Lighting Networks
A New Approach for Designing Lighting Algorithms

Philipp Slusallek Marc Stamminger Hans-Peter Seidel

University of Erlangen, Computer Graphics Group
Am Weichselgarten 9, D-91058 Erlangen, Germany
e-mail: {slusallek,stamminger,seidel}@informatik.uni-erlangen.de

Abstract

In the past, new global illumination algorithms have usually been designed as a single module that was responsible for the simulation of all aspects of illumination in a scene. A recently developed alternative is the design of small and specialized algorithms (lighting operators) together with an infrastructure for creating more complex algorithms by connecting these building blocks – the *Lighting Network*.

In this paper, we discuss the benefits of the Lighting Network approach for designing new and improved global illumination algorithms. Lighting Networks not only provide a flexible infrastructure for new algorithms, they also support a better theoretic understanding of the lighting simulation process. We show that a small number of global light propagation operators already provides the basis for creating many of todays illumination algorithms. Their illumination results are converted into more suitable representations by purely local conversion operators that are specific to an illumination algorithm. Varying the composition of these operators and introducing new elements allows us to create and explore the benefits of new simulation algorithms.

We demonstrate the potential of Lighting Networks with several examples, implementing a diverse set of algorithms, such as density estimation, irradiance gradients, and a composite lighting simulation.

Keywords: composite lighting simulation, global illumination, illumination representation, software design

1 Introduction

A large number of very sophisticated algorithms have been designed for solving the radiance equation [6], thereby computing the global illumination in a virtual environment. The basic algorithms like stochastic ray-tracing [7] or finite-element radiosity [8] have been modified and extended in order to increase their robustness, accuracy and simulation domain, to accelerate the computation, or to reduce their memory requirements [10, 9, 18, 16, 22, 23, 21]. Many of these algorithms are tailored towards specific environments: e.g. radiosity is designed for diffuse inter-reflection, while most Monte-Carlo algorithms are best suited for complex and general environments.

In addition, combinations of the basic algorithms have been developed that exploit the advantages of two or more of these basic algorithms, leading to hybrid or multi-pass techniques [24, 19, 11, 5, 26].

Hybrid methods may offer the best trade-off between the specialized algorithms and the requirements for efficiently computing a global illumination solution for a particular environment. However, existing hybrid methods are fixed combinations of algorithms that make it impossible to adapt them to a different environment. Removing this restriction leads to the idea of Lighting Networks that provide the infrastructure for designing new and improved global illumination techniques by flexibly combining specialized algorithms. In that sense, Lighting Networks are similar to the idea of *building block shaders* [1] but applied to the area of global illumination computations instead of local shading.

In this paper, we explore the potential of the Lighting Network infrastructure for the design of new algorithms. We present a tool-box that consists of two main types of components: light *propagators* and *converters*. The small number of propagation operators are algorithms that compute the transport of light through an environment and are the essential ingredients of the simulation. These algorithms are augmented by a large number of converters that use the data provided by the propagators and convert it into a different representation better suited to the requirements of a specific lighting simulation technique.

As we show below, large numbers of existing global illumination algorithms vary mostly in their use of different converters and illumination representations, while using the same basic propagation algorithms.

The advantages of the Lighting Network approach are not simply the increased flexibility in software design and reuseability of algorithms, but they lie much more in its conceptual simplicity and the explicit structure of algorithms that becomes visible by using the Lighting Network approach. Splitting existing algorithms into distinct components allows for better identifying basic differences and similarities between them. These distinct components then become subject to being redesigned, optimized, or replaced by other existing or new implementations. We discuss several such examples in Section 5 below, after introducing the concept of Lighting Networks and presenting our current tool-box of light propagation and conversion algorithms.

2 Lighting Networks

We start our presentation with a brief description of the basic Lighting Network approach. A more detailed description including extensions and a more formal definition of Lighting Networks can be found in [20].

2.1 Lighting Operators

The basic idea of the Lighting Network approach is to split the computation of global illumination into separate steps. In each step, one algorithm performs a part of the whole simulation process. Each algorithm receives a description of illumination in the scene as input, performs a partial simulation by propagating light, and makes the updated illumination available to other algorithms. In that sense, each of these algorithms modifies the illumination in the scene and is thus called a Lighting Operator (or LightOp for short). By connecting the input and output of different LightOps in the form of a graph, we can obtain more complete simulations of the global illumination.

For a more formal derivation, we take the operator form of the radiance equation $L = L_e + \mathbf{KG}L$, where $\mathbf{K}$ is the local reflection operator and $\mathbf{G}$ is the global propagation operator [4]. Due to the properties of $\mathbf{KG}$, a well-known formal solution of this equation is given by the *Neumann series*

$$L = L_e + \mathbf{KG}L_e + (\mathbf{KG})^2 L_e + (\mathbf{KG})^3 L_e + \ldots . \quad (1)$$

In the Lighting Network approach, we approximate the two operators as sums of operators $\mathbf{K} = \sum \mathbf{K_i}$ and $\mathbf{G} = \sum \mathbf{G_j}$. Each of these terms may describe a term of a reflection models (e.g. diffuse and specular component) or reflection and propagation in different subsets of the scene.

This concept allows us to use different algorithms for computing the light transport, e.g. using a radiosity algorithm for simulating diffuse reflection only, while another LightOp is responsible for the non-diffuse reflection. The lighting simulation then needs to account for the different paths light can take as it is being reflected in the environment. In that sense, light paths correspond to different permutations of the operators $\mathbf{K_i}$ and $\mathbf{G_j}$ in $(\mathbf{KG})^n$.

Each of our propagator LightOps is now responsible for computing a subset of the possible light paths in this environment (similar to [11]). For example, one algorithm could compute the direct illumination of surfaces by light sources, another would compute any number of diffuse reflections between surfaces, and yet another could account for specular or caustic light paths. By connecting the input and output of these algorithms in a suitable manner, we are able to simulate more complex light paths. The concrete choice of the basic algorithms and their connections determines, which light paths are simulated by a particular Lighting Network.

2.2 Representing Illumination

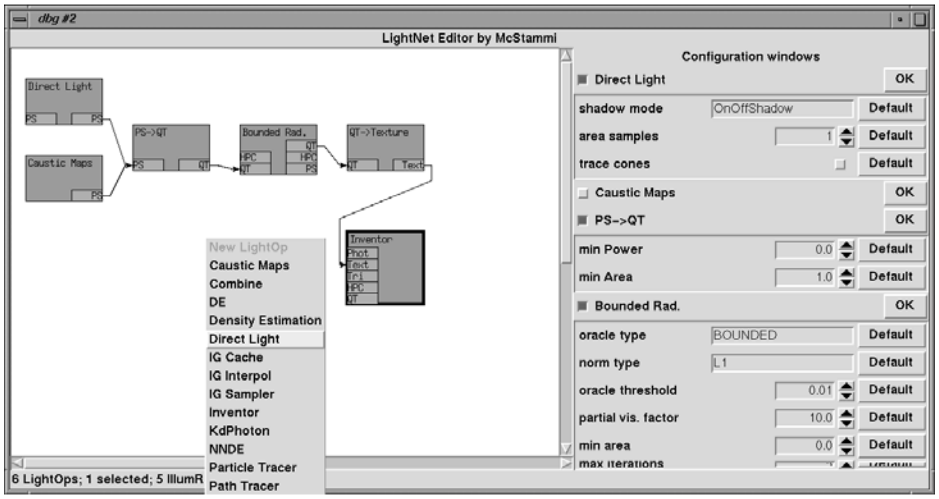
The main issue of the Lighting Network approach is connecting the individual LightOps in order to form a consistent network. We use the concept of a data-flow graph, where illumination information flows between LightOps that form the nodes of a directed and possibly cyclic graph. The graph starts at the light sources defined in the scene, connects the LightOps and ends at nodes that represent the illumination solution computed by the graph.

Because different algorithms work in terms of different representations of illumination, we cannot simply connect any LightOp with any other. Introducing a common unified representation is no option here. Instead, each of our LightOps maintains a list of illumination representations that it can take as input or may generate as output. Commonly used representations are point sampled radiance values or a Haar-wavelet representation of irradiance on surfaces.

With this approach, we can guarantee that connections between LightOps are meaningful and efficient. Because an upstream and a connected downstream LightOp agree on a common representation, the actual flow of information can be made very efficient, introducing almost no overhead compared to a more traditional, monolithic approach. In our experiments the overhead was always well below 5%. Using a visual programming technique, new Lighting Networks can interactively be created as shown in Figure 1.

2.3 Converter LightOps

In cases where we want to connect two LightOps that do not share a common representation, we introduce the concept of a *converter* LightOp. In contrast to a propagator LightOp, a converter does not actually perform any simulation of light transport. It simply converts between two different representations of illumination, taking one



Configuration windows

Direct Light
shadow mode OnOffShadow
area samples 1
trace cones

Caustic Maps

PS->QT
min Power 0.0
min Area 1.0

Bounded Rad.
oracle type BOUNDED
norm type L1
oracle threshold 0.01
partial vis. factor 10.0
min area 0.0
max iterations

6 LightOps; 1 selected; 5 IllumR

Figure 1: The user interface for creating new Lighting Networks using a visual programming approach. A Network with its LightOps and their input and output representations are displayed. Each LightOp can be configured using the user interface components on the right.

as input and supplying the other as output. In the context of the operator notation above, this is equivalent to a basis transformation of L.

It turns out that this converter concept is very powerful and subsumes a lot of processing that is otherwise hidden deep inside current algorithms. For example density estimation algorithms (discussed in more detail in Section 5.1) almost exclusively process raw photon hits and convert them into a more useful representation. Making these conversions explicit allows for reusing them easily in other combinations of LightOps, as well as for optimizing, redesigning, or replacing each of the different converter LightOps separately.

Another example is the conversion between a specific finite element basis (e.g. piecewise constant) to smoother and thus visually more pleasing representation (e.g. using piecewise linear interpolation). By separating the conversion from the propagation of light, we can use the same converters with any of the propagation LightOps (e.g. classical, hierarchical, wavelet, and clustered radiosity, etc.). The same is true for any new and improved converter.

2.4 Processing of Lighting Networks

The processing of a Lighting Network actually starts at the result nodes of the graph. Whenever illumination information is required by a renderer, these LightOps are accessed. The request is then processed by each LightOp usually recursively requesting illumination informations from those LightOps connected to its input. Thus, individual requests are processed in a "pull-model". However, there are also some LightOps that operate in "push-mode", requiring some preprocessing of their input data before individual requests can be processed. Examples of such processing are the solution of a finite-element system or the initial tracing of photons from light sources. This kind of preprocessing is performed by a special recursive preprocessing request [20].

2.5 More Flexibility

While this idea of a network of algorithms may seem like an unnecessary complication compared to a single multi-pass algorithm, it offers a lot of flexibility that we are going to use to our advantage. This added flexibility may also be directly exploited by a user of a lighting simulation system by adjusting the lighting simulation algorithm for a specific scene or visual effect. Also note, that all of this additional flexibility comes at almost no computational costs [20].

Lighting Networks allow for using different algorithms for different parts of the scene. By assigning specific algorithms to only some parts of the scene, we can tune the simulation algorithm for speed, accuracy, or may achieve a special effect with a particular combination of algorithms. Using *tags* assigned to surfaces, each LightOp operates only on some tagged subsets of the environment. One subset is used for input (i.e. light sources for this particular algorithm), one for reflecting light, and a third subset indicating where illumination output should be computed. Of course, these subsets may overlap.

One application of this feature is to restrict the radiosity computation to only some large surfaces in an environment thus ignoring small details. However, the average ambient illumination computed by radiosity is applied to the ignored surfaces. This is a rough approximation, but can result in much faster computations.

This concept also allows for restricting particular costly algorithms to those parts of the environment, where their effects are most noticeable. A more detailed discussion of this flexibility can be found in [20].

In this paper, we concentrate on exploring the benefits of this flexibility for designing new lighting algorithms by combining the basic building blocks in new and sometimes unusual ways.

3 Global Propagation LightOps

The propagation LightOps that are presented in the following section are derived from our previous work in designing lighting algorithms. Revisiting these algorithms, it became apparent that many of them were build on only four basic propagation techniques, which are presented below.

It is an interesting theoretical result that only four propagators cover the requirements of a wide variety of lighting techniques. The major difference between most techniques is in the specific processing of the raw illumination data and its conversion into other representations. These issues are discussed in Section 4.

While the propagation operators are *global* in the sense that they transport light through the entire scene, the converters require only *local* operations. This is an interesting distinction and has important consequences: Only propagators need to deal with the complex issues of visibility. Isolating these difficult issues into separate operators allows to handle them once and optimize them for the specific propagation method used, e.g. using particular acceleration methods and data structures. On the other hand, converters are purely local algorithms, which simplifies them considerably. This will become apparent in Sections 4 and 5.

3.1 Direct LightOp

One of the fundamental propagation algorithms is computing the direct illumination at a point in the environment from a number of point and area light sources. The Direct LightOp only operates on a pointwise representation of illumination, both for input and for output. For each receiving point in the environment it provides an irradiance value as well as a set of incident radiance samples. These values are computed by requesting illumination information from point light sources as well as point sampling the radiance emitted by area light sources. This sampling adapts according to the solid angle of the light source and the received power. Shadow rays are used to compute the illumination incident at the receiving sample.

The Direct LightOp will use the primary light sources in the scene if no other LightOp is connected to it. Otherwise, it uses the data provided on its input to compute the illumination from these secondary area light sources. This could, for example, be used to implement a brute force final gathering technique if connected to the output of a radiosity LightOp.

3.2 Radiosity LightOp

Another fundamental building block in our tool-box is the computation of illumination due to diffuse interreflection using one of the well-known finite element radiosity algorithms. Almost all of our radiosity algorithms exclusively take a piecewise constant representation of irradiance on surfaces as input, so that separate converter LightOps are required to obtain this representation. As a result, these radiosity algorithms do not have to deal with the issues of adaptively sampling primary point and area light sources that may have complex emission characteristics, e.g. due to using RenderMan light source shaders. This is left to the Direct LightOp and a converter that generates a piecewise constant representation (see Section 4 for more details).

3.3 ParticleTracer LightOp

An important class of algorithms uses some form of particle tracing for computing illumination in the environment [2, 3, 11, 15, 14, 17]. We use the ParticleTracer LightOp to provide the basic functionality for these algorithms. Similar to the Direct LightOp, the ParticleTracer either uses the primary light sources or the secondary light sources from its input for generating starting configurations for virtual photons. These photons are then traced through the environment until they are absorbed, recording the hit points of photons on surfaces. The parameters of the LightOp can be adjusted to obtain the required set of photons, e.g. for caustics by only allowing reflections on specular surfaces.

As output the ParticleTracer simply provides a sequence of photon hits together with associated data at these points. Except for the sequential generation of particle hits, no other, possibly more efficient access methods are implemented. Those are provided by specific converters, some of which are described in Section 5.

3.4 Monte-Carlo Path-Tracing

The PathTracer LightOp is yet another basic algorithm employing Monte-Carlo techniques but in the reverse light direction. It again uses either the primary light sources in the scene or the illumination provided as input. They are sampled by stochastically casting rays into the environment from each sample point for which illumination is requested by a down-stream LightOp. In its illumination request, the down-stream LightOp can provide information on the required sampling density. As output the LightOp provides a set of incident sample directions with the radiance from that direction. By selecting a different illumination representation provided by the PathTracer, each sample may also contain other sampling data such as the distance to the emitting surface.

This LightOp may be used directly, simulating illumination using simple hemispherical path-tracing, but it may also generate single radiance samples from a specific direction, or can be used to generate sample sets of the environment for use by other algorithms. An example for the latter use is the Irradiance Gradients algorithm (see Section 5.3).

3.5 Discussion

Because each of the described propagation LightOps computes different light paths through the scene, the four propagation LightOps already cover the requirements of a wide range of illumination algorithms. However, there are some algorithms which would be difficult to map to Lighting Networks as they require too much communication between the different modules.

So far only illumination information is propagated in Lighting Networks. However, some form of importance is already used for requesting illumination from the PathTracer and this approach could be extended also to other LightOps.

4 Illumination Representations and Converters

In the following we present some of the more important illumination representations used in our Lighting Networks and discuss associated converters.

Point Sampling (PS) A point sample describes the incident illumination at a particular point in the environment. It provides the irradiance and a set of directional samples containing the incident radiance.

Quad-Tree (QT) A quad-tree represents irradiance hierarchically using Haar-wavelets on a unit-square domain. The conversion from a quad-tree representation to point sampling is trivial, simply returning the value of the piecewise constant function at the location of the sample.

In the reverse direction, we use a user specified initial sampling on the surfaces for estimating the irradiance. Using a given error threshold, we perform a wavelet compression on the data set using the Haar basis. These conversions also account for the mapping between parametric space to 3D.

TriMesh A TriMesh is a triangle mesh in parametric space that provides irradiance values at its vertices. These are linearly interpolated within the triangles. Converters exist that perform linear interpolation in a quad-tree and output the results as a TriMesh. Another converter can point sample the TriMesh.

Photons The photon illumination representation is mainly used for the output of the ParticleTracer LightOp. Several slightly different illumination representations exist for describing photon hits in the scene. They differ in the data associated with each hit. This representation can then be converted into point sampled irradiance values by using density estimation LightOps (see Section 5.1).

Many of the algorithms presented in the next section use special illumination representations for passing illumination information between different LightOps that together implement the algorithm. These internal illumination representations will be discussed in the examples below. While they are currently only used by LightOps of a specific algorithm, they may also be used by other LightOps, once they proved to be useful in other contexts.

4.1 WireTap LightOps

A special form of conversion LightOps are WireTap LightOps. These LightOps do not even change the illumination, but allow for accessing the data that flows through the operator. Most often these LightOps are used to export or import the illumination data in various formats. One example for their use is as a cache of data values or as a wire tapping method during the development of Lighting Networks.

An important use of this kind of LightOps is for generating intermediate representations of a solution. For example, a special OpenInventor or VRML LightOp can write to a file or display directly the geometric models of the scene illuminated by the data flowing through it (see Figure 4). The effect of individual LightOps in a network can simply be viewed by inserting such a wire tap before and after the respective LightOp. Difference images are generated by a LightOp that computes the difference in the illumination coming from the two wire taps. This simple way of accessing and displaying intermediate results can speed up the development of new algorithms dramatically.

5 Examples

The following section presents and discusses some of the lighting simulation algorithms that have been implemented using the Lighting Network approach. Our aim is to present in several examples the benefits of designing algorithms as a decomposed set of individual LightOps.

The most important benefits of using Lighting Networks are

- *flexibility* for (re-)combining different algorithms in order to simulate new illumination effects,
- *distinction* between the *global complexity* of propagation operators and the reduced *local complexity* of converter LightOps,
- *simplicity* with which existing algorithms can be reused in different contexts,
- *better understanding* of different algorithms by looking at the particular usage of basic simulation algorithms and transformations of their results,
- *potential for optimizing* each of the algorithms separately and for providing specialized versions of them for different environments.

5.1 Density Estimation

An off-line density estimation technique has been presented by Shirley et al. [17]. This technique is inherently a multi-pass process. It consists of a lengthy particle-tracing pass that stores the photon hits off-line. These photons are then sorted by which surface they hit and density estimation is performed on each surface, which reconstructs a piecewise linear representation (represented by triangle meshes) of the irradiance on each surface. A final simplification pass compresses the representation by reducing the number of triangles, while maintaining a high quality representation.

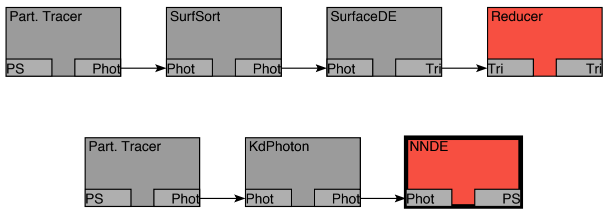


Figure 2: The mapping of the global and nearest-neighbor density estimation algorithms to Lighting Networks. Both implementations act as converters that reorganize and convert input data provided by the ParticleTracer into more suitable representations.

Another density estimation algorithm has been used by Jensen [12]. It uses a two-pass approach that differs in the density estimation phase, where it uses an on-the-fly, nearest-neighbor (NN) algorithm. An improved density estimator has also been presented in [13].

Both the Shirley and Jensen algorithms can directly be mapped to suitable Lighting Networks as shown in Figure 2. In both cases the same ParticleTracer LightOp performs the initial distribution of the photons from the light sources.

The Lighting Network for the PhotonMap algorithms consists of the *NN-Density-Estimator* (NNDE) LightOp that performs the density estimation based on the N nearest photon hits it receives from the *KDPhotonTree* LightOp. In the preprocessing phase, this LightOp sequentially requests the photon hits from the ParticleTracer and organizes them in a kd-tree for faster access during the rendering and estimation phase.

The density estimation algorithm by Shirley et al. is implemented using three cooperating LightOps: The *PhotonSurfSort* LightOp sorts the photon hits by the surfaces that were hit. This sorted data is then made available to the *SurfaceDE* LightOp that requests them a surface at a time and performs the density estimation for each surface separately. The result of the density estimation phase is represented as a TriMesh. This TriMesh is then made available to the *TriMeshReducer*, which uses a mesh reduction algorithm for compressing the TriMesh by eliminating unnecessary triangles.

Please note, that most computations performed by the density estimation LightOps are done by converters. New algorithms can easily replace either the NNDE or the SurfaceDE operators for improving the illumination results, while the rest of the network would stay the same. It is even possible to have different density estimation algorithms adapted to different parts of the scene.

5.2 Composite Lighting Simulation

This example illustrates the potential of the new method by combining several LightOps to form a composite lighting simulation. Several of the algorithms used in this example are candidates for being replaced by other (possibly better) algorithms, some of which will be described in subsequent sections.

In this example, we use three propagation and four converter LightOps (see Figure 4): Both the Direct LightOp and the LightOps computing caustic light paths compute illumination from the primary light sources. The Caustic LightOps uses the particle tracing approach for simulating caustic light paths [12]. The photons are traced from the light sources only via specular reflection and are deposited once they hit a diffuse surface where NN density estimation is performed.

In order to compute the diffuse indirect illumination, the output of the Direct and Caustic LightOps are connected to the PStoQT converter. This converts the sum of direct and caustic illumination into a quad-tree representation as described above, which is then used as input for the radiosity LightOp. After calculating the indirect illumination, the output quad-tree is linearly reconstructed

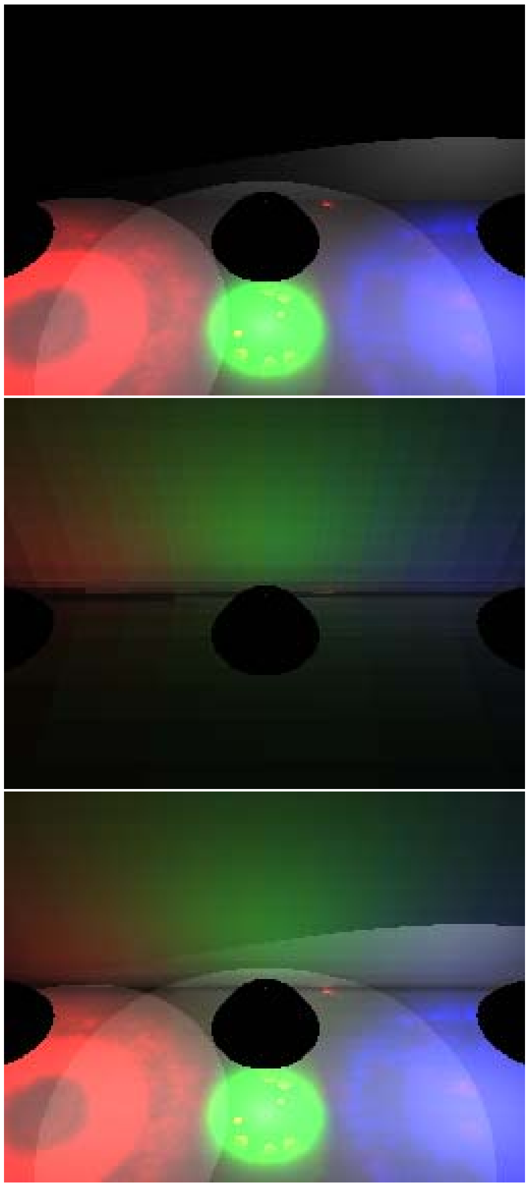


Figure 3: Three spotlights emitting white light housed in colored parabolic reflectors. The direct and caustic illumination can clearly be seen on the bottom surface (top). The center images shows the indirect illumination due to this illumination, while the full simulation is shown at the bottom. The Lighting Network used, is the same as in Figure 4.

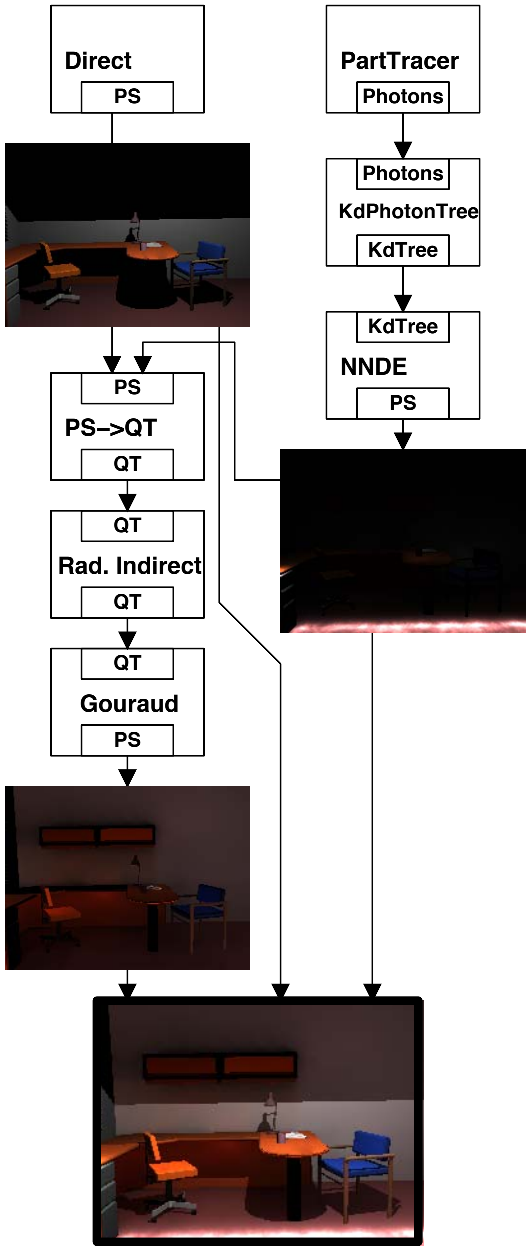


Figure 4: A Lighting Network computing diffuse indirect illumination for both direct and caustic illumination. The caustic illumination is due to a light source with a semi-cylindrical reflector focusing the light onto the floor. The figure shows the network with the different operators. Some wire-tap LightOps are visualized as images and show intermediate illumination results.

with the Gouraud LightOp converter for achieving a better visual quality by interpolating the piecewise constant illumination values. As a last step, the indirect illumination is also added to the final solution.

The new algorithms allow us to include caustic illumination into the radiosity simulation (see Figure 4). Figure 3 shows the same illumination effect more directly in a simpler environment.

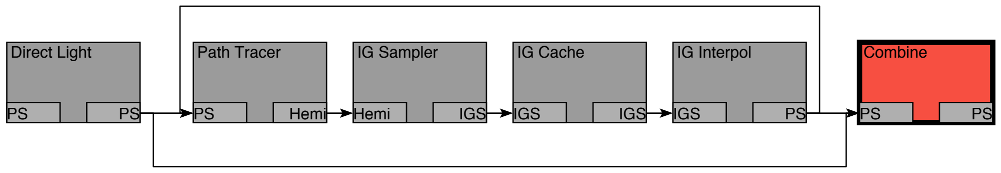


Figure 5: The Lighting Network for implementing Irradiance Gradients. The sampling, caching, and interpolation steps are implemented in separate LightOp.

5.3 Irradiance Gradients

Irradiance Gradients is a well-known technique for computing good estimates of the indirect irradiance in an environment [27, 26, 25]. It is a hybrid method that uses path tracing for propagating illumination, while interpolation, caching and sampling are implemented as converters.

5.3.1 Mapping to a Lighting Network

Figure 5 shows a fine grained mapping of this algorithm to a Lighting Network using four different LightOps. The *IGInterpolator* is the main algorithm that performs the interpolation of nearby radiance samples it requests from its input LightOp. This *IGCache* LightOp either returns nearby cached samples or requests a new sample from the *IGSampler* in case too few samples are available. The IGSampler in turn uses the generic *PathTracer* to trace the rays in order to sample the incident radiance field.

The IGSampler LightOp computes the irradiance and gradient values for the sample and returns them to the IGCache. The PathTracer is connected to an input LightOp, which determines one part of the illumination at each point hit by the PathTracer (direct illumination). The PathTracer is also connected back to the IGInterpolator for recursively obtaining the indirect illumination in the environment. This cycle in the Lighting Network elegantly represents the recursive structure of this algorithm.

While the three specific LightOps could also be combined into a single LightOp, this fine-grained decomposition better supports separate experiments and optimization of individual algorithms. We did that with all three algorithms and were able to improve them, avoiding sampling artifacts (light and shadow leaks) in the IGSampler, speed up cache access in the IGCache, and experimenting with other interpolation techniques in the IGInterpolator.

5.3.2 Final Gathering

The new implementation of Irradiance Gradients has also been used in combination with other LightOps. By eliminating the recursion in Figure 5, the algorithms simply performs a single gathering step of incident illumination. Due to caching and interpolation, the algorithm is very well suited as a “final gathering” step for smoothly interpolating the indirect illumination computed by other algorithms such as a coarse radiosity or density estimation simulation. Used this way, Irradiance Gradients uses only a small fraction of the time of traditional final gathering methods. Also, the Irradiance Gradients approach is almost independent of image resolution.

6 Conclusion

In this paper, we introduced a new approach for designing lighting simulation algorithms. Instead of implementing large monolithic modules that are responsible for all aspects of illumination in a scene, the Lighting Networks approach provides the infrastructure for connecting multiple small algorithms in the form of a data-flow network. The benefits of this approach were demonstrated with several examples.

Probably the most important aspect of Lighting Networks is the better insight offered by decomposing the lighting algorithms into global propagation and local conversion operators. This decomposition allows for optimizing and replacing individual components of lighting algorithms in order to improve their speed or accuracy.

It turned out, that already a small number of basic propagation operators supports many existing lighting algorithms, which spend most of their computation time in converters. This has been a somewhat surprising result, which could change the way we view the development of lighting algorithms in the future.

Lighting Networks seem less suited for implementing tightly integrated algorithms that require a lot of communication between modules. However, these can still be integrated as a more complex single LightOp.

More complex LightOps probably require some form of hierarchical organization of LightOps that allow to treat a group of LightOps as a single macro LightOp. We are currently implementing the full Photon-map algorithm described by Jensen [12]. This requires a non-trivial amount of LightOps and would benefit from Macro LightOps to make it easier to handle. However, most of the LightOps required for this approach have already been described in this paper.

7 References

[1] Gregory D. Abram and Turner Whitted. Building block shaders. *Computer Graphics (SIGGRAPH '90 Proceedings)*, 24(4):283–288, August 1990.

[2] A. Appel. Some techniques for shading machine rendering of solids. *AFIPS 1968 Spring Joint Computing Conference*, pages 37–49, 1968.

[3] James Arvo. Backward ray tracing, developments in ray tracing. *ACM Siggraph '86, Course Notes*, pages 259–263, 1986.

[4] James Arvo, Kenneth Torrance, and Brian Smith. A framework for the analysis of error in global illumination algorithms. *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 75–84, 1994.

[5] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglas Turner. A progressive multi-pass method for global illumination. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):165–174, July 1991.

[6] Michael F. Cohen and John R Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press, 1993.

[7] Robert L. Cook, Tom Porter, and Loren Carpenter. Distributed ray tracing. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):137–145, July 1984.

[8] C. M. Goral, K. E. Torrance, and D. P. Greenberg. Modeling the interaction of light between diffuse surfaces. *Computer Graphics (SIGGRAPH '84 Proceedings)*, 18(3):212–222, July 1984.

[9] Steven J. Gortler, Peter Schröder, Michael Cohen, and Pat M. Hanrahan. Wavelet radiosity. *Computer Graphics (SIGGRAPH '93 Proceedings)*, 27:221–230, August 1993.

[10] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):197–206, 1991.

[11] Paul Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 145–154, aug 1990.

[12] Henrik Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Rendering Techniques '96 (Proceedings Seventh Eurographics Workshop on Rendering)*, pages 21–30. Springer, June 1996.

[13] Karol Myskowski. Lighting reconstruction using fast and adaptive density estimation techniques. In Julie Dorsey and Ph. Slusallek, editors, *Rendering Techniques '97*, pages 251–262. Springer, June 1997.

[14] S. N. Pattanaik. *Computational Methods for Global Illumination and Visualization of Complex 3D Environments*. PhD thesis, Birla Institute of Technology & Science, Pilani, India, February 1993.

[15] S. N. Pattanaik and S. P. Mudur. Computation of global illumination by Monte Carlo simulation of the particle model of light. *Third Eurographics Workshop on Rendering*, pages 71–83, May 1992.

[16] H. E. Rushmeier and T. E. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. *Computer Graphics*, 21(4):293–302, 1987.

[17] Peter Shirley, Bretton Wade, Philip M. Hubbard, David Zareski, Bruce Walter, and Donald P. Greenberg. Global illumination via density-estimation. In *Rendering Techniques '95 (Proceedings Sixth Eurographics Workshop on Rendering)*, pages 219–230. Springer, June 1995.

[18] François Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), September 1995.

[19] François X. Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):335–344, July 1989.

[20] Philipp Slusallek, Marc Stamminger, Wolfgang Heidrich, Jan-Chrisitan Popp, and Hans-Peter Seidel. Composite lighting simulations with lighting networks. *IEEE Computer Graphics and Applications*, 18(2):22–31, March 1998.

[21] Marc Stamminger, Philipp Slusallek, and Hans-Peter Seidel. Bounded radiosity – illumination on general surfaces and clusters. *Computer Graphics Forum (EUROGRAPHICS '97 Proceedings)*, 16(3), September 1997.

[22] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques (Proceedings Fifth Eurographics Workshop on Rendering)*, pages 145–167, Darmstadt, June 1994. Springer.

[23] Eric Veach and Leonidas J. Guibas. Metropolis light transport. *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 65–76, aug 1997.

[24] John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):311–320, July 1987.

[25] Gregory J. Ward. The RADIANCE lighting simulation and rendering system. *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 459–472, July 1994.

[26] Gregory J. Ward and Paul S. Heckbert. Irradiance gradients. In *Third Eurographics Workshop on Rendering*, pages 85–98, May 1992.

[27] Gregory J. Ward and Francis Rubinstein. A ray tracing solution for diffuse interreflection. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):85–92, August 1988.