artificial test scenes but not practical for general scenes.

Advantages of surfels over triangles: A surfel is treated logically as a point, which makes it much easier to build hierarchies on top. It is represented as an oriented disk with some material information. A bounding volume hierarchy does not need to split large or bad-shaped primitives to avoid ambiguities.
Since a point is naturally small there is also no need for textures. Furthermore there is nothing like a displacement map or any other geometrical post process once a scene

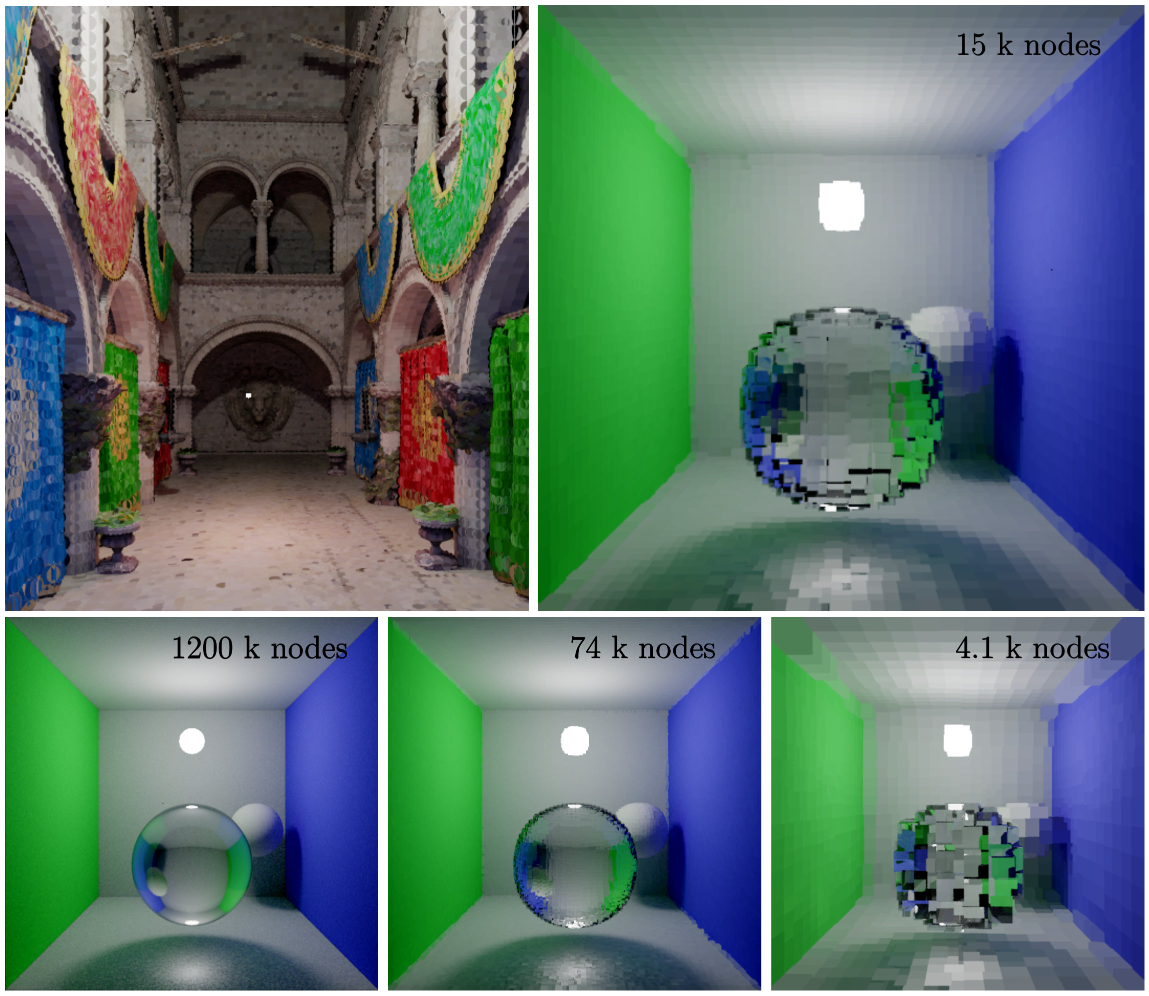


Figure 4 Scene resolution: *Top-Left*: 4.6 mio surfels are too few to represent the Crytek-Sponza; *Right and Bottom*: illumination on forced cuts of the kd-tree with different numbers of nodes

is converted into surfels. This makes all following steps easier to be implemented.
The right and bottom of figure 4 visualizes the hierarchy built on the surfel representation. The visible bounding boxes from the kd-tree are very regular and show that points can be clustered well.

Disadvantages of surfels: Most scenes are made of triangles meaning they fit the intended geometry perfectly. Contrary, the discs are always overlapping and it is difficult to ensure a hole-free surface. Additionally a surfel does approximate edges only up to its size where the representation of a disc cannot capture straight lines. Also, if the surfels are too large, the color resolution on a surface is worse than that of textures. Both reasons result in a need of a very large number of points increasing the required memory amount. This is visible in figure 4 on the left where the scene is underrepresented by an amount of surfels which takes 10 times the memory of the original scene. The noticeable dark surfels in the floor just have a different diffuse color from the texture - this is no artifact of illumination.
Another fact is that surfels must be generated from triangles first which requires an additional precomputation step before illumination is possible.

We decided to use surfels because of their simplicity. To import a scene, triangle meshes are tessellated using Phong tessellation [BA08]. This method displaces triangles on curved surfaces to better approximate smooth geometry. Then optional displacement maps are applied and the material is sampled at the respective position. Finally for each triangle a disc with the circumcircle is computed. In section 4.7.1 a problem with the estimation of the visible area, originating in the overlapping surfel discs, is discussed. Considering that it probably would have been a better choice to use the original fine tessellated triangles.

4.2 Compressing Tree Node

Both the generated surfels and the hierarchy built on them are stored the same way in a kd-tree. Using the same parametrization avoids that the illumination must consider different cases and facilitates to port the algorithm to GPU easily because everything can be maintained in arrays. The required memory per node is a crucial factor for the maximum number of elements in-core. The target is that there is enough space to capture all relevant nodes of the scene to avoid typical out-of-core loading overheads during illumination itself. Therefore the data is compressed by using the smallest possible numerical representations:

Colors Some colors like the diffuse reflectance ρ_{RGB} are always in $[0,1]^3$. These are stored in the usual 8-bit per channel format. Most other colors are HDR and have an unknown range. They are encoded in a 64-bit shared exponent format with 8-bit exponent, red and green each 19-bit and blue 18-bit mantissa. Often 32-bit shared exponent formats are use for HDR images but we require the higher precision to store intermediate results too.

BRDF-Parameters If a parameter is in $[0,1]$ it is stored in a single byte as a fixed point number (value / 255). For parameters like shininess with a higher range IEEE754-`halfs` (sign:exponent:mantissa 1:5:10) or a custom variant `uhalf` (0:6:10) are used.

Normal There are many ways to compress normals by angles or just two components. Since performance is also a concern all 3 components are kept in a 16-bit signed fixed point each.

Position The positions and bounding volumes stay uncompressed because already a small error would introduce visible gaps in the scene.

Child-References In the tree children are indexed by 32-bit values. This is sufficient because it must only be able to address all nodes in memory and not that out of core.

The full list of node information is as follows. The effects of most things are explained in the next sections. On the left side there are geometry descriptors (from `Position` till `Curvature`), followed by BRDF descriptors (till `Reflectiveness` on the right) and finally values to store the current diffuse illumination.

Data	Type	Data	Type
`Position`	$3x$ `float`	`RefractionIndex`	`uhalf`
`Normal`	$3x$ `int16`	`Refractiveness`	`uint8`
`Area`	`uhalf`	`Reflectiveness`	$5x$ `uint8`
`VisibleArea`	`uint8`	`DiffuseLuminace`	`ColorE8R19G19B18`
`Radius`	`uhalf`	`CausticLuminace`	`ColorE8R19G19B18`
`Curvature`	`half`	`ReceivedDiffuse`	`ColorE8R19G19B18`
`SelfLuminance`	`ColorE8R19G19B18`	`ReceivedCaustics`	`ColorE8R19G19B18`
`Translucency`	`ColorR8G8B8`	`NumRays`	`uint16`
`Diffuse`	`ColorR8G8B8`	`NumNewRays`	`uint16`
`Shininess`	`uhalf`	`Flags`	`uint8`

In total the node implementation takes 128 byte. Hence 8,388,608 nodes can be stored per gigabyte. The 32-bit indexing of the tree allows to address 4.3×10^9 nodes, which would result in 512 GB tree size.

Besides the static information like parametrization and position, the illumination information is stored per node (already included in the 128 byte). It consists of a double buffered color for the diffuse illumination and for the caustics. Additionally, two 16-bit counters are required to count the number of samples which are summed in the color buffers. The core parametrization without this temporary data takes 56 bytes.

4.3 BRDF Representation

On the one hand the BRDF is required to compute the fraction of light reflected into a certain direction. On the other hand it defines how rays should be sampled while doing Monte Carlo integration. Usually the BRDF is defined by the material, i.e. by textures and other input parameters. For our approach it is necessary to store the BRDF in each node individually because they all must be able to produce a likely scattering direction for the whole cluster.

Hence, the problem is that a cluster can have many different primitives with different BRDFs and parametrizations. At least for the cluster nodes there must be a function which can smoothly adapt to all underlying information. This also includes patterns which cannot be observed in normal materials as multiple reflections into different directions combined with refraction and diffuse refections in any amount. Therefore it is necessary that all elements can be interpolated independently. Despite that, the parametrization should be as small as possible because of the huge number of nodes.

One general approach would be the use of spherical harmonics. These can capture a function in respect to a single direction in 3D. It would be possible to encode a spherical harmonic inside another one or to use a four dimensional generalization. This way a function which depends on two directions can be encoded. The disadvantages are the memory consumption (nested: 81 floats for a 3-band SHs base) and the low precision. It would only be possible to store very smooth reflection patterns.

As stated in section 3.2 there are models to express general BRDFs using factorizations. The methods from [KM99] and [LRR04] are fairly general but need space in the KB range.

Using the Lafortune model [LFTG97] with a medium count of cosine lobes could be fit to the cluster's reflection. Here the problem is the large amount of nodes which must be fit. For each of million nodes a non-linear optimization must be done which takes too much time. After a fitting on the leaf level the parameters of two children cannot be simply interpolated if they have different normals. Thus each cluster must be computed individually.

For the reasons of space and computation time we used a simpler, less expressive custom approach. The idea is to make as much as possible implicitly computable and to derive a variance for clusters for different kinds of reflections. The solution proposed in the following is interpolable as long as the surface normals are relatively similar. Afterwards it tries to create reflections heuristically.

Like in other BRDF models there are certain physically plausible reflections which can be modeled independently. We considered three possibilities: total reflection at the surface normal, refraction and diffuse characteristic.

4.3.1 Specular Fraction

As stated above, we modeled reflections and refractions at the normal only. Effects like retro-reflection are neglected so far. To reduce the number of parameters, the directions are computed from the node itself (section 4.4) and the BRDF just controls the probability for the three cases including the diffuse term.

The probability for a reflection is stored in a 1D-lookup table function $p_{\text{reflect}}(\theta)$, which depends on the angle θ between incident ray and normal. This enables materials to have a Fresnel reflectiveness increasing for grazing angles. Expressed by the function is the pure probability of a reflection, it does not define into which direction this will happen. To determine that, the law of reflection is used. The probabilities for refraction and diffuse terms are derived from this function:

$$p_{\text{refract}}(\theta) = p_{\text{refractiveness}} \cdot (1 - p_{\text{reflect}}(\theta)) \tag{4.1}$$

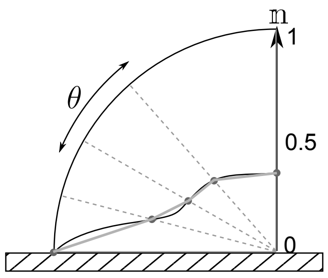
$$p_D(\theta) = 1 - p_{\text{refract}}(\theta) - p_{\text{reflect}}(\theta) \tag{4.2}$$

The scalar value $p_{\text{refractiveness}} \in [0, 1]$ is an additional parameter for the transparency. For (partially) transparent materials the fraction of refracted light is modeled as a constant part. So the remaining non reflected light $(1 - p_{\text{reflect}}(\theta))$ is distributed among the refractive and diffuse terms. All three together add to one so the energy is preserved. Absorption is part of the reflectance in the diffuse term.

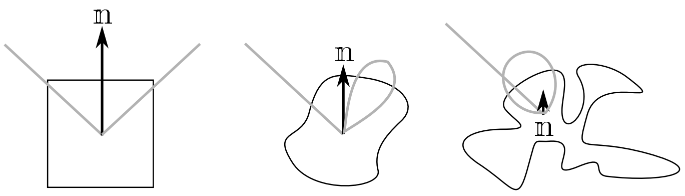
Figure 5 (a) shows how $p_{\text{refract}}(\theta)$ is encoded in form of a lookup table. The gray line is the bilinear interpolated result of the stored samples. Instead of regular angles, the $\cos\theta$ is used to compute the indices in the table. This leads to an increased resolution toward flat angles decreasing the error for Fresnel like reflections. We computed the relative error $\int_0^{\pi/2} |f(x) - g(x)|/f(x)dx$ in respect to a Schlick-Fresnel [Sch93] approximation and a rescaled Schlick-Fresnel version [LSK05] with different parametrization.

Samples	2	3	4	5	6	7
Error to Schlick	10.8%	2.68%	1.12%	0.646%	0.409%	0.261%
Error to Rescaled Schlick	61.8%	4.41%	1.35%	0.503%	0.333%	0.214%

Hence, we used 5 samples with one byte each because less than 1% deviation seemed to be acceptable. The error table was computed from this byte-compressed form so the precision related deviation is already contained. An advantage of the sampled version is that it can be interpolated smoothly between nodes. It is also able to represent other materials with different probability distributions up to the sampling resolution,



(a) Polar plot of encoding of $p_{\text{reflect}}(\theta)$, the probability of a reflection for different incident angles with respect to the normal $\mathbb{n}$



(b) Influence of cluster normal $\mathbb{n}$. The more random a cluster is the less sharp the reflection. Additionally, it becomes a retro-reflection because the average expected normal shows into the view direction

Figure 5 Reflection model encoding and normal heuristics

whereas it is representative for a single frequency of light only. This is intended and also true for most other models.

Figure 5 (b) shows the reflection direction itself, dependent on the roughness of the cluster. The function $p_{\text{reflect}}(\theta)$ scales this reflection term but does not influence its shape. This is going to be explained later in section 4.4.

4.3.2 Diffuse Fraction and Interpolation

The diffuse reflection is parametrized by a single color $\rho_{RGB} \in [0,1]^3$, as usual in the Lambertian diffuse model. A number smaller than one will cause a loss of energy due to the absorption of the material.

A hierarchical approximation for the diffuse and the specular parametrization can be computed by an area weighted average. The area is a good measure for the probability to choose the child node's BRDF when computing the reflection for a cluster from a random direction. A better method would be to consider the projected area instead, but this would make all parameters dependent of the angle. E.g. assume two perpendicular faces which get clustered. In this extreme case the real hit probability interpolates from one for the first face, to one for the second face depending on the angle. In consequence each single parameter would change with the angle which would lead to the need of spherical functions and much more memory per node. So computing a single average value based on the areas is the best way to pay attention to the nodes' importances.

4.4 Hierarchical Approximation of Light Reflection Directions

All the computations above assume a single normal for the calculation of the reflection pattern (diffuse + specular). Such a simple normal is not present when different surfels are clustered (see figure 5 (b)). Again, an agglomerative normal can be created from the area weighted average of children. The trick is not to normalize the result. If normals point in different directions the average of them becomes shorter than one. Hence, the normal length $|\mathbb{n}|$ itself is a parameter for the smoothness of the surface.

Knowing a roughness the reflection direction can be modified heuristically (adding diffusion) to approximate the statistic effect of choosing a random child. For any calculation which uses angles between incident/excident ray and surface normal the

$\cos\theta$ term must be exchanged. As long as $|\mathbb{n}| = |\mathbb{d}| = 1$, where $\mathbb{d}$ is a general direction vector, equation 4.3 holds true per definition of the scalar product.

$$\cos\theta = \langle \mathbb{n}, \mathbb{d} \rangle \tag{4.3}$$

$$\begin{aligned} \cos\theta &\approxeq (1 - |\mathbb{n}|) \cdot \frac{1}{4} + |\mathbb{n}| \cdot \frac{\langle \mathbb{n}, \mathbb{d} \rangle}{|\mathbb{n}| \cdot |\mathbb{d}|} \\ &= (1 - |\mathbb{n}|) \cdot \frac{1}{4} + \langle \mathbb{n}, \mathbb{d} \rangle = \Theta \end{aligned} \tag{4.4}$$

The heuristic Θ is a linear interpolation between the correct term for normals with length 1 and a constant. The interpolation factor is the length of the normal and partially hidden in the scalar product. The constant $\frac{1}{4}$ describes the amount of diffuse reflection invariant to the direction. It is derived from two factors: the visible area and the expected value of $\cos\theta$. If the normals are completely random, the probability to see a front face is $\frac{1}{2}$. Furthermore, the expected value of the uniformly distributed $\cos\theta$ is:

$$\mathcal{E}(\cos\theta) = \frac{1}{2\pi} \cdot \int_0^{2\pi} \int_0^{\pi/2} \cos\theta \cdot \sin\theta \cdot d\theta d\phi = \frac{1}{2} \tag{4.5}$$

Together the front face visibility and the expected $\cos\theta$ yield the constant 1/4. The precondition for formula 4.4, that there are many faces with a uniform distribution, does not hold true if there are only few opposing faces. An alternative approach would be to store the density of normals in spherical harmonics. One possible way to do so is to use the projected area, which is also a measure of how many faces are oriented in which direction. A sampling or even integration would yield a better estimation than the derived constant. Nevertheless, the constant alone already reduces the error during illumination as shown in section 5.5.

Now Θ can be used to replace the terms in the illumination equations. Let $L_D(\omega_i)$ be a sample of the Monte Carlo sum for the diffuse illumination. $(x)_+ \mathrel{\widehat{=}} \max(0, x)$ is clamping the negative values if the light approaches from the backface.

$$L_D(\omega_i) \approxeq \frac{\rho_{RGB}}{\pi} \cdot (\Theta)_+ \cdot L(\mathbb{x}, \omega_i) \cdot \Delta\omega_i \tag{4.6}$$

For specular reflections we use a modified Phong model with a cosine lobe. The roughness influences the shininess (exponential coefficient s), the angular probability $p_{\text{reflect}}(\theta)$ and the reflection direction itself. To sample the probability function the same approximated cosine term Θ is used. The difference is that the dot product's absolute value is taken, because in transparent objects the ray can come from the inside (backface). We modified the values for the exponent and the reflection direction heuristically. In the following equation the reflection direction $\mathbb{r}$ is computed with the law of reflection from the normal and the ingoing direction $\mathbb{d}$.

$$s' = s \cdot |\mathbb{n}| + 1 \cdot (1 - |\mathbb{n}|) \tag{4.7}$$

$$\mathbb{r}' = \mathbb{r} \cdot |\mathbb{n}| - \mathbb{d} \cdot (1 - |\mathbb{n}|) \tag{4.8}$$

Through the interpolation the shininess s' goes towards one for disappearing normal cases. Then the lobe becomes a perfect sphere which is a maximal directed diffuse reflection. Also, the reflection direction is interpolated towards the negated incident direction. This is likely because the average normal, when looking at a diffuse cluster, shows into the direction of the observer. Hence, the reflection direction tends to the diffuse-retro-reflection. In the degenerated case where $\mathbb{n}$ becomes $\mathbb{0}$ the direction $\mathbb{r}$ does not need be computed because it is multiplied with 0.

Using the two modified values the importance sampling is done as usual for the phong model. First two angles are created from two independent uniform random samples ξ_0 and ξ_1.

$$p(\theta, \phi) = \frac{s' + 1}{2\pi} \cdot \cos^{s'} \theta \tag{4.9}$$

$$\Rightarrow \qquad \theta = \arccos\left(\xi_0^{\frac{1}{s'+1}}\right) \tag{4.10}$$

$$\Rightarrow \qquad \phi = 2\pi \cdot \xi_1 \tag{4.11}$$

Finally these two angles are transformed into a vector in the tangential space defined by $\mathbb{r}'$ and a rejection sampling takes place if the sample falls in the half space of the surfel which is showing away from the normal.

4.5 Cone Casts

Ray casts are a very basic operation to obtain an incoming luminance, test visibilities or shoot photons. Starting at some point, it is the goal to find the closest intersection on the ray as fast as possible. It is a common procedure to use hierarchies (see e.g. [AK89]) to increase the performance of such a query. For out-of-core approaches a ray might be delayed or must be discarded if the required part of the hierarchy is not in memory. The idea of this thesis is to use cones instead of rays and to accept higher level nodes as correct intersection targets. This makes the finer geometry levels unnecessary and avoids that they need to be loaded. Additionally, using cones can increase the performance because of the geometry reduction. In contrast to [Ama84] we use the cone radius for the choice of the LOD only and do all intersection tests and tracings with the central ray.

In general the idea of cone casts is to collect the lighting information from a larger solid angle with a single operation. Starting at the eye rays are cast through a pixel. A pixel already has some extend, so for each pixel there is a thin cone. When this cone hits distant geometry a whole object may fall into the same pixel. It is not important to compute the whole lighting within the object as long as the final pixel color remains the same. Furthermore, if a cone cast would be stable and reliable for extreme cases, a diffuse reflection could be seen as a cast with a 180° opening angle. This would speed up the indirect illumination much. In our implementation the maximum cone opening angle is at about 30°.

One of the main difficulties is that the cone can diverge to arbitrary complex patterns. Figure 6 shows three different options how a cone might be reflected by the scene. It is also possible that the light paths split into different directions and the shape becomes unfeasible. Ignoring these cases does not cause severe problems, as long as cones are thin or their interior is sampled stochastically as is shown later on. In our implementation the idea of a cone is changed to a frustum. In general, each of the frustums could be extended to a cone which starts somewhere behind or in front of the surface. However, the range between virtual origin and surface is not of interest for the lighting.

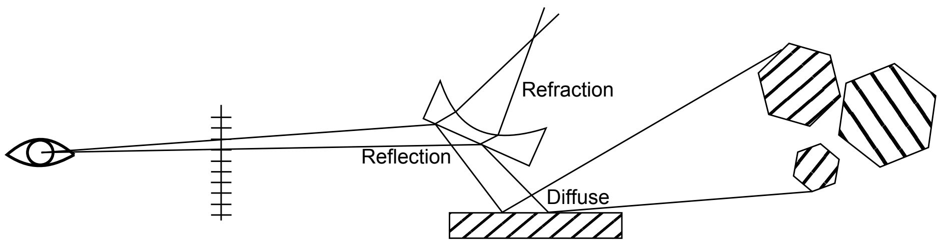


Figure 6 Different possibilities of reflection for cone paths. The primary opening angle depends on the pixel size, then reflections and refractions can widen, lessen or invert the angle. A diffuse reflection causes a 180° angle which leads to degenerated cones so it is only widen again.

Our first approach was to do a micro rendering [REG+09] for each cone/frustum. This means the scene is be rendered to a small framebuffer for the cone using the hierarchy and then integrated afterwards to get the incoming intensity. Seeing multiple objects in a frustum would deliver a correct result up to the chosen buffer resolution. The rendering had several problems for the targeted usecase:

- **Specular recursion**: For each object in the frustum, if not diffuse, there is a further cone necessary to determine the specular reflection. This causes an exponential increase in the ray count. The recursive illumination could also be solved by choosing only one object in the frustum probabilistically leading to more noise, but particular specular reflections cause strong noise anyway. An alternative would be to store the reflection in the nodes in form of a spherical function. Unfortunately this consumes much memory and any compression of this function increases the error. E.g. with SHs it would not be possible to render mirrors this way.
- **Performance**: Micro rendering is capable to determine the whole integral over a hemisphere relatively fast, but for rendering the scene for each ray cast it is still quite slow.
- **Appropriate level of detail**: For the rendering itself surfels or other approximative shapes for the objects are required. The more details are used, the more precise the result, but the more nodes must be loaded and illuminated too. It would be best to chose a level where nodes have roughly the same size as the cone radius, but then a rendering does not have much of an advantage anymore, because each surfel would cover many pixels. In this case an integration over the rendered buffer does not reduce the error with respect to a few randomly sampled rays much.

Especially, because of the non-feasibility of specular reflections and also for the other reasons we discarded micro rendering.

4.5.1 Stochastic Cone Casts

Instead of micro rendering we used a simpler heuristic in the end. A cone is treated like a simple ray. So, there is exactly one intersection point between the ray and the scene. The difference to ray casts is that the selected scene LOD changes with the cone radius. For the primary cones from the point of view this performs like a distance

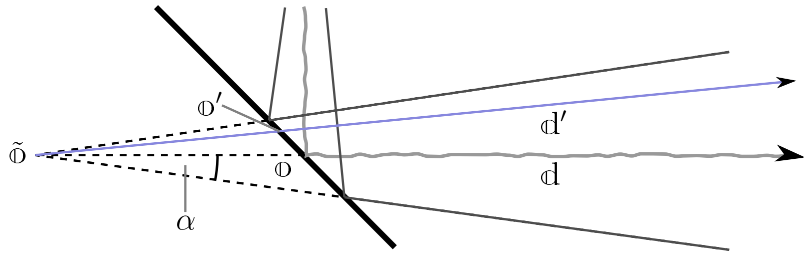


Figure 7 Probabilistic re-sampling of a cone. The jittered ray is the original light path, $\mathbb{o}$ the new origin of the ray, $\tilde{\mathbb{o}}$ the virtual origin of the reflected cone and $\mathbb{d}'$ the randomly resampled direction

based LOD. The radius itself is used in the heuristic which choses the approximation level:

$$r_{\text{cone}}(t) > \max\left(\text{BB}_{\text{sides}}\right) \tag{4.12}$$

Where $r_{\text{cone}}(t)$ is the cone radius at the distance to the ray origin t and BB_{sides} is the set of the three side lengths of the node's bounding box.

The bounding box turned out to be a better choice than the summed area in the node or some other geometry, like an oriented disc. This happens because at some points, in the hierarchy geometry from very different locations is fused. The maximum of the bounding box sides gets large if at least one dimension clusters nodes which are far apart. This measure guaranties that all geometry of the node fits into the cone independent of the view direction. Even when hitting the border of a node, the associated geometry must be inside because the whole node is smaller than the radius. Using coarser approximations introduces visible artifacts.

If rays are chosen randomly inside the cone, all objects lying inside will be hit with a probability proportional to their projected size. Assuming that a node always contains a good approximated BRDF, a standard path tracing can be performed this way. The difference is that the scene is discretized to a resolution which matches the importance of the scene for the current view. Therefore at least the diffuse light can be stored in each node to reuse the recursive results during tracing of other rays.

Assuming that the chosen LOD is the input scene representation, the illuminance (referring equation 2.8) can be computed unbiased. Therefore the casting direction must be randomly sampled, according to a uniform probability density function, inside the cone. This way all objects intersecting the cone are sampled relative to their coverage area. The luminance $L(\omega_i)$ returned from each cast is multiplied by the solid angle and the $\cos\theta$ of the cone. With these samples a Monte Carlo integration can be done to get the unbiased expected value for the integral of incoming illuminance.

$$\mathcal{E}(E) = \frac{1}{N}\sum_{i=1}^{N} \frac{L(\omega_i)\cdot\cos\theta\cdot\Delta\omega_i}{p(\omega_i)} \tag{4.13}$$

So each time a cone is casted its direction must be re-sampled once to generate a random variation inside the cone. Doing that, the cone casting itself does not introduce a new error. For the uniform sampling we generated two independent angles $\theta \in [-\alpha, \alpha]$ and $\phi \in [0, 2\pi]$ similar to a sphere where α is the half opening angle of the cone.

$$p(\theta, \phi) = \frac{1}{\Delta\omega_i} \tag{4.14}$$

$$z = (1 - \cos\alpha) \cdot \xi_0 + \cos\alpha$$

$$\Rightarrow \quad \theta = \arccos(z) \tag{4.15}$$

$$\Rightarrow \quad \phi = 2\pi \cdot \xi_1 \tag{4.16}$$

This yields a vector $(\sqrt{1-z^2}\cos\phi, \sqrt{1-z^2}\sin\phi, z)$ which is uniformly distributed in the correct solid angle but must still be transformed into the cone coordinate system. First the direction must be rotated and then a new origin must be computed, because the virtual origin $\tilde{\mathbb{o}}$ is rotated too. Since we use a frustum the used origin is not at the cross section of the cone, but at $\mathbb{o}'$. Note that modifying the direction alone and still starting the ray at $\mathbb{o}$ would change the cone's virtual origin which is wrong. Figure 7 shows the resampling of an intermediate step during ray tracing.

4.5.2 Maintaining the Frustum during Reflections

As was stated in Figure 6 calculating good cone radii can be very difficult. Further, the origin can lie on any side of the surface and we are searching for the intersection closest to the surface without hitting the surface itself.

In contrast to a cone, a frustum includes a near and a far plane which can be set to avoid self-intersections with the origin. The far plane can be used for optimization of visibility casts where the distance to the target is known. The scene behind does not need to be tested.

We parametrized the frustum through `Position`, `Origin`, `Radius0` (r_0), `TanAlpha` ($\tan\alpha$) and `FarPlane`. The `Origin` of the frustum is the point on the surface where the ray cast starts. Thus a near plane is not necessary. Instead `Radius0` determines the radius of the frustum begin. `TanAlpha` is the tangent of the half opening angle as is visible in 8.

To cover the case in refractions where the frustum might have a cross-section, the radius at the near plane and the $\tan\alpha$ are saved instead of two radii. For each ray distance parameter t, a positive radius can be computed, as well as a new tangent when starting a ray at this point. The function of the radius is implemented to be always positive since this size is used in the heuristic as measure for the required node size. A possible cross-section has no relevance for our method. When starting a new ray behind a cross-section the sign of the tangent must be inverted. Figure 8 visualizes the three important cases. With that the radius and the tangent can be expressed as:

$$r_{\text{cone}}(t) = |r_0 + \tan\alpha \cdot t| \tag{4.17}$$

$$\tan\alpha(t) = \begin{cases} |\tan\alpha| & \text{if } -r_0/\tan\alpha < t \\ -|\tan\alpha| & \end{cases} \tag{4.18}$$

The initial angle is calculated from the camera parameters, near clipping plane and opening angle in x-direction ψ, and the screen's resolution $w \times h$ where we assume quadratic pixels.

$$\tan\alpha_0 = \frac{\text{near} \cdot \tan\psi}{w} \tag{4.19}$$

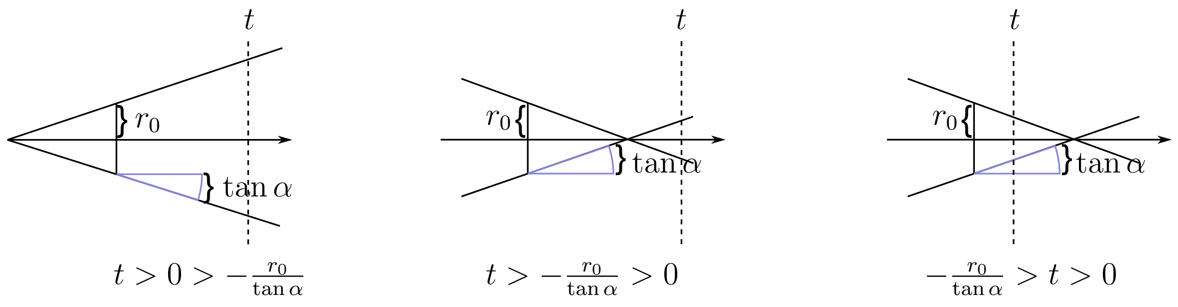

t
r_0
$\tan\alpha$
$t > 0 > -\frac{r_0}{\tan\alpha}$
t
r_0
$\tan\alpha$
$t > -\frac{r_0}{\tan\alpha} > 0$
t
r_0
$\tan\alpha$
$-\frac{r_0}{\tan\alpha} > t > 0$

Figure 8 Cases for $r_{\text{cone}}(t)$ and $\tan\alpha(t)$.

When creating reflected frustum rays, the following heuristics are applied to change the opening angle of the cone depending on the kind of reflection. Section 5.4 evaluates the influence of this heuristics which is relatively small.

Diffuse A diffuse reflection always increases the opening angle because in theory a single cone with 180° would be correct. A larger cone decreases the precision but improves the performance due to the earlier usage of coarse LODs, but for too large radii there is no meaningful cluster, so the angle must be clamped to a smaller fixed size.

$$\tan\alpha' = d \tag{4.20}$$

We found $d = 0.3$ ($\alpha \approx 16.7°$) experimentally to show no noticeable artifacts.

Specular reflections Any mirror can enlarge, decrease or invert the opening angle depending on its curvature. Here ray differentials [Ige99] could be used instead. This would increase the reliability but also needs more memory since the whole tensor must be saved.

As a heuristic an offset can be added to or subtracted from the previous $\tan\alpha$ to compute a new angle for the outgoing ray. Thereby the sign depends on the curvature direction in respect to the ray. If the surface bends away the factor must be positive, otherwise negative. We use the signed average curvature κ_s to describe the surface with respect to its normal. Since a ray can approach from the backface the sign must be inverted conditionally. The cosine between ray and normal is a reasonable factor to get a smoothed version of this sign which means in near perpendicular cases the ray just keeps its angle.

$$\tan\alpha' = \tan\alpha \cdot (1 - \kappa_s \cdot b \cdot \cos\theta) \tag{4.21}$$

b is a control constant which is necessary to scale the average curvature κ_s to avoid a too fast change of the rays. In section 5.4 this parameter is evaluated to be good in the range $[0.5, 1]$.

Refractions A lens behaves the same way as a reflecting surface but the sign must be handled in another way. Depending on the direction the quotient of the refraction indices is inverted. The angle still depends on the curvature direction but additionally on the material properties.

$$\tan\alpha' = \tan\alpha \cdot (1 + \kappa_s \cdot (n_1 - n_2) \cdot c \cdot \cos\theta) \tag{4.22}$$

c is again a control constant which we set to 1.0 after the testing in section 5.4.

The estimation of the average curvature κ_s is calculated from the triangles during the conversion to surfels. In [Rus04] the second fundamental curvature tensor is computed from a single triangle. Since there is no need for an adjacency this method can be used to derive the scalar value during surfel generation. Instead of computing the tensor, where only a scalar is required, we did not solve the equation system but averaged the intermediate results directly:

$$\kappa_s = \frac{1}{6} \sum_{(i,j)\in E} \frac{\langle \mathbb{n}_i - \mathbb{n}_j, \mathbb{u}\rangle}{\langle \mathbb{p}_i - \mathbb{p}_j, \mathbb{u}\rangle} + \frac{\langle \mathbb{n}_i - \mathbb{n}_j, \mathbb{v}\rangle}{\langle \mathbb{p}_i - \mathbb{p}_j, \mathbb{v}\rangle} \tag{4.23}$$

Where $E = \{(2,1),(0,2),(1,0)\}$ is the set of edge indices, $\mathbb{n}_x$ the normals at the vertices, $\mathbb{p}_x$ the vertex positions and $\mathbb{u}, \mathbb{v}$ the tangent space directions. For a sphere this average resulted in a value close to $\frac{1}{r}$ (less than 1% deviation) which is the mean curvature of a sphere.

One last problem is the intersection of a cone with the originating surface. It is possible to offset the frustum but, due to the heuristic which choses all nodes inside r_0 this can become relatively large, hence thin objects could be skipped this way. Masking the source node itself also does not lead to a solution because on finest level the surfel overlap to a high degree such that a masking of the previous hit does not exclude enough surfels. Instead, the origin must be outside each node while searching in the hierarchy to take that node into account as possible intersection target. If this fails the children are observed or if it is a leaf it is rejected. This works similar to masking but also covers overlapping neighbors. It still does not solve the problem entirely, as visible in the experiment from section 5.7

4.6 Using Cones for Eye Casts

Cone casts are used in two directions to illuminate the hierarchy in our approach. Once from the point of view (eye casts) and once from the light sources to distribute light. The combination of both introduces a bidirectionality which utilizes the strength of both and is able to generate caustics faster. Each of the two variations uses the discretization of the hierarchy to store results and has to face different problems.

The idea behind eye-casting on the scene is a standard path tracer [Kaj86]. As stated in section 4.5 a probabilistic choice of the cone directions improves the correctness of the results with respect to other cone casting. Still the approximations of the BRDFs and the geometry introduce a large bias. Usually a path tracer only averages the ray results on the target image itself. Contrary we solve the integral for the diffuse illumination for each node on the path in parallel.

The Monte Carlo integration (see 2.3.1 for details) assumes that the sample $f(\omega_i)$ is the correct function value. In a recursive path tracer this sample is again an expected value from the recursive call, determined by a further sample divided by the probability for the chosen direction. The performance can now be increased by solving the formula (2.18) for each node on the path and returning the expected value instead of a single weighted sample. To do so the number of samples N must be counted for each node independently because it depends on the number of rays passed through the node.

It is not possible to store all relevant information in each node. Light which is reflected specularly depends on the view direction and cannot be saved that easily, because rays can come from all directions. Again spherical harmonics could be introduced, but they consume too much memory and are too smooth to be effective.

4.6.1 Ray Counting

When using importance sampling, the number of rays increases very quickly for certain nodes, e.g. that of the light sources. In the Cornell box scene (figure 9) a first 32-bit counter overflows after a few minutes. Using a 64-bit value would suffice but would also needs more space.

We used a 16-bit counter and build in a safeguard against overflows. A node which reached the maximum is tagged as converged. The recursion of diffuse samples is stopped when reaching a converged node. Specular paths through this node are traced as usual. The evaluation in section 5.8 demonstrates that even a small number of stored diffuse samples ($\approx$ 20 to 100) is relatively noise free. So a threshold of 65636 rays is sufficient. This relatively small number also reduces the iteration duration over time because rays can be stopped earlier inside well sampled regions.

4.6.2 Importance Sampling and Excluded Paths

The importance sampling of the BRDF has to distinguish between specular and diffuse reflections. The diffuse part is accumulated in the node and the average of all passed rays is returned. Additionally, a specular sample L_S including refractions is taken and returned too. Both parts are weighted by the probabilities for diffuse p_D and specular p_S reflections respectively. Let L_D be the accumulated light in a node and $L(\omega_i)$ be the luminance sample returned by the node.

$$L'_D = L_D + L_D(\omega_{\text{rnd}}) + L_{D_{\text{light}}} \tag{4.24}$$

$$L(\omega_i) = L_e + p_S \cdot L_S + p_D \cdot \frac{L_D}{N+1} \tag{4.25}$$

For each diffuse sample two rays are casted. One in the direction of a randomly chosen light source $L_{D_{\text{light}}}$, which has a recursive depth of zero, to compute the direct lighting. The other one is randomly sampled according to the diffuse BRDF ($L_D(\omega_{\text{rnd}}) \mathrel{\widehat{=}}$ function 4.6) excluding the direction of the light. This exclusion in the random sample is important to avoid an overestimation of the light from a certain direction because both samples are added (two independent light samples contribute to the illumination additively). The importance sampling of the light source is done by the solid angle from the origin to all possible sources and their intensity. Then the direct illumination is computed from the solid angle $\Delta\omega_o$, the probability to choose the light source p_L and the diffuse BRDF (again with reference to equation 4.6):

$$\begin{aligned}
L_{D_{\text{light}}} &= \frac{L_D(\omega_i)}{p_L} \\
&= \frac{\rho_{RGB}}{\pi} \cdot (\Theta)_+ \cdot L(\mathbb{x}, \omega_o) \cdot \frac{\Delta\omega_o}{p_L} \\
p_L &= \frac{\Delta\omega_o \cdot I_o}{\sum_{l=1}^{m} \Delta\omega_l \cdot I_l} \\
L_{D_{\text{light}}} &= \frac{\rho_{RGB}}{\pi} \cdot (\Theta)_+ \cdot L(\mathbb{x}, \omega_o) \cdot \sum_{l=1}^{m} \frac{\Delta\omega_l \cdot I_l}{I_o}
\end{aligned} \tag{4.26}$$

Due to the importance sampling of the BRDF, many terms can be canceled when computing the random diffuse sampling.

$$\begin{aligned}
p(\theta, \phi) &= \frac{\cos\theta}{\pi} \\
p_D(\theta, \phi) &= \frac{(\Theta)_+}{\pi} \\
L_D(\omega_{\text{rnd}}) &= \frac{\rho_{RGB}}{\pi} \cdot (\Theta)_+ \cdot L(\mathbb{x}, \omega_o) \cdot \frac{\Delta\omega_o}{p_D(\theta, \phi)} \\
&= \rho_{RGB} \cdot L(\mathbb{x}, \omega_o) \cdot \Delta\omega_o
\end{aligned} \tag{4.27}$$

The exclusion of the direct illumination, which is already estimated by $L_{D_{\text{light}}}$, is done by masking the emitted luminance L_e in the recursion of $L(\mathbb{x}, \omega_o)$ for the next hit.

The reflection/refractive L_S is traced straight forward and weighted by $p_S = 1 - p_D$. There are still two alternatives: Either one can trace only one of the two by a random choice or both of them. In general a picture cannot get noise free if not each possible path was considered at least once. Additional random choices increase the variance and the implementation showed to be comparably noisy because specular reflections are not stored and reused. Finally, we decided to trace both even if this means a faster growth in the ray count.

4.7 Light Casts and Distribution

Certain effects like caustics are very noisy when created probabilistically by a unidirectional ray cast from the observer. Furthermore, scenes where the light source is not or barely visible and the main influence comes from indirect lighting are difficult and converge slowly. By adding a bidirectional component, the importance of light sources is increased and the noise on the image is decreased in general. Many other approaches like bidirectional path tracing [LW93, VG95] and photon mapping [Jen96] are using this fact.

We introduced a new method similar to photon mapping relying on the hierarchy. The idea is to illuminate a node which was hit by a photon directly instead of storing photons and gathering them later.

To allow an interleaved calculation of eye and light casting passes as well as to make the illumination pass optional, we divided the light into several components. All paths LD(S*D*)* beginning with a diffuse reflection are added directly to the results of the eye-pass. All caustic paths with at least one specular reflection LSS*D are stored in a second independent value. The same paths which end with a specular reflection (DS*SL) are masked out in the eye pass.

Light is distributed from each self-illuminated surfel independently. The number of rays N is determined by the surfel's brightness and a scalar input parameter for the number of photons. The intensity of each ray depends on the intrinsic luminance L_e and the area A of the source surfel:

$$I_r = \frac{L_e \cdot A}{N} \tag{4.28}$$

The rays are traced from the light through the scene and add their illumination directly to the hit surfels. At each hit surfel $\mathbb{s}$ we want to compute the expected value $\mathcal{E}$ for the diffuse illumination. This is again done through Monte Carlo integration. Therefore we need the probability p_{hit} that the target surfel was hit, which depends on global features. If only rays which hit surfels are counted the rays which miss the surfel would never be considered. A target which is only seldom hit would still have the maximum illumination because each ray which intersected with it would do so at full intensity. Similar to the gathering step of photon mapping it is necessary to estimate the number of photon hits in the direct environment to get this value.

$$\mathcal{E}(I(\mathcal{S})) = \frac{1}{M} \sum I_{r_n} \cdot p_{hit} \tag{4.29}$$

A possible solution is to count the light passes M globally. Instead of letting the tracing of light itself increase the counter in the surfel $\mathcal{S}$, the counter is increased exactly once per pass for all nodes. In the formulation above i.e. to set $p_{hit} = 0$ when $\mathcal{S}$ is not hit in this pass and $p_{hit} = 1$ if it is.

Semantically this means that we distribute the energy of all sources once and assume that everything got its appropriate portion of it. For an infinitely large count of photons and infinitesimal small surfels this assumption is true because every surfel accumulates exactly as many energy portions as probable. For a finite amount of photons the result contains noise which can be reduced by further light casting passes. As mentioned in the second assumption the area of surfels must be considered too. A node which has twice the area of another one is hit twice as often. Thus the illumination at a target is

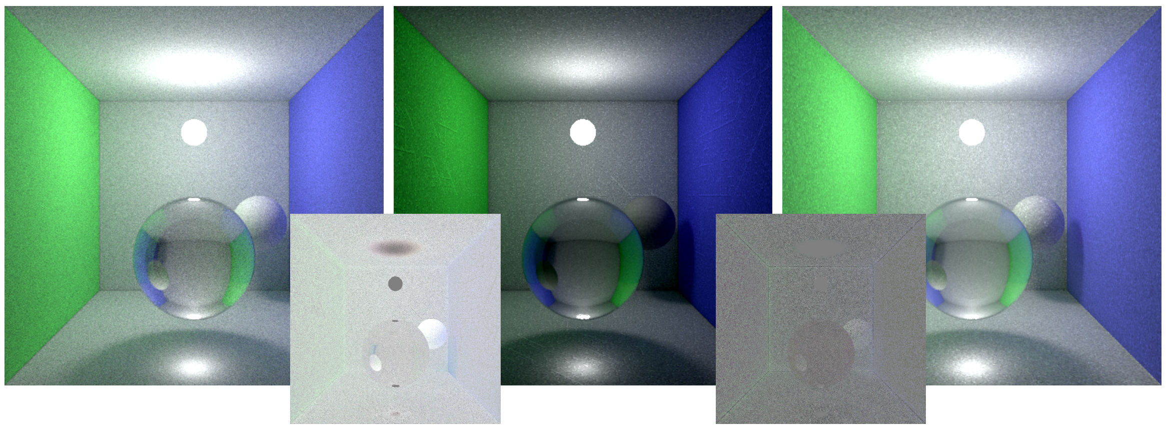
$$L'_D = L_D + \frac{I_r}{A'} \cdot \rho_{RGB} \cdot p_D \tag{4.30}$$

where ρ_{RGB} is the diffuse reflectance of the node and p_D the probability to be reflected diffus which is one minus the probability of a specular reflection for the incoming direction. The term A' is a modified area term with $A' \leq A$ which is necessary because the area of the geometry A is not that which is visible due to the representation. This is shown in the next section.

4.7.1 Visible Area Correction

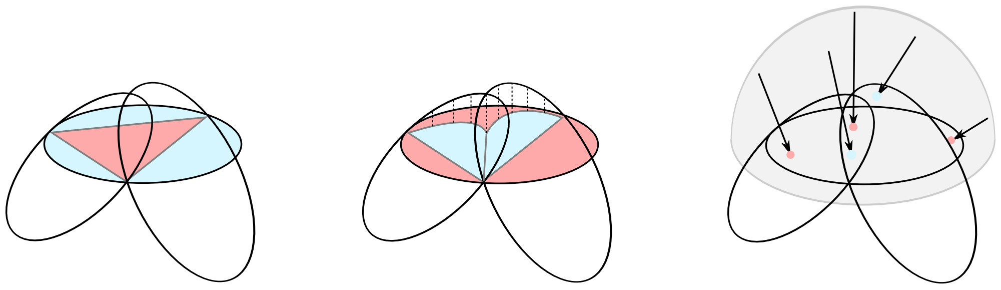
Using the area A from the input geometry (triangular) causes a severe underestimation of the illumination. The same happens when using the area of the surfel-disc itself as shown in figure 9. Moreover this error is variable for different objects. Most obvious the smaller sphere's illumination is underestimated stronger than the other parts (see error image 9). This is caused by a self shadowing of the adjacent surfels (see figure 10). Thus the estimation of the hit-probability is overestimated. Also there are two more artifacts. The surfels at the box's corners are a little bit too dark and there are "scratches" on the wall (most noticeable on the blue one). In picture (c) a corrected area is used which estimates the real area visible by random rays. The corners get now slightly too bright but the overall error is reduced effectively.

The estimation of the actually visible area A' is done as a precomputation step after the scene tree is built. For each leaf node a number of rays is casted towards the surfels from a short distance. The number of hits divided by the number of casts accounts for the shadowing from adjacent surfels. For ray generation we generated a random position and direction as follows. The direction is chosen as the negative normal of the surfel plus some noise (we used a uniformly distributed random direction to recognize all directions). The noise is very important to get a universal area estimation. Later



(a) Path traced reference 9h[1] **(b)** sing the original surface area yields a wrong illumination (20min[1]) **(c)** The area correction term reduces the errors by far, but not completely (20min[1])

Figure 9 Pure light distribution pass, Cornell box, 0.75 mio surfels. The small pictures are the signed error images.



(a) Real triangle area used in the left image of figure 9 **(b)** Projected (from top) visible surfel area (red) has nothing to do with the original triangle **(c)** Brute force estimation of the average visible area by counting hit points

Figure 10 Visible area estimation

the photon rays will also come in from random directions. Then the origin is placed on the surfel disc and translated in the negative ray direction.

Still, the light cast pass is not performing correct results. This is further evaluated in section 5.6 and 5.7. A valid solution could be obtained when the real projected area is computed for each incoming ray, but this consumes too much time. Note that we did not test a triangle based version (no surfel based geometry), which could work fine.

4.8 Push-Pull Subpasses

Diffuse illumination is stored in the scene similar to hierarchical radiosity from both illumination passes. To avoid a convergence towards a wrong value the results are double buffered and must be applied to the node's outgoing radiosity in an extra pass. The double buffering during ray casting itself avoids that later cast rays get more

[1]The test system for this and any other benchmarks is a notebook with: Intel Core i5-3210M @ 2.5 GHz (all 4 hardware threads used), 8GB RAM, sometimes running compilers simultaneously

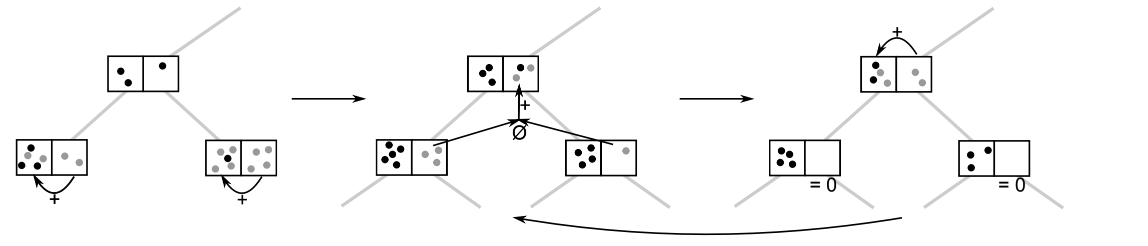


Figure 11 Diffuse pull after eye-pass. Each node contains the diffuse color and a buffer for received values (right box), the gray points are the samples which get applied in the respective step

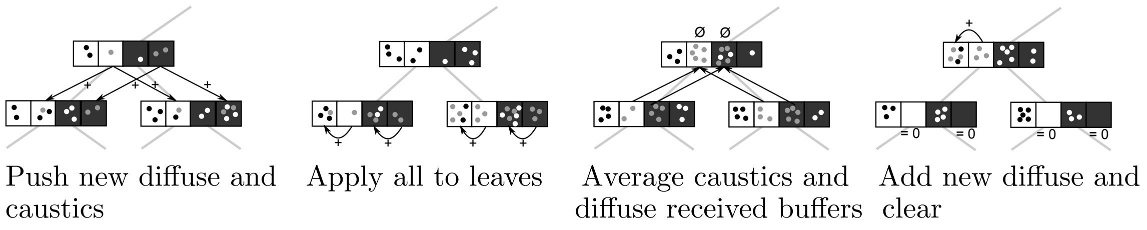

+
Ø
= 0
Push new diffuse and caustics
Apply all to leaves
Average caustics and diffuse received buffers
Add new diffuse and clear

Figure 12 Pull-Push after light-pass. Each node contains the two colors for diffuse and caustic light paths and two buffers for received values, the black boxes are that for the caustics.

information than earlier casts due to the fact that nodes can be hit multiple times per frame. Additionally, it is necessary to push and pull the illumination after a light distribution.

4.8.1 Pull after Eye-passes

During the eye pass the ray casts create samples for each node independently. Whenever a ray passes through a node the diffuse illumination sample is weighted according to the Monte Carlo integration and added to the sum of this single node. The light is neither under nor over estimated, it is only represented with a different number of samples. Therefore, it is not necessary to push or pull anything. However, pulling light to the upper levels increases the quality for two reasons. First the illumination on lower levels is more detailed and more precise because the geometry is less approximated. Secondly, the number of samples for hierarchical nodes is increased which decreases the variance at these nodes.

Figure 11 shows the necessary steps exemplary. For leaf nodes the double buffered value is copied to the old one. For all other nodes the samples from the children are collected first. It is not possible to simply add the children's samples because they can have different areas. Averaging with weights based on the areas solves that problem. Then again the results can be applied and the no longer required buffers can be cleared for the next pass.

4.8.2 Push-Pull after Light-passes

After distributing light in the hierarchy not every node might contain its correct lighting value. It may happen that a node somewhere above the leaf level was illuminated and its child nodes are still black. In contrast to the eye pass, the hierarchy is indeed erroneous and will never converge to the correct result due to the LOD selection. Hence the illumination must be *pushed* down *and pulled* up to fill the gaps. This is essentially the same as for hierarchical radiosity approaches. Another problem arises from the fact that a node might contain diffuse lighting from a previous eye pass which has to be preserved and thus should never be pushed. As mentioned this is done to avoid the introduction of artifacts when pushing high level information to the detailed levels. It is possible to push and pull the new received results only and add them to the old ones, instead of pushing or pulling the information from the aggregated buffers. This way, the light-pass remains independent. If used it contributes to the convergence of the diffuse value from a different point of view but does not disturb the previous information.

The four steps in figure 12 can be done in a single tree traversal. Steps 1 and 2 are performed in pre-order before visiting the children and the other two steps in post-order. Note that step 3 immediately stores the average of caustic values instead of doing the extra work necessary in the diffuse component.

It would be possible to store the caustics in the diffuse colors too. Two RGB color values per node could be saved this way. On the other hand the light distribution pass would be necessary in each frame. The independent buffered solution makes the convergence of caustics and other passes independently which enables to switch off the light casting path after a short rendering time.

5 Evaluation

So far many heuristics and ideas were presented to be able to render large scenes. This section will cover how different resolutions influence the memory requirements and performance, the errors are made by the algorithm and finally how the stored diffuse illumination improves the performance.

The first two sections 5.2 and 5.3 show that the cone casts reduce the number of nodes from the hierarchy which must be loaded and speedup the intersection queries. If the resolution decreases, the size of the scene tree does the same, but the upper bound (200 nodes per pixel) is too large to guarantee that any scene could be rendered in-core.

Afterwards the sections 5.4, 5.6 and 5.7 handle different errors. The parameters for the heuristics are evaluated and chosen to make as few errors as possible. In all experiments the light distribution path introduces severe problems, most times visible as color shifts in the error images. Despite that, the diffuse illumination is computed fast and close to the reference images. In section 5.8 the benefits of storing the diffuse illumination in the hierarchy are demonstrated.

5.1 Path Tracing as Ground Truth

For reasons of mathematical correctness we are using a simple non-optimized path tracer for the reference image generation. Still our reference images contain two errors. First, the path tracer will never generate light paths longer than a given maximum. This means light intensity is systematically underestimated, but this error is usually smaller than the noise. In contrast the hierarchical approach is indeed generating arbitrary long paths.

The second error source are floating point inaccuracies. Hence collision detection might fail and colors can deviate from the true values. The first error can be reduced by a higher maximum path length and longer computation time, whereas for the second a higher precision and therefore more memory and time would be required.

The implemented path tracer is using the same material inputs and parametrization as the hierarchical approach. Therefore images can be used to compare the hierarchical approach to the reference with respect to the algorithmic correctness.

5.2 Node Count on Different Resolutions

One of our targets was to be able to render a large scene with only a partition of the original information in memory. Consequently, the question is if the number of required nodes is proportional to the image resolution? In a scene the total number of nodes is twice as high as the number of primitives since they are saved in a binary tree. During rendering we marked all nodes which were directly hit by a ray and all their parents. Therefore the number of 'used' nodes is that which is really necessary to create the shown images. To test this, we rendered scenes at different resolutions and also included numbers from other tests if these experiments did not change the tree usage.

Figure 13 shows a scene with 20.8 million nodes which was rendered at many resolutions. The graph in figure 14 plots the number of nodes per pixel of this and other scenes. The number varies for different maximal resolutions, rendering times and scene setups. The longer an image is rendered the more nodes are hit by chance up to a threshold. At the beginning each iteration adds many nodes to the number of used

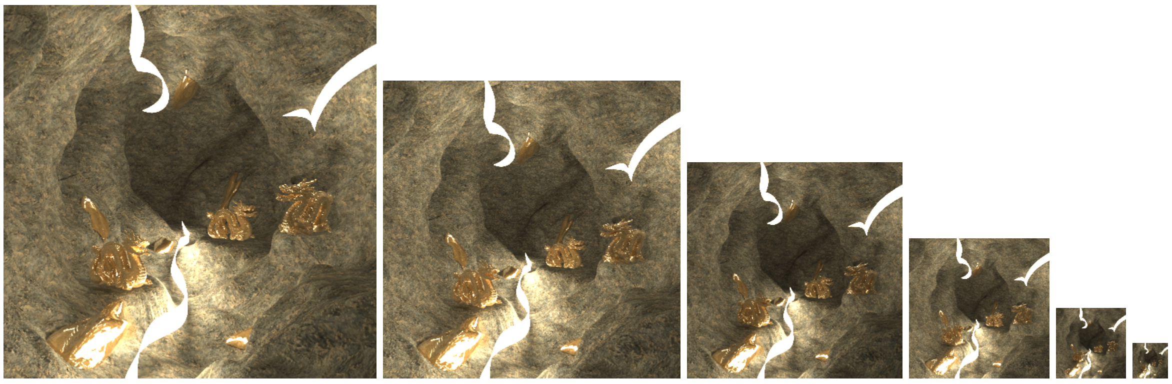


Figure 13 Dragon Cave scene with area lights. Scene complexity per resolution (f.l.t.r.): 500 × 500 - 17053k nodes, 400 × 400 - 13413k nodes, 300 × 300 - 10334k nodes, 200 × 200 - 5771k nodes, 100 × 100 - 2228k nodes, 50 × 50 - 444k nodes

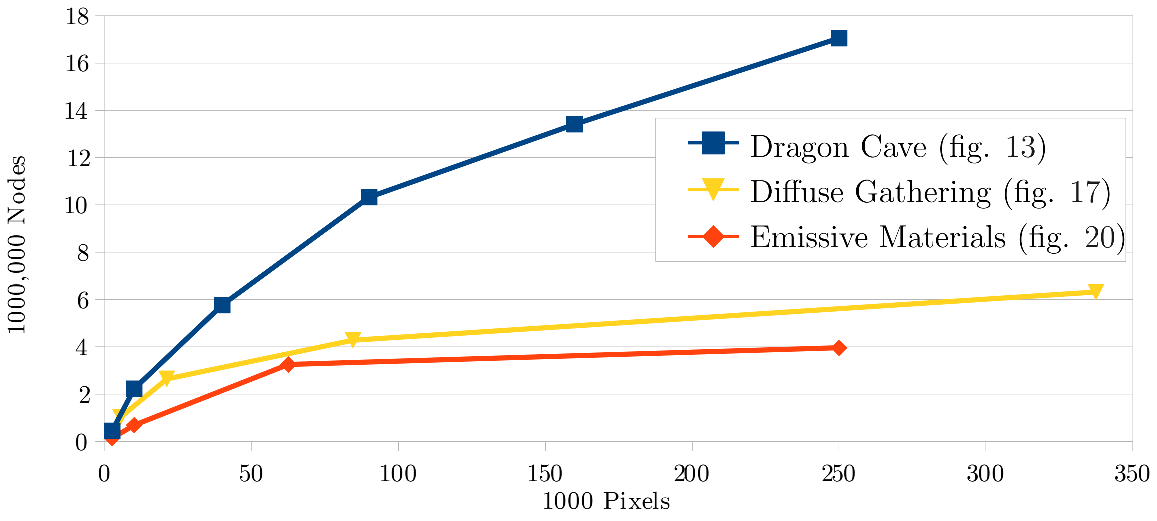


Figure 14 Used nodes on different resolutions. The data comes from experiments with different maximum node counts and resolutions.

nodes, then the slope decreases and the number converges slower. The shown values are taken when the noise in the images is relatively similar. It is visible that the node count tends to zero with the decreasing resolution. In higher resolution images the number of available nodes is smaller than the resolution sampled by the rays, so the number of used nodes remains more or less the same. This is a logical consequence of the fact that the scene resolution is converging to the available maximum. For lower resolutions the function gets more linear with a slope of 100-200 nodes/pixels. Therefore, to render a full quality two megapixel image, at least 200 million nodes would be necessary. Approximately 24 GB tree data are required with the current encoding.

These results demonstrate that the number of nodes required in a hierarchical approach is limited according to the image resolution. The still very high number justifies the heavy compression of the BRDF and other parameters. A possible alternative would be to store a reduced set of parametrization in a lookup table, like in color-indexed images, to reduce the memory footprint or to increase the expressiveness of the BRDF.

5.3 Iteration Time on Different Resolutions

Together with the shrinking memory usage the number of ray cast per second is expected to increase, because a cut through the hierarchy decreases the recursive depth of the intersection test. We expected to get a speedup proportional to the logarithm of the image resolution since the solid angle per pixel increases for lower resolutions. Thus the number of nodes decreases proportionally and therefore the depth of the scene tree logarithmically. The decrease of nodes is evaluated in the previous section which has demonstrated that it becomes proportional when the scene resolution is close to the image resolution. To determine the speedup factor we measured the throughput of elementary ray casts to the tree per frame and took the average over 20 frames.

	no cones	full res.	half res.	quarter res.	eighth res.
	$100\ ray/s$	$100\ ray/s$	$100\ ray/s$	$100\ ray/s$	$100\ ray/s$
Dragon Cave	941 ± 35	1314 ± 56	1728 ± 30	2460 ± 258	3043 ± 232
Diffuse Gathering	686 ± 51	1250 ± 89	1589 ± 68	1477 ± 32	2068 ± 113

As the table shows the number of rays per second is increasing when the image resolution decreases. The first column contains the reference values using rays and no cone casts on full resolution. The factors are 1.4×, 1.3×, 1.4× and 1.2× for the first scenario and 1.8×, 1.3×, 0.9× and 1.4× for the second. Enabling cone casts improves the performance. Especially the diffuse gathering scene (visible in figures 16 and 20) which has a high amount of indirect lighting benefits from the cone optimization.

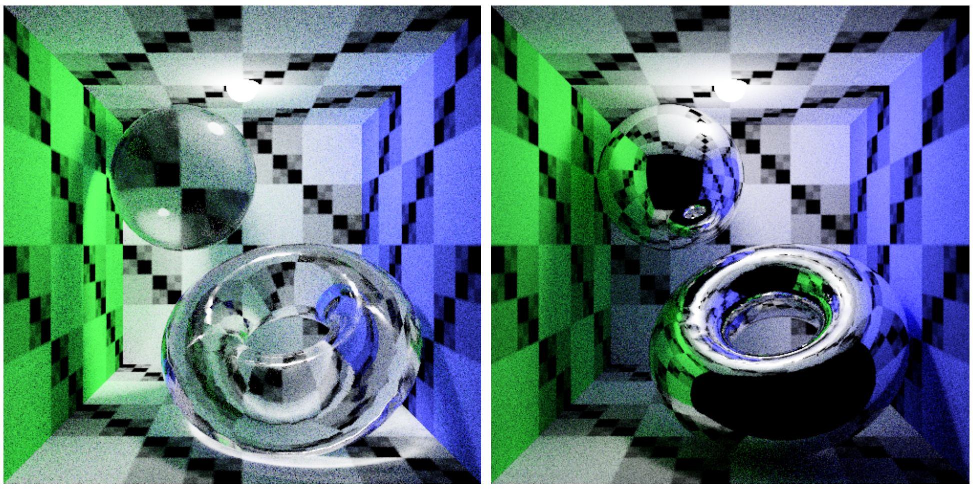
Taking the numbers of used nodes n_i at the respective resolutions we can estimate expected value with the quotient of the tree depths: $\frac{\log_2 n_i}{\log_2 n_j}$ The expected factors are 1.05×, 1.07×, 1.08× for the dragon scene and 1.03×, 1.03×, 1.07× for the other scene respectively. The achieved values are even better than expected. It is likely that increased cache coherency also improves the performance due to the reduced jump-width in the tree.

5.4 Cone Heuristics

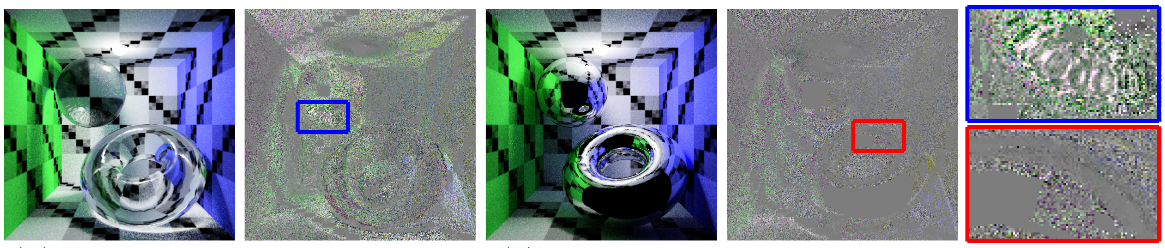
The first two experiments have shown that the performance increases with the use of cone casts. They improve the speed and might be able to render arbitrary large scenes in-memory. In the following we evaluate the errors made by the approach. The larger the used cones the more the performance improves, but bad parameters cause severe problems.

In section 4.5.2 heuristics were introduced to determine the new cone properties for recursive casts. They are used to estimate the new opening angles from a single scalar value. We rendered two scenes with a number of different control parameters b, c to test their influence. Figure 15 shows a subset of the made renderings at a resolution of 500×500. We also rendered the same images at 200×200 to force the usage of higher hierarchy levels. In both cases nearly no differences are visible. In earlier development, the introduction of the heuristic fixed some errors, but we were not able to reproduce these artifacts which might be a result of the later introduced probabilistic cone resampling.

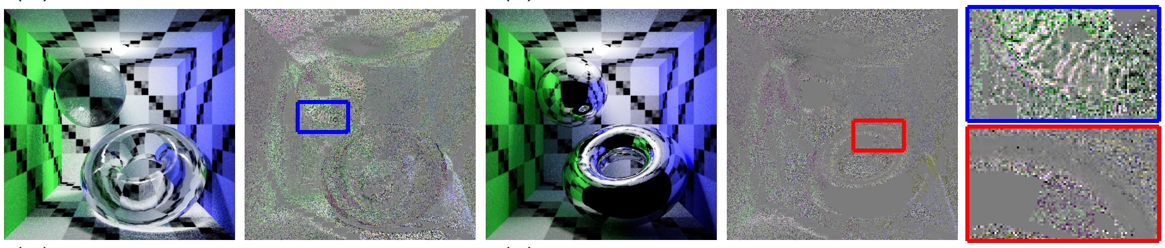
However there are a few noticeable artifacts. On closer look, a little difference is observable in the lower left corner of the glass sphere. In image 15 (f) the visible error is primary noise whereas all other renderings 15 (b,d,h) show structures in this



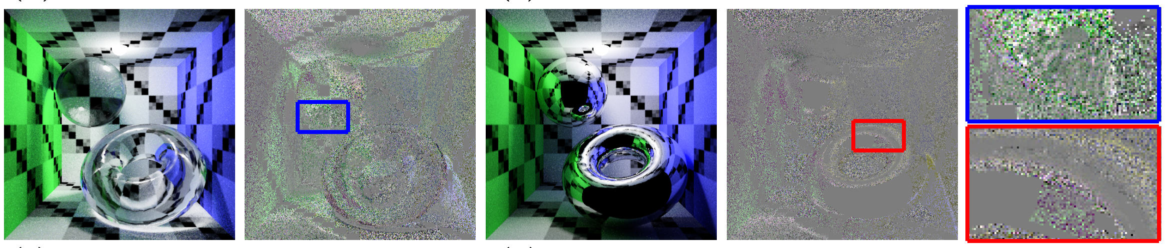
(a) Path traced references. Textured Cornell-box. *Left*: refraction indices $n_{sphere} = 1.2$ and $n_{torus} = 1.5$, *Right*: mirroring surfaces only.



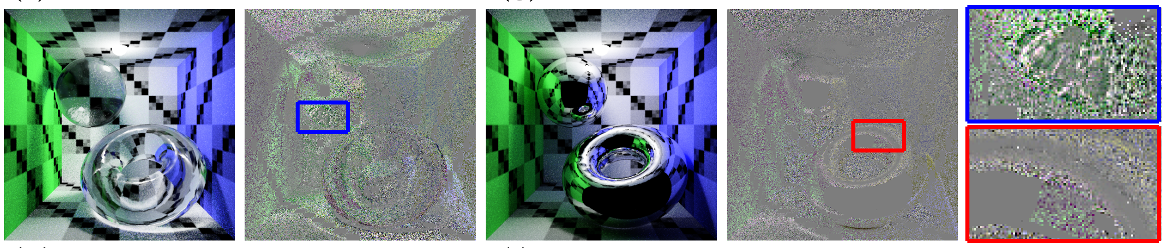
(b) $c = 0$ cone constant, 4388 k nodes **(c)** $b = 0$ cone constant, 2919 k nodes



(d) $c = 0.5$, 4395 k nodes **(e)** $b = 0.5$, 2906 k nodes



(f) $c = 1.0$, 4412 k nodes **(g)** $b = 1.0$, 2888 k nodes



(h) $c = 2.0$, 4395 k nodes **(i)** $b = 2.0$, 2842 k nodes

Figure 15 Evaluation of the heuristic parameters b, c from section 4.5.2. The node counts are measured after iteration 5.

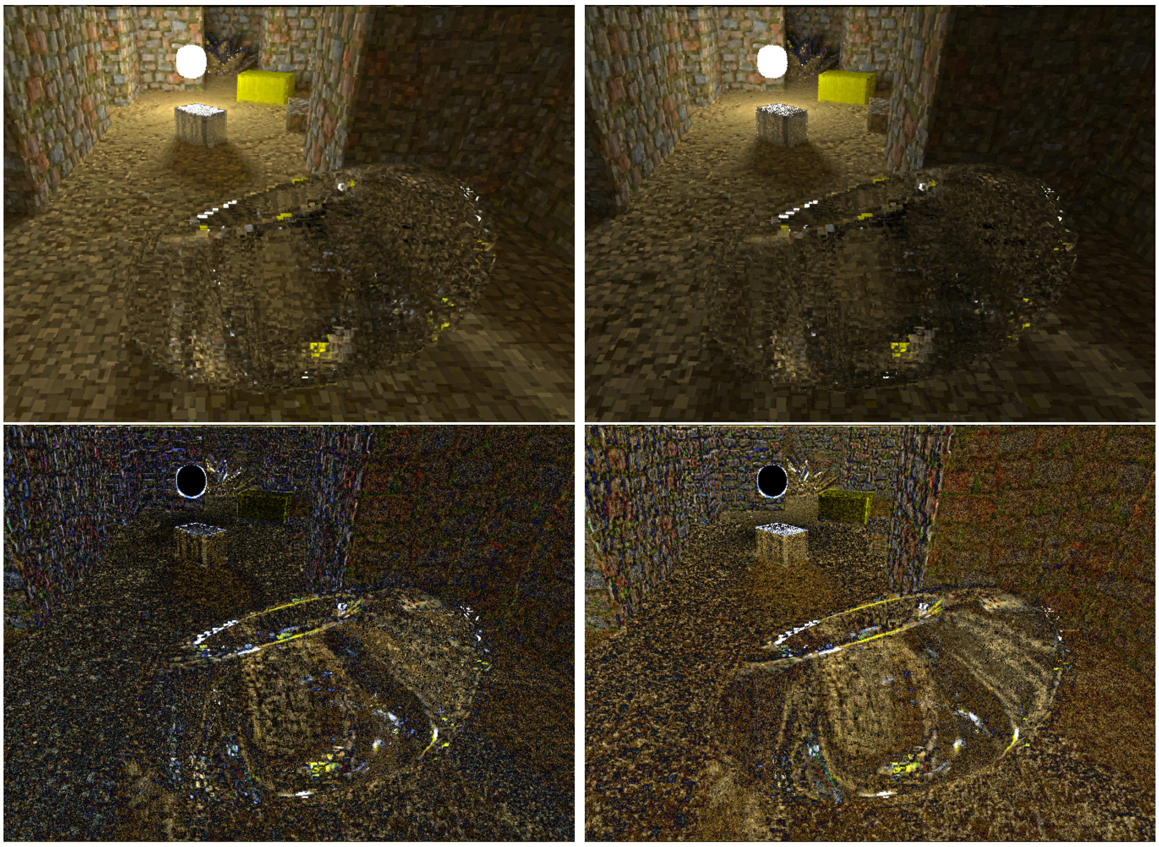


Figure 16 Influence of the smooth normal: The left column was rendered using the heuristic, the right with normals of length one. The bottom row shows the absolute error $\times 2$ (darker means smaller error) with respect to the reference (figure 19 (a))

region which do not belong there. Also, the number off used nodes is maximal for the refraction-heuristic parameter $c = 1$. Therefore the precision of the refracted image is maximal and $c = 1$ seems to be the best choice.

The reflection case in image (i) is erroneous at the bottom edge of the torus. There are no artifacts for a too small reflection-heuristic parameter b observable. Probably parabolic mirrors would cause artifacts when rendered without the heuristic. Using too large cones in the test scene does not cause errors fast. However, lenses and other objects which focus the rays and narrow the cone lead to errors, as happened in the refraction case. Increasing b decreases the number of required nodes in the test scene monotonously. Hence using $b \in [0.5, 1]$ slightly decreases the memory requirements but does not change the quality visibly.

The relatively small influence of the heuristics justifies their simplicity. Using ray differentials would make the cone computation more precise but would not increase the quality much while requiring considerably more memory for the tensor.

5.5 Surface Normal Heuristic

A further heuristic Θ was introduced in section 4.4. In figure 16 the tree is cut at a constant node size to force the casting to use clustered nodes. The left image which was created with the heuristic is slightly brighter than that with usual normals (with length one). The error images show that using the heuristics gets closer to the reference in average. Of course both have a high error because of the artificial coarse scene resolution.

5.6 Feasible Scope of Materials

The currently used BRDF has some limitations. It cannot handle anisotropy or translucency and two sided materials are not supported because this would make another color value for back faces necessary. Also, it is not possible to have multiple specular reflection directions which can happen for clusters of reflecting surfaces.

Nevertheless, there are many different materials possible. The following experiment evaluates how materials behave in a close view as well as in distant clusters. To that purpose the material test scene was rendered 3 times. Once with the reference path tracer which is not using the hierarchy but still uses the same surfel-material descriptor and two times from different distances with our hierarchical illumination. The scene consists of a Cornell box with a displacement mapped floor, a box with the material under observation and two light sources. One in the upper right corner and a weak one in front of the scene.

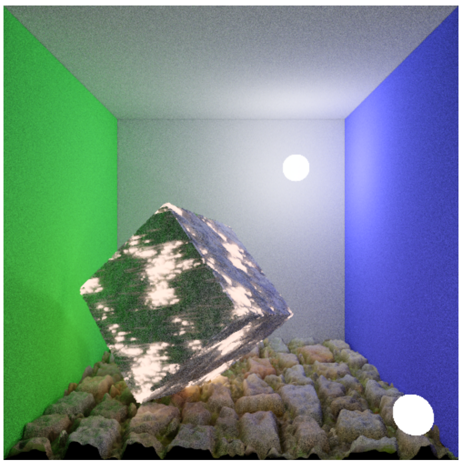
The tessellation is the same for all three scenes (around 5,636 thousand surfels) but the number of used nodes differs. For the emissive scenario light distribution was disabled. In both other cases it is used without masking of nodes which were not hit from the eye casts. Therefore the number of used nodes mainly depends on the distance to the light sources. In the next section (5.7) the consequences of masking LODs are evaluated in detail.

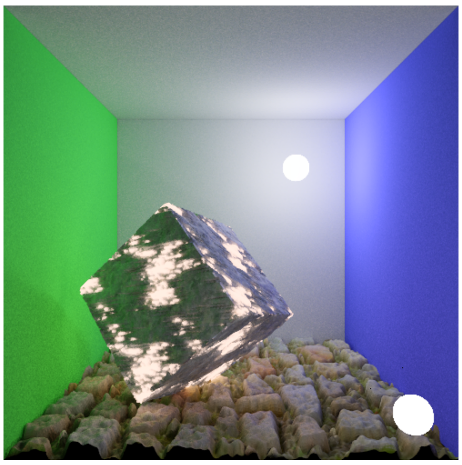
All three scenes contain complex illumination situations which all cause some errors. A systematical artifact occurs in the corners of the Cornell-box (best visible in the upper left on the images from figure 18). This is caused by the geometrical representation through surfels, because rays originated at some location on the disc may lie outside the box. This also happens in the reference images themselves.

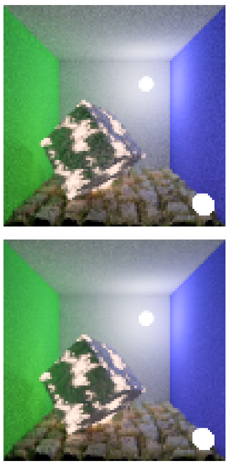
A similar error occurs at the high frequency scratches in the stone block of the emissive scene (figure 17 (a), (b)). It is possible that the hierarchical approach increases the error by using a cluster instead of a surfel, but this is not likely the case here because the maximum geometry resolution is used in that area. This was verified by rendering the used node level which created a completely black image meaning that no hierarchical node was used for the front sides. But, other than in the reference, the diffuse color of a surfel is buffered and influences rays at neighboring pixels. This sharing leads to the over- and under- illuminated areas in high frequency details on the cube. Apart from these differences, only noise is observable in the difference image. Neither the large nor the small rendering show noticeable differences when compared visually.

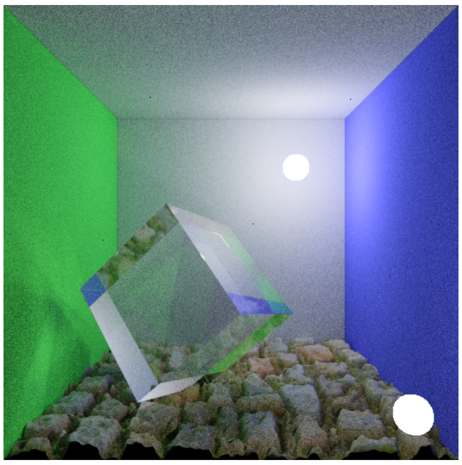
Much higher deviations are observable in scenarios (d) to (i). In contrast to scene (a) the additional light distribution pass is enabled because caustics are an essential part of the illumination. As was already noticeable in figure 9 (section 4.7.1) even the area corrected illumination underestimates the luminance. This is again visible in the direct comparison. The error images in figure 18 show that indirect light is underestimated in most scene parts. The effect increases for the smaller renderings where the estimated area for hierarchical nodes is worse. It is certain that the error is caused in the computation of illumination as the additional experiments show.

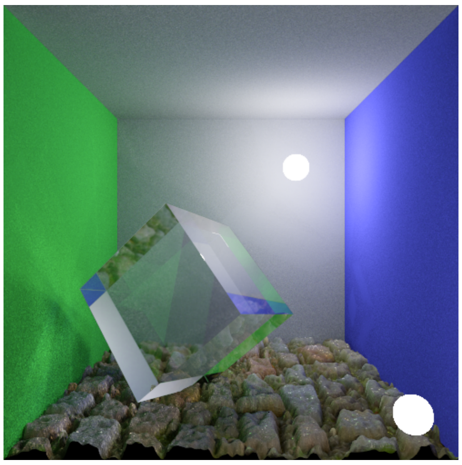
Both scenes were rendered again on 500×500 resolution where the contribution to the diffuse illumination was disabled. This means that the light pass is used only to generate caustics. The error images can be found in figure 18 on the lower right of the 3 images. In this configuration the remaining error appears only in the areas affected

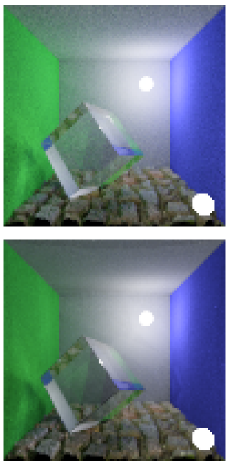

(a) Emissive texture

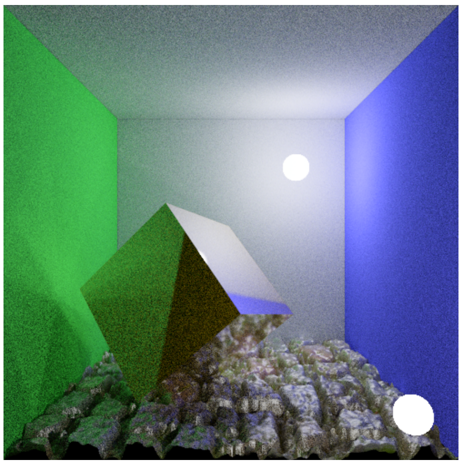

(b) 3959/5636 k Nodes

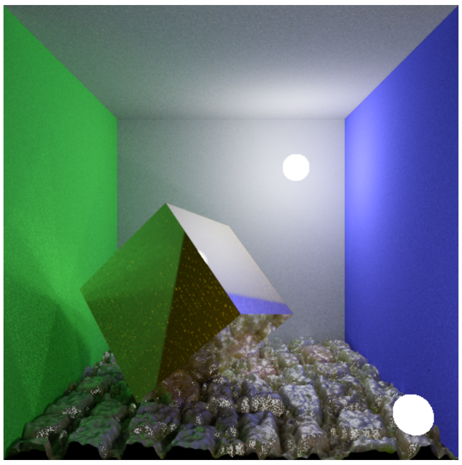

(c) 689 k Nodes


(d) Glass with $n = 2.5$


(e) 4381/5636 k Nodes


(f) 4217 k Nodes


(g) Copper cube and glossy floor


(h) 4389/5636 k Nodes

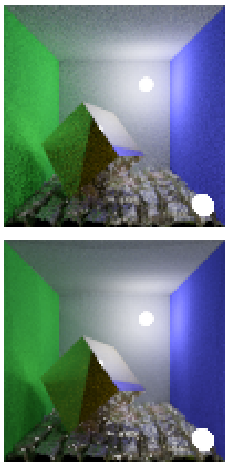

(i) 4095 k Nodes

Figure 17 Materials within different scene resolutions.
Left: Path traced Reference, path length 6-8
Middle: Hierarchical illuminated
Right: 100x100 pixel renderings, *Top*: reference, *Bottom*: Hierarchical on a higher level

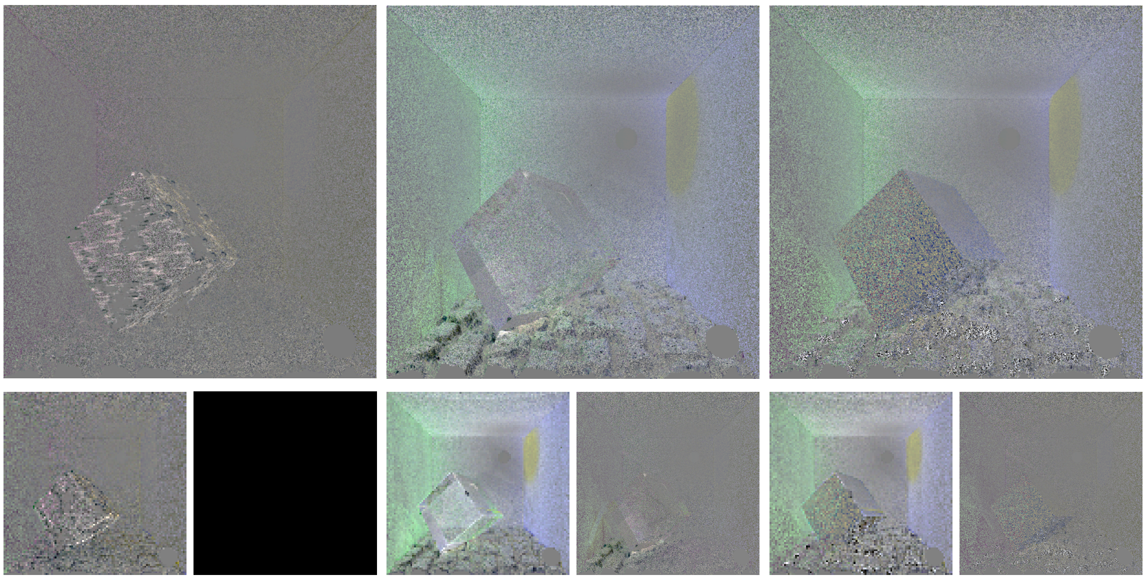


Figure 18 Amplified (x 4) difference images from figure 17
Top row: Errors of the large samples (a)-(b), (d)-(e), (g)-(h)
Bottom row: Errors of the small samples (c), (f), (i) left and additional tests right

by the caustics. Therefore, the problem is not originated in the different materials but in the illumination from the photon distribution itself.

5.7 Light Distribution With and Without Masking

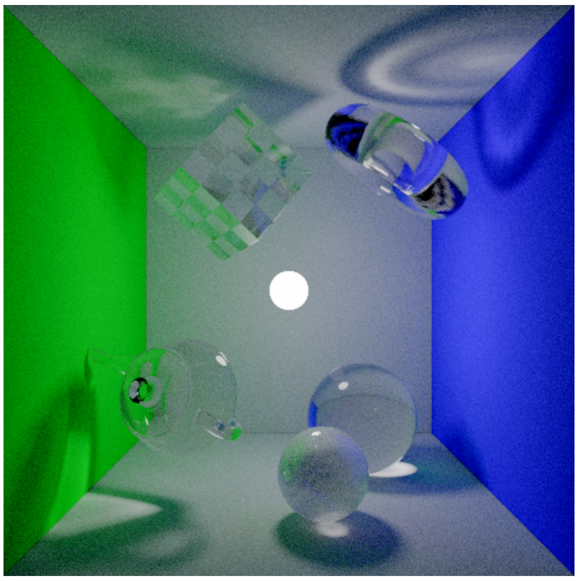
So far the whole scene resides in the memory. In figure 17 (f) and (i) the number of used nodes is close to the high resolution setups (e) and (h). This is because the light source used all nodes, which leads to higher scene resolutions depending on the light distance. For the purpose of loading only the scene which is relevant to the view, it is important to know how rays originating at light sources are influenced.

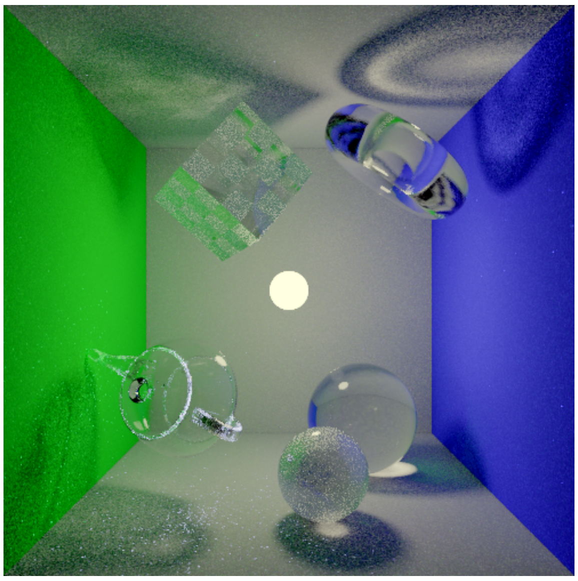
We implemented a masking which forbid the light distribution pass to use untouched nodes. This effectively simulates a partially loaded tree. In such a tree the intersection problem must be solved differently. There are three different cases to be considered:

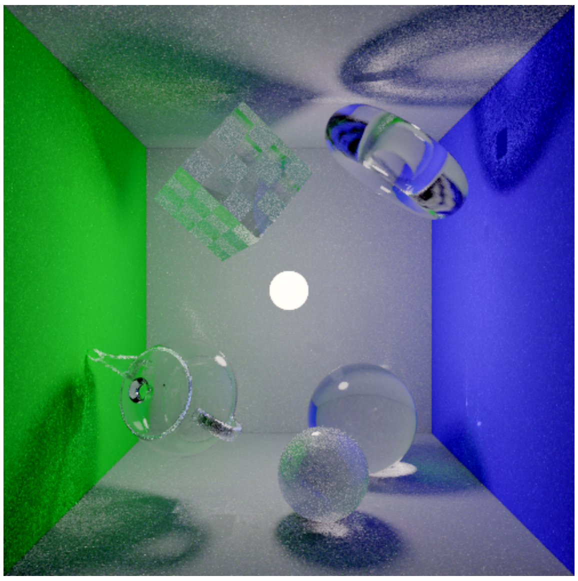
1. Both children are in memory (unmasked): There is nothing to do, the intersection is determined by a recursive call.
2. Both children are masked: If the current parent is hit this is reported as the tree intersection, otherwise there is no intersection with this node.
3. One child is masked: If the child is hit this result will be used because it is computed on a more detailed level. If not, the decision must be made on the parent level.

In figure 19, image (c) was rendered using the described masking. Many artifacts (dark and bright boxes) occur due to wrong decisions in the third case. Picture (d) is created with a modified masking: For each loaded node its sibling must be in memory too. Thus the tree nodes have either zero or two children. Case three is avoided this way and the memory footprint slightly increased.

In all three renderings, with and without masking, the caustic of the teapot is wrong. This is a general problem caused by self intersections between rays and the geometry


(a) Reference


(b) Full hierarchy


(c) Masked: use only nodes which were hit by eye casts

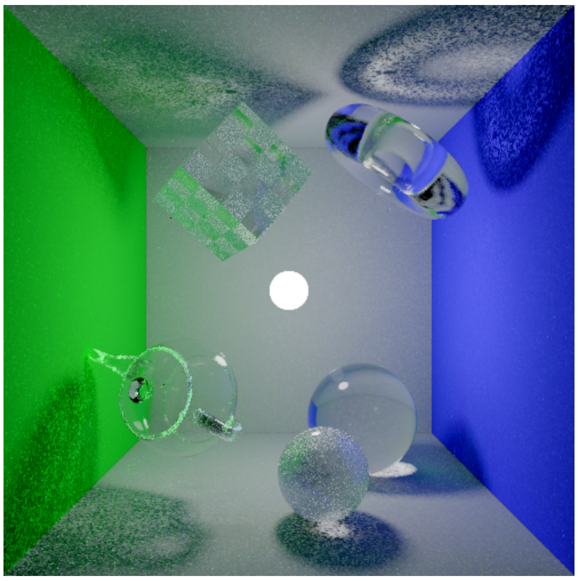

(d) Modified masking: also use the siblings of the touched nodes in the tree

Figure 19 Behavior of light distribution on view importance reduced scenes

close to the origin. In case of refractions at more complex objects (such as the teapot) the test described in the last paragraph of section 4.5.2 does not succeed. We tested to add an offset which caused more correct intensities but also different shapes in the caustics.

Comparing (b) and (d) the masking produces similar results, except that there is a coarser noise in all caustics. More light paths or photons cannot solve the problem because this is caused by the reduced scene resolution, thus certain light passes are not possible. At least with the current heuristics and the restriction to bounding box intersections, light paths on the reduced hierarchy cannot be done. However, it seems possible to use the reduced hierarchy in a bidirectional path tracer because after the additional indirection between light-pass and eye-pass sample the noise should not be noticeable.

5.8 Stored Diffuse Illumination

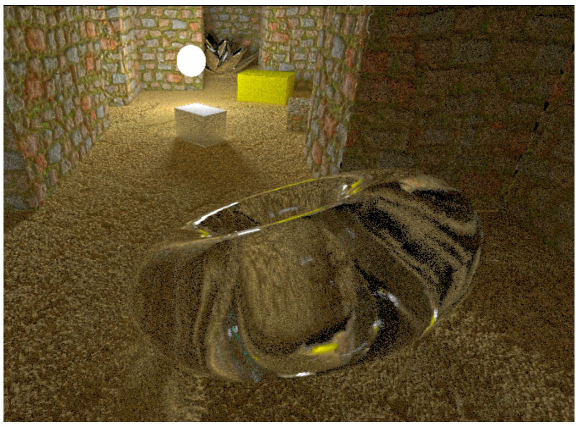
Another point of interest is the correctness of the stored illumination. It is one of the parts which is expected to increase convergence most, because all rays are reused for diffuse illumination. To visualize the stored information we let the rendering run for 20 frames and then reset the image and changed the path length to 0 diffuse bounces. The number of specular reflections is kept high to gather the illumination on specular objects. As a result this gathering step does not compute new direct or indirect illumination and the image is created from the stored light only. Furthermore, the experiment demonstrates what happens if the opening angle of the primary ray (equation 4.18) is chosen larger than the pixel can cover.

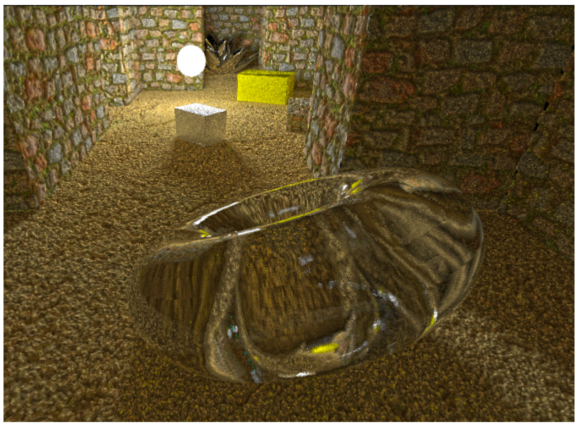
The light distribution pass and therefore the generation of caustics was disabled. Previous experiments have shown several errors and the additional costs for this pass are relatively high. So the difference in caustics to the reference image is not an error.

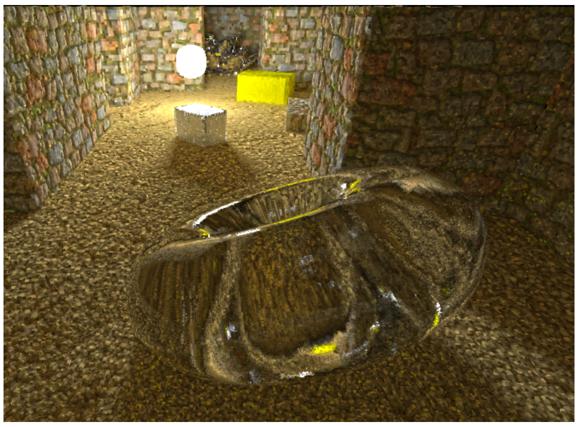
The experiment shows (figure 20) that the results per node are quite well converged after a short time. Except the corner in the right which is only indirectly illuminated, noise is barely visible. Moreover the specular reflections could be gathered much better because of the four times smaller iteration-duration due to the excluded steps for the lighting. The following table shows how much time an iteration takes for different path lengths (maximum lengths: diffuse + specular). After computing the diffuse illumination only specular reflections must be traced. This increases the iteration throughput by 4.09×. The last column shows that simpler scenes could be gathered much faster again.

Num. reflections	3 diff. + 6 spec.	0 diff. + 6 spec.	0 diff. + 2 spec.
Eye casting pass	49400 ms	19100 ms	3850 ms
Pull diffuse	1450 ms	—	—
Light distribution 6.32M photons	25900 ms	—	—
Light push/pull	1475 ms	—	—
Total iteration	78225 ms	19100 ms	3850 ms

In the bottom row of figure 20 higher levels in the kd-tree of the same 20 ray preillumination are shown. These are the results from the pull step. To visualize the higher levels the opening angle ($\tan \alpha$) of the primary ray was multiplied with 8 and 32 respectively. This is interesting for several reasons: First it shows that the higher level


(a) Path traced reference: depth 8, 10000 rays per pixel, 22 h


(b) Gathering on high resolution: 135 min.


(c) Gathering on a 8x less resolved cut: 97 min.

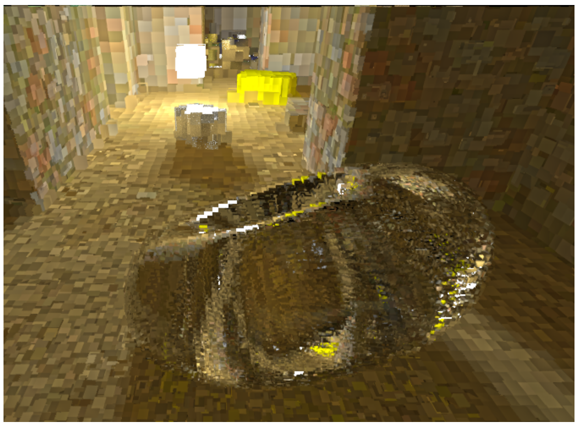

(d) Gathering on a 32x less resolved cut: 88 min.

Figure 20 Stored diffuse illumination after 20 rays per pixel; The light is gathered for another 140 rays with the lighting from the first 20 iterations; all three renderings took 6314 k of 18920 k nodes

information which is otherwise used from larger indirect cones is still meaningful. Second it also demonstrates that even the specular reflections are handled relatively well on the hierarchical level. The torus in the last image begins to have severe artifacts and the glossy box in the 8x resolution shows some errors at the edges. Without the forced LOD cones with such a large radius are seldom used and if that happens the ray is of small importance, because otherwise it would not have been this wide. This makes the error made here acceptable.

Concluding the use of the scene discretization to store intermediate results of the ray casts is performing very well. While this experiment is using, but not relying on the hierarchical approximation, this optimization could be applied to other ray tracing based approaches too.

6 Conclusions

We found an approximate solution to solve the global illumination problem directly on hierarchical information. To do so, we used cone casts to estimate the level of detail which is probably sufficient for the local illumination according to heuristics. The heuristics were introduced to reduce the memory footprint and to increase the performance where a correct solution is not possible due to the hierarchical approximation anyways. Also, we stored intermediate illumination results to improve rendering performance and to generate caustics without a photon map. The developed method partially succeeds. It can render diffuse scenes relatively well while bounding the number of necessarily loaded scene information. In scenes with specular objects the generation of caustics introduces larger errors which is an unsolved problem yet.

We showed that the usage of cones increases the rendering performance and decreases the memory requirements if the image resolution is lessen. On the other hand we did not achieve to render the full spectrum of reflection effects. Especially caustics cannot be generated properly when the scene is reduced to a lower level of detail by the eye-casts.

A further advantage is that the cone radius can be decreased smoothly to get a more correct result. For an opening angle of zero the renderer becomes a path tracer which makes use of intermediate results. Anything up to a maximal angle decreases the accuracy by choosing coarser LODs only, because the stochastic cone casting itself does not introduce a further bias. The maximal angle of a cone is primary given by the pixel's solid angle. Afterwards the cone angle depends on the reflections and the scene complexity and representation.

One reason for some errors is the chosen representation of pure surfel-based geometry because of the necessary overlap to create a hole free surface. The light distribution pass, which should generate the caustics and increase the performance due to added bidirectionality, fails in computing the correct visible surface area. This leads to a systematic under-illumination which is visible in the resulting images (figures 10, 18). There are multiple ways to avoid this problem. The scene could be represented by atomic small triangles which have only a single set of parameters like the surfels we used. These would have the same advantages as the surfels but would be overlap free. Another option is to keep the representation and try other methods, like to use a bidirectional path tracing and omit the light distribution pass.

However, storing diffuse illumination and utilizing the discretization yields a remarkable increase in convergence speed. As section 5.8 shows a few iterations are sufficient to create a good diffuse illumination. It is possible to do a gathering only after some time, to speed up the illumination further. Additionally the usage of stored information during sampling leads to a theoretically unbound number of light bounces. Similar to iterative radiosity approaches each iteration makes use of the former more converged result.

Surprisingly, the heuristics introduced in section 4.5.2 and evaluated in section 5.4 have a relatively small influence to the results. In case a cone hits one or multiple surfaces tracing the shape which would be necessary to cover the reflection is unfeasible in general. Instead we trace only a single ray and maintain an opening angle for the cone. The cone itself is sampled stochastically where the probabilistic casting seems very robust and reduces the error made by the heuristics for the angle computation. Nevertheless it could be replaced by the much more reliable ray differentials to ensure a more correct reflection.

Another problem we tried to solve is the proper representation of a cluster node's BRDF. We introduced a placeholder reflection function which allowed easy interpolation between nodes. It is based on a non-unit normal to smooth the effect of bad represented clusters where reflections deviate extremely.

7 Future Work

There are many parts of our approach which require further investigations. Using cone casts to render large scenes allows to determine the scene resolution top down. This should be combined with a deferred loading to enable the rendering of scenes which would otherwise not fit into the main memory. At the current state it is not possible to render caustics properly with only partially loaded data. We believe this can be solved by better solutions for the node representation, including BRDF and geometry, and the usage of other methods like bi-directional path tracing or normal photon mapping. Moreover, a node importance which considers light sources and view-rays could choose a better tree node level for the caustic generation.

To cover the reflection function of a cluster better, we propose to use general BRDF models which are designed to fit arbitrary measured materials. These suffer from the problem to consume too much memory which could be solved by using lookup tables for similar patterns. Together with a more realistic sampling for clusters, ray differentials should be introduced to compute better cones for the outgoing rays.

The geometric representativeness of cluster nodes could be increased by two things. First the bounding volumes, currently axis aligned boxes, could be exchanged by better fitting shapes. Thus the stochastic cone casting would generate a sampling closer to the original scene. Additionally the tree could be generated bottom up to optimize the density of clustered geometry and similarity in respect to the reflection behavior. This would increase the representativeness of nodes in higher tree levels and would reduce the problem of diverging reflection patterns which would support the idea of lookup tables.

Finally a GPU implementation could be done to increase the performance. Thereby the two main problems to solve are how to load the required LOD at the beginning and how to avoid write-write hazards. The later can happen when multiple parallel traced rays try to store their intermediate results in the nodes. It should be possible to use atomic adds straight forward without loosing much performance except at some focus points where many rays cross.

We used a kd-tree to build the bounding volume hierarchy in $\mathcal{O}(n \log n)$. Still the build time takes some seconds which is too much for animated scenes. Animating objects or light sources need further investigations in the recomputation of hierarchical information. Nevertheless, it should be easy to animate the camera and reuse the stored diffuse illumination. Thus camera walks might even be possible at real time frame rates.

References

[ABR01] Sheldon Axler, Paul Bourdon, and Wade Ramey. *Harmonic Function Theory*, volume 137. Springer, 2001.

[AK89] James Arvo and David Kirk. A Survey of Ray Tracing Acceleration Techniques. *An introduction to ray tracing*, pages 201–262, 1989.

[Ama84] John Amanatides. Ray Tracing with Cones. In *ACM SIGGRAPH Computer Graphics*, volume 18, pages 129–135. ACM, 1984.

[AS00] Michael Ashikhmin and Peter Shirley. An Anisotropic Phong BRDF Model. *Journal of Graphics Tools*, 5:25–32, 2000.

[BA08] Tamy Boubekeur and Marc Alexa. Phong Tessellation. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 27(5):141:1–141:5, December 2008.

[Bli77] James F. Blinn. Models of Light Reflection for Computer Synthesized Pictures. In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '77, pages 192–198. ACM, 1977.

[BSH02] Philippe Bekaert, Mateu Sbert, and John Halton. Accelerating Path Tracing by Re-using Paths. In *Proceedings of the 13th Eurographics Workshop on Rendering*, EGRW '02, pages 125–134. Eurographics Association, 2002.

[Bun05] Michael Bunnell. Dynamic Ambient Occlusion and Indirect Lighting. *GPU Gems*, 2(2):223–233, 2005.

[CB04] Per H. Christensen and Dana Batali. An Irradiance Atlas for Global Illumination in Complex Production Scenes. In *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques*, EGSR'04, pages 133–141. Eurographics Association, 2004.

[CCWG88] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P Greenberg. A Progressive Refinement Approach to Fast Radiosity Image Generation. In *ACM SIGGRAPH Computer Graphics*, volume 22, pages 75–84. ACM, 1988.

[Chr08] Per H. Christensen. Point-based Approximate Color Bleeding. *Pixar Technical Notes*, 2(5):6, 2008.

[CJ02] Mike Cammarano and Henrik Wann Jensen. Time Dependent Photon Mapping. In *In Proceedings of the 13th Eurographics Workshop on Rendering*, pages 135–144, 2002.

[CNS$^+$11] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive Indirect Illumination Using Voxel Cone Tracing. In *Computer Graphics Forum*, volume 30, pages 1921–1930. Wiley Online Library, 2011.

[Cra11] Cyril Crassin. *GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes*. PhD thesis, Universite de Grenoble, July 2011.

[CT82] Robert L. Cook and Kenneth E. Torrance. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics (TOG)*, 1(1):7–24, 1982.

[GCT86] Donald P. Greenberg, Michael F. Cohen, and Kenneth E. Torrance. Radiosity: A Method for Computing Global Illumination. *The Visual Computer*, 2(5):291–297, 1986.

[Gre03] Robin Green. Spherical Harmonic Lighting: The Gritty Details. In *Archives of the Game Developers Conference*, volume 2, pages 2–3, 2003.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the Interaction of Light Between Diffuse Surfaces. In *ACM SIGGRAPH Computer Graphics*, volume 18, pages 213–222. ACM, 1984.

[HBGM11] Niklas Henrich, Jakob Baerz, Thorsten Grosch, and Stefan Müller. Accelerating Path Tracing by Eye-Path Reprojection. *International Congress on Graphics and Virtual Reality (GRVR)*, 2011.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive Photon Mapping. In *ACM Transactions on Graphics (TOG)*, volume 27 number 5, page 130. ACM, 2008.

[HSA91] Pat Hanrahan, David Salzman, and Larry Aupperle. A Rapid Hierarchical Radiosity Algorithm. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 197–206. ACM, 1991.

[Ige99] Homan Igehy. Tracing Ray Differentials. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 179–186. ACM, 1999.

[Jen96] Henrik Wann Jensen. Global Illumination using Photon Maps. pages 21–30. Springer-Verlag, 1996.

[Kaj86] James T. Kajiya. The Rendering Equation. In *Computer Graphics*, pages 143–150, 1986.

[KM99] Jan Kautz and Michael D. McCool. Interactive Rendering with Arbitrary BRDFs using Separable Approximations. In *Rendering Techniques' 99*, pages 247–260. Springer, 1999.

[KTO11] Janne Kontkanen, Eric Tabellion, and Ryan S Overbeck. Coherent Out-of-Core Point-Based Global Illumination. In *Computer Graphics Forum*, volume 30, pages 1353–1360. Wiley Online Library, 2011.

[Laf96] Eric P. Lafortune. Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering. Technical report, 1996.

[Lew94] Robert R. Lewis. Making Shaders More Physically Plausible. In *Computer Graphics Forum*, volume 13, pages 109–120. Wiley Online Library, 1994.

[LFTG97] Eric P. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear Approximation of Reflectance Functions. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 117–126. ACM, 1997.

[LRR04] Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Efficient BRDF Importance Sampling Using a Factored Representation. *ACM Trans. Graph.*, 23(3):496–505, August 2004.

[LSK05] István Lazániy and László Szirmay-Kalos. Fresnel Term Approximations for Metals. 2005.

[LW93] Eric P. Lafortune and Yves D. Willems. Bi-Directional Path Tracing. In *Proceedings of third international conference on computational graphics and visualization techniques (Compugraphics '93)*, pages 145–153, 1993.

[Mar03] George Marsaglia. Xorshift RNGs. *Journal of Statistical Software*, 8(14):1–6, 2003.

[MN98] Makoto Matsumoto and Takuji Nishimura. Mersenne Twister: a 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[MU12] Rosana Montes and Carlos Ureña. An Overview of BRDF Models. LSI Technical Reports;2012-001, 2012.

[NRH$^+$77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometric Considerations and Nomenclature for Reflectance. *National Bureau of Standards*, 1977.

[Pho75] Bui Tuong Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[PPI98] Ingmar Peter, Georg Pietrek, and Fachbereich Informatik. Importance Driven Construction of Photon Maps. In *In Rendering Techniques '98 (Proceedings of the 9th Eurographics Workshop on Rendering*, pages 269–280. Springer-Verlag, 1998.

[PZVBG00] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 335–342. ACM, 2000.

[REG$^+$09] Tobias Ritschel, Thomas Engelhardt, Thorsten Grosch, Hans-Peter Seidel, Jan Kautz, and Carsten Dachsbacher. Micro-Rendering for Scalable, Parallel Final Gathering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2009)*, 28(5), 2009.

[RL00] Szymon Rusinkiewicz and Marc Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 343–352. ACM, 2000.

[Rus04] Szymon Rusinkiewicz. Estimating Curvatures and Their Derivatives on Triangle Meshes, 2004.

[Sch93] Christophe Schlick. A Customizable Reflectance Model for Everyday Rendering. In *In Fourth Eurographics Workshop on Rendering*, pages 73–83, 1993.

[Sch94] Christophe Schlick. A Survey of Shading and Reflectance Models. *Computer Graphics forum*, 13:121–131, 1994.

[Slo08] Peter-Pike Sloan. Stupid Spherical Harmonics (SH) Tricks. In *Game developers conference*, volume 9, 2008.

[Slo13] Peter-Pike Sloan. Efficient Spherical Harmonic Evaluation. 2(2), 2013.

[VG95] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*, pages 145–167. Springer, 1995.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis Light Transport. In *Computer Graphics (SIGGRAPH '97 Proceedings*, pages 65–76. Addison Wesley, 1997.

[War92] Gregory J. Ward. Measuring and Modeling Anisotropic Reflection. *ACM SIGGRAPH Computer Graphics*, 26(2):265–272, 1992.

[WUS03] M. Wand, Ma Usa, and W. Straßer. Real-Time Caustics, 2003.