Efficient Raytracing Methods for Light Simulation

Andreas Bielawny, Ulrich Linnemann,

Brandenburg GmbH, Technologiepark 19, D-33100 Paderborn

Für die lichttechnische Simulation in der Entwicklung von Leuchten und Scheinwerfern sind verschiedene Rechenmethoden verfügbar. Im Bereich der Darstellung optischer Oberflächen kann grundsätzlich zwischen exakten und approximativen Methoden unterschieden werden. Dieser Artikel soll einen Überblick verschaffen über drei verschiedene Verfahren zur Strahlverfolgung (genauer: non-sequential raytracing), die gegenwärtig u.a. in der Software LucidShape Verwendung finden.
Spezielles Augenmerk liegt hierbei auf der leistungsfähigsten Methode, dem „GPUtrace“. Hierbei wird die Computerhardware eines Grafikbeschleunigers verwendet, um die Rechengeschwindigkeit - vor allem für hohe Strahlzahlen - um ein Vielfaches zu steigern. Anhand exemplarischer und anschaulicher Lichtmodelle werden Verfahren verglichen.

For optical simulations in lighting development there are different mathematical methods available. Among procedures for the representation of optical surfaces for non-sequential raytracing we can distinguish between exact and approximative methods. In this paper, we will give a short summary on three methods which are currently applied in the LucidShape software package, with emphasis in the GPU trace feature. This powerful method allows the use of graphics hardware to speed up the simulation process drastically, especially for large numbers of rays to be traced. We discuss four examples of lighting solutions to demonstrate the improvements in simulation time.

A Short Introduction to Raytracing Simulations

In raytracing simulations, there are certain processes that consume more time than others. Especially intersections do cost a lot of time. Intersections occur always when a ray is hitting an optically relevant component, i.e. a geometry that has been assigned an optical material property. The way, intersections are handled is directly linked to the way in which the surfaces are treated in a simulation experiment. Thus, most calculation time is spent on solving the intersections. Naturally, this leads to some conclusions about the impact of a model's complexity on simulation time.
The duration of a simulation is in the dimension of time, while the speed of a simulation is the amount of rays that are traced in this amount of time. Some experiments do not require a large number of rays (e.g. < ~10^7 rays), others intrinsically require a lot of "light" – sometimes very large ray numbers. The latter are typically considered "overnight" or "weekend" candidates for common off-the-shelf PC hardware. The complexity of a virtual lighting setup is defined by several factors, such as the outline of optical components, as well as the surface material types of the interfaces involved. Another factor which is directly influencing the calculation time of a simulation task is the average number of intersections for each ray.

The most simple situation consist merely of a light source and a sensor, an example being a bulb in front of a wall. Ignoring the light source itself for now, we count just one single intersection: the one with the sensor. Let us call this the intersection No. 0 (zero).
Adding a reflector to such a setup increases the intersection count to 1. Although very simple, this is a common situation in lighting development.
The more components an experiment contains, the longer the trace will take for each ray to trace. Extreme examples with very long simulation times are those that either require many rays or those that expose the rays to a multitude of intersections. Prominent examples for complex and thus slow models are light pipes and backlighting applications in general. But also experiments that allow parasitic multiple internal reflections will suffer noticeably from increased simulation times.
Simple reflector systems, single lens setups, or especially redundant systems which do not require the simulation of the complete lighting unit (such as multi LED with identical light engines) in contrast can be considered fast models.
Raytracing speed is obviously important for complex and slow models, while a simple setup can already be solved in a couple of minutes (or even seconds) with a sufficiently large number of rays.

Also, the focus of interest of the light engineer plays an important role.
While designing a general outline of a lighting system, only a coarse sensor resolution may be sufficient. This results directly in a small number of rays required to achieve a decent signal to noise ratio (SNR) in the results: fast simulation! Fine details, small artefacts and all simulations in the final phase of development are generally executed with large numbers of rays, since fine sensor resolutions are being used (keep up a good SNR).
Logical consequence: using a sensor with a resolution (inverse cellsize) that is too fine for the question at hand is a waste of precious (simulation) time. Using an insufficiently low sensor resolution of sacrificing the amount of rays traced for a speed up of the simulation on the other hand is very risky – results may be subject to statistical variations (noise) or similar deviations or may just be utterly wrong.

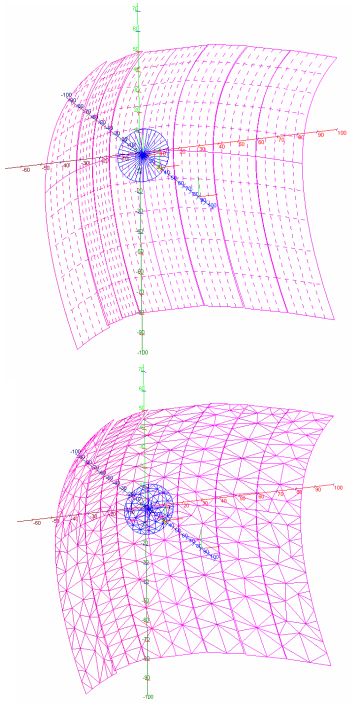
The perfect simulation parameters are always those that will clearly process a certain result within the minimum amount of time possible while achieving the required precision.

It is mandatory to note that we discuss here only the principle methods of simulation. Still, differences between specific algorithms and individual implementations will account for a great deal of variations in precision and calculation time – even among programs that use the same methodological class of simulations methods. All results regarding actual simulation times in this paper solely refer to the LucidShape Software.

NURBS grade precision

The highest level of precision in determining the optical response from a surface is achieved only by calculating the exact position of an intersection as well as the exact normal of the surface hit in that particular location. In order to achieve this, an understanding of NURBS[1] mathematics is required, as well as a some basic mathematical considerations. Once these problems have been solved, the results from any simulation are based on the highest possibly achievable precision. This method is sometimes referred to as “analytic” grade or “accurate” simulation. Unfortunately, the effort made to create this precision does cost a measurable share of calculation time, which makes all exact methods relatively slow in comparison with other raytracing methods.

Tessellation / meshing / approximation methods



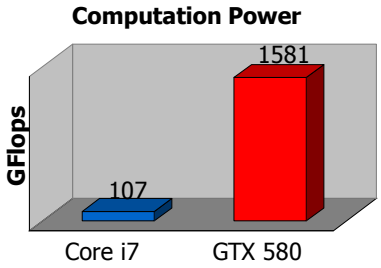
The tessellation or meshing of the surfaces in a model is a common method in computer graphics and simulation. In a multitude of different applications, meshing is used to define the precision and resolution of a simulation in trade-off for calculation speed. The option to decide what to emphasize is the main feature of all approximation methods of this type.
All geometric surfaces are broken down into a triangular mesh. This is done using different heuristic approaches and weighting algorithms to create a minimalistic yet sufficiently smooth mesh of all surfaces in a lighting model. Limiting the smallest building blocks and the largest deviations from the original NURBS surface define the precision of the meshed surface replica. The figure to the right shows a simple tessellation of a reflector surface.
One very interesting aspect of efficient tessellation procedures is the relation between the actual number of triangles in which an experiment is divided and the resulting simulation time for the raytrace. This relation is non-linear but of logarithmic nature. Thus an increase in the number of triangles will result in a much smaller increase in simulation time - without further drawbacks on the gain in surface precision due to the finer meshing.
It is helpful to calibrate any approximative method by comparison to a precise method. Tessellation parameters should be refined until the results meet the desired agreement with those light patterns obtained in a preliminary NURBS grade simulation run.

[1] NURBS: non-uniform rational B-spline

GPUtrace

The latest simulation method “GPUtrace” is a special variant of the tessellation principle. The actual variation are not so much of systematic nature, but rather found in the interplay of software and hardware.
GPUtrace method makes use of the graphics processors of modern graphics cards. These

GPU (Graphics Processing Unit) in contrast to the CPU (Central Processing Unit) of a computer does not consist of a small number of powerful, flexible, and omnipotent workhorses (the cores), but of a large number of highly specialized graphic processors. These graphics cores (CUDA cores) for example on a state of the art nvidia card[2] can be combined in clusters of 512 - a very large number when compared with 2, 4, or maybe 8 cores of a CPU. The specialty of these CUDA cores is their ability of massive parallel processing.
A rather general command set allows more than the basic geometrical operations of older graphics card types. Clusters of CUDA cores can compute blocks of problems together, providing a large number of threads. GPU trace uses the CUDA core (or graphic processors in general) to execute the time consuming calculations on the graphics card. Additionally, the CPU still has to do some part of the work, including feeding the GPU with calculation jobs. Thus, only a combination of halfway decent CPU hardware with a fast graphics card is really efficient in the end. In comparison of two specific CPU and GPU, a state of the art GPU can provide about 15 times the computation power (switching processes per time) of a modern CPU.

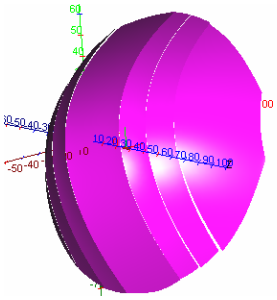


Since the GPU trace requires some preparation time in dependence of the number of triangles an other factors, relative speed increase is higher at large numbers of rays.

GPUtrace examples – simulation time

We have investigated four examples of lighting models with focus on their simulation times. To compare apples with apples, results from Tessellation and GPUtrace will be shown.

Single reflector, macrofocal type

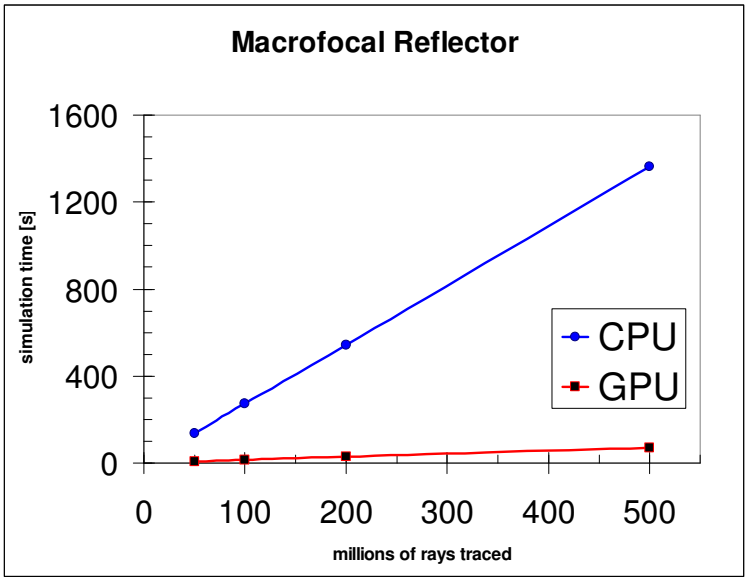


The first example consists of a single reflector surface of MF (macrofocal) type, taking into account the physical extension of the light source using the edge-ray principle. This is an example for reflector-based front lighting situations, especially the low-beam with its precise cut-off line, as well as signal applications in automotive lighting. Classical reflector lights in general lighting also belong to this type of simulations. This model shows mostly single interactions for each ray (reflected only once).

[2] All NVIDIA gtx and tesla cards are fully compatible with GPUtrace and are being recommended

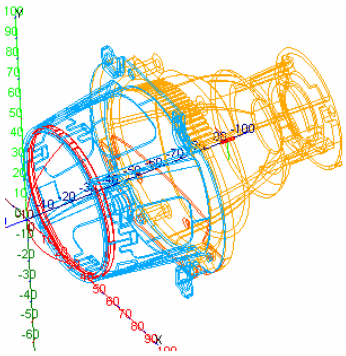
Less accurate meshing will show already relatively good results for the overall pattern evaluation.

Number of triangles: ~ 6.4x10^4
Max. acceleration measured: x17



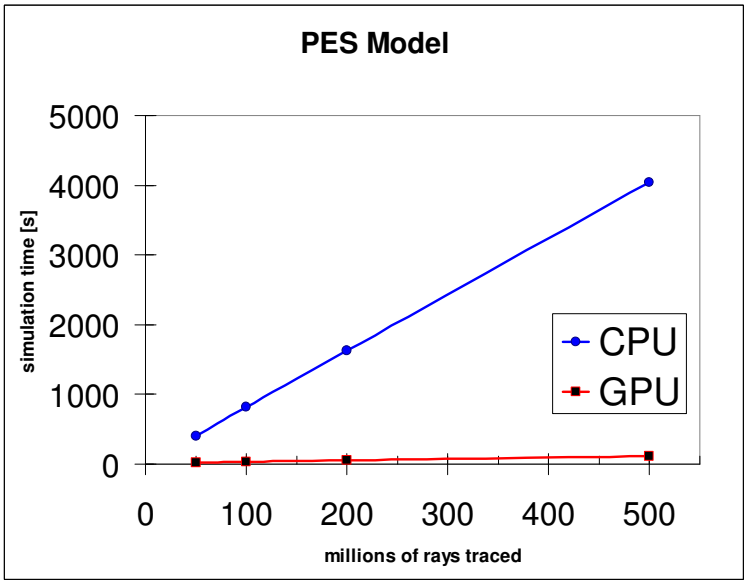
The GPUtrace will be able to speed up this type of simulation by a factor of about 17.
The graph shows the simulation time as a function of the numbers of rays traced. The relation is linear, but with a bias for the preparation time of GPU trace.
The CPU clearly needs more time, which will become dramatic for large ray numbers in ultra-precise simulations.

Poly-Ellipsoid-System (PES), projector unit



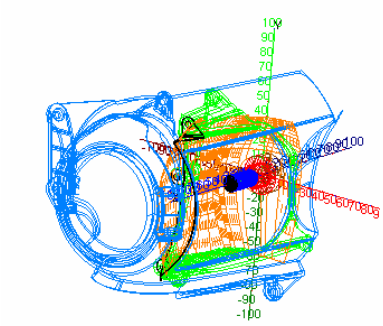
Another automotive frontlighting system, the PES is combining a poly-ellipsoidal reflector for light capture and for shaping the light pattern with an aspheric lens for projection onto the road and a shield for the creation of the cut off line. Thus, it exposes the rays to a minimum of 3 intersections (1x reflector, 2x lens). Additional parts, such as the housing are included to obtain a realistic optical output with all possible influences of stray light or multiple reflections. PES type headlamps are a unique class of lighting systems, closely related to other projector setups, such as beamers or gobo lights.

Number of triangles: ~ 10^6.
Max. acceleration measured: x34



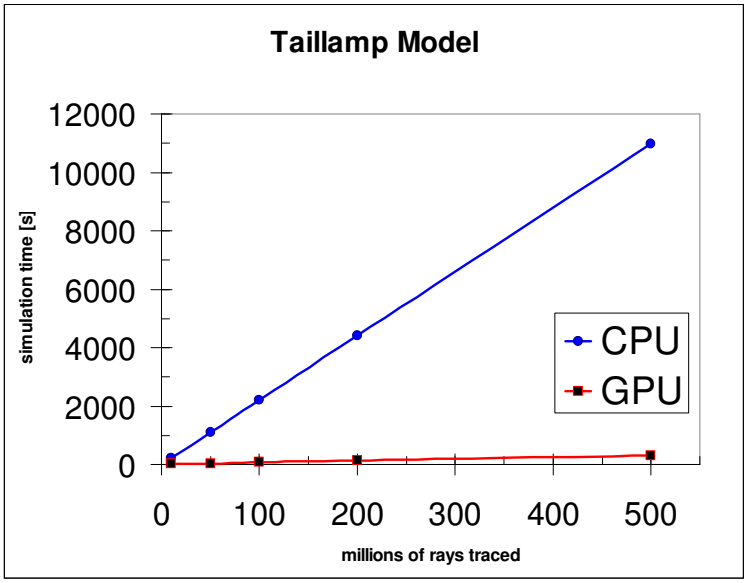
GPUtrace speeds up this type of simulation by a factor of about 30.
The higher number of intersections allows for a greater speed advantage. This is to some extend compensated by the need for high precision. Especially the imaging properties of the lens have to be precise to correctly project the light pattern on the road. Surface errors (in terms of meshing artefacts) in the representation of the reflector and especially the A surface of the lens would otherwise add up.

Tail lamp model, integrated



Automotive lighting in the rear of the car is not so much focussed on the less demanding lighting function (when compared with frontlighting), but strongly emphasizes the lit appearance of the lights. Nevertheless, chromium-look parts and housing-integrated setups require a high depth of details in many cases.
Number of triangles: ~ 1.6x10^6.
Max. acceleration measured: x37

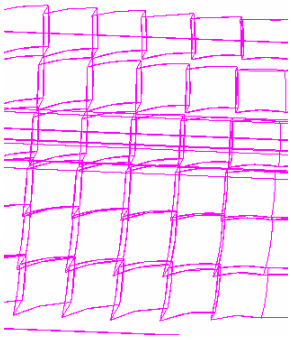
GPUtrace speeds up this type of simulation by about x35 times.
In this classic automotive tail lamp we find a halogen bulb, a reflector and a red PMMA lens. The model includes the optical parts, as well as a full geometric model of the light source and the relevant absorbing or diffusely reflecting parts of the housing.
Accurate meshing was used to correctly shape the bulb and fine styling details.
Large ray numbers are being used for luminance evaluation here.



Lens for LED center high mounted stop light (CHMSL)

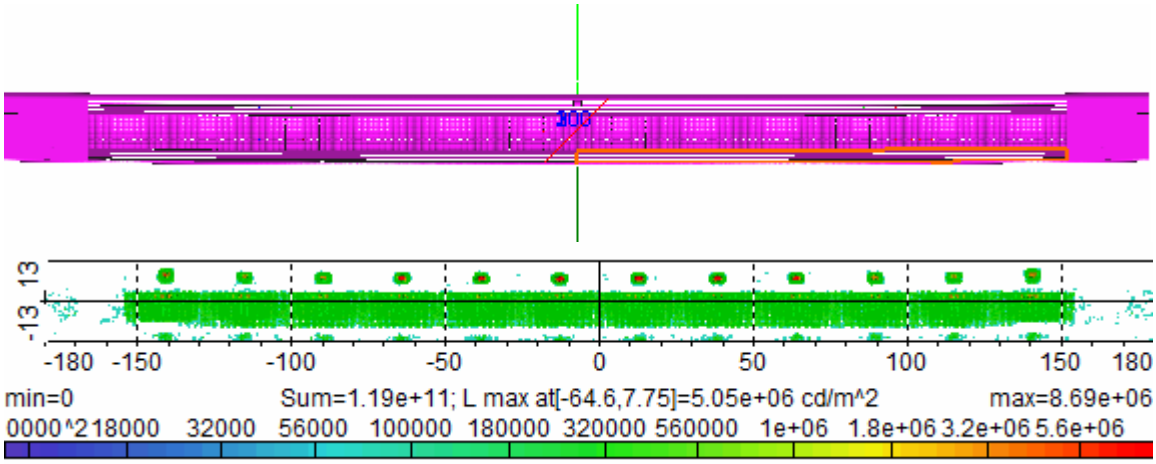
LED CHMSL have been in use already for quite a while. Here, a cluster of 12 LEDS is using a single lens part with facetted surface to create the light pattern. This is an archetype of lens-based multi-LED setups without individual optics.

Number of triangles: ~2x10^6
maximum acceleration measured: x26



GPUtrace typically speeds up this type of simulation by a factor of about x19.

Rather sharp edges and variations in curvature require a fine meshing here. Cross-talk between the emitters and neighbouring lenses as well as the rear of the light (PCB etc.) suggest also rather large ray numbers here. Raytracing with up to 2x10^9 rays has been performed with this model.
Especially investigations of luminance uniformity is being done with light of this type (see last images). Large numbers of rays allow for reliable results.



Geometric View into the CHMSL (top) and off axis Luminance image (bottom) with direct view on the LED emitters and very smooth and uniform light spread from the lens.

Summary

Among today's simulation methods for lighting design, the best procedure for a certain task can be chosen. In terms of calculation speed however, the GPUtrace is the most powerful technique so far, allowing time saving factors of up to 30x, depending on hardware configurations. It can be run on low-cost hardware (off-the-shelf nvidia graphic cards for gaming) and thus compliments any existing hardware cost-effectively.
This technique is constantly being improved and will of course take full advantage of all future developments in graphics hardware.