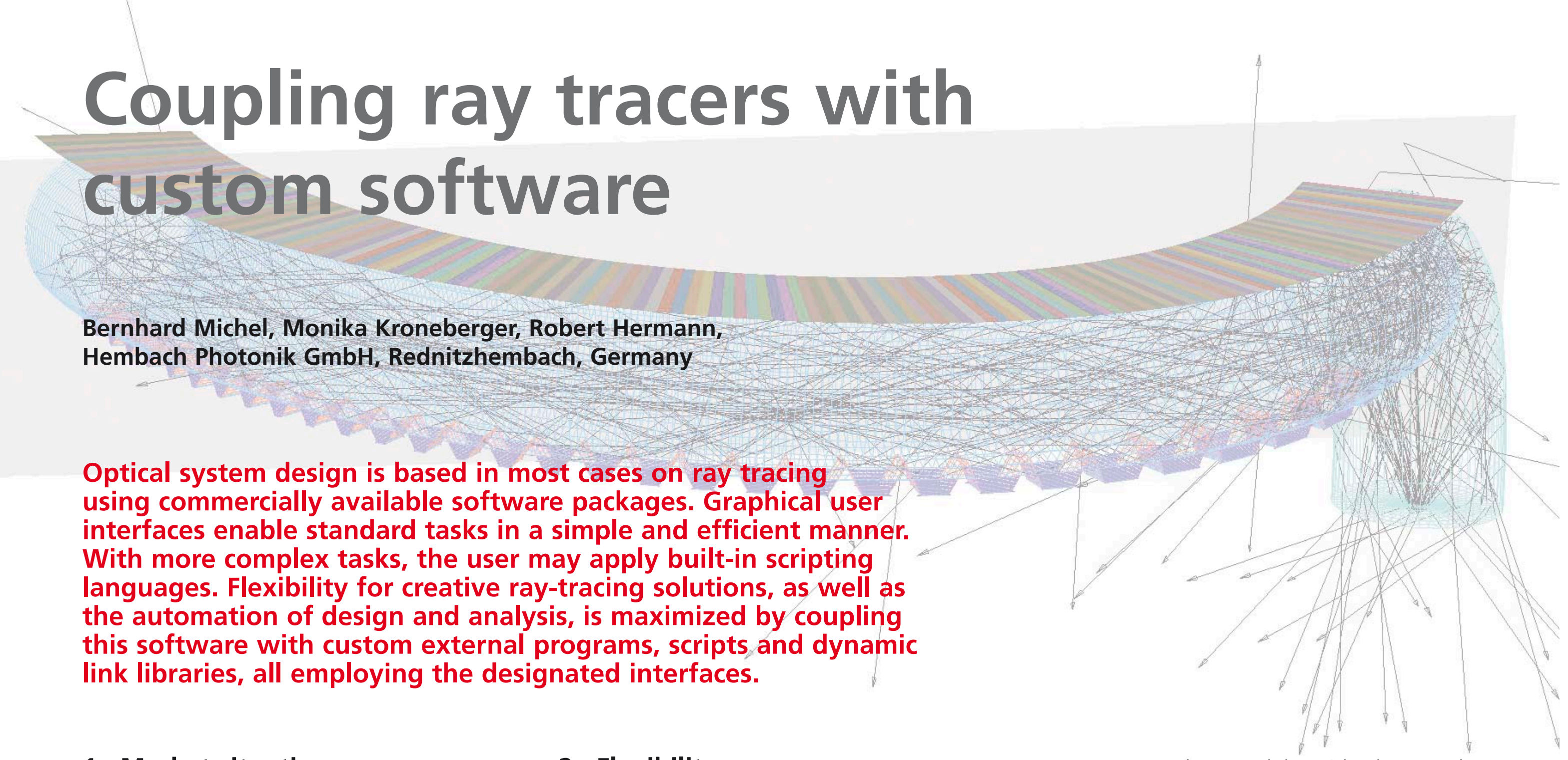
Coupling ray tracers with custom software



Bernhard Michel, Monika Kroneberger, Robert Hermann,
Hembach Photonik GmbH, Rednitzhembach, Germany

Optical system design is based in most cases on ray tracing using commercially available software packages. Graphical user interfaces enable standard tasks in a simple and efficient manner. With more complex tasks, the user may apply built-in scripting languages. Flexibility for creative ray-tracing solutions, as well as the automation of design and analysis, is maximized by coupling this software with custom external programs, scripts and dynamic link libraries, all employing the designated interfaces.

1 Market situation

Ray tracing has become the standard method for the computer-based development of optical systems. Whereas so-called sequential ray tracers are used for the design of imaging systems, non-sequential ray tracers are more suitable for a general class of tasks, in which the physical correctness of the model is in the foreground. Powerful commercial software packages, for instance, ASAP, FRED, LightTools, LucidShape, SPEOS and ZEMAX, support the work of the optical designer.
The target market of such non-sequential ray tracers has expanded considerably in the past two decades. It grew beyond core optical industry, aerospace and automotive industries to reach industrial branches that the inveterate optical engineer would not expect – namely to any application in which light needs to be manipulated. For example, "white goods" or major household appliances are equipped with increasingly complex display elements and signal lamps, all of which must be illuminated according to specifications. These requirements are satisfied by backlight systems and light guides, which often pose challenging development tasks.
Commercial ray tracers have been adapted to this wide market; they now offer intuitive graphical user interfaces (GUI) with dialog boxes, wizards and other interactive elements. Future adaptations require, that ray-trace software producers anticipate potential applications. This is why the GUI is generally limited to standard tasks.

2 Flexibility through custom software

Success with specialized tasks unforeseen by the ray-trace software producer relies on not only more flexibility in using the software, but also greater optics know-how from the user. Many commercial ray tracers provide a scripting language to enable success. Scripts offer advantages like re-usability, modularizing of projects, and – extremely important in complex projects – the option of comprehensive documentation and transparency.
Ideally, a scripting language should be easy to learn, offer all typical elements of a programming language, and allow for the access of actual software functionality on more than one level. For example, geometry elements could be built and manipulated, single rays could be traced on the lowest level. And on a higher level, it would be possible to control whole simulations. However, many scripting languages do not come close to this ideal in reality: either they are complicated and cryptic or they only allow for restricted access to the core functionality of the software.
It is possible to maximize flexibility and operability in ray tracers by coupling them together with customized programs (if indeed interfaces are available). In general, coupling ray tracers with external programs can be done in different ways:

- The pre-processing (e.g. the manufacturing of the geometry of the systems, definition and design of the light sources, etc.) is done outside of commercial ray-tracing software. This is often the only acceptable way of "feeding" the ray tracer complex models with thousands of objects or special light sources.
- Special scripts take on the post-processing where usually giant amounts of data are rapidly produced by the ray tracer. These data are then thinned out for a desired tolerance or stray light analysis to finally yield a compact result. A 'pass/fail' analysis is an extreme case, where elaborate simulations with many gigabytes of data lead to one single bit: compliant or non-compliant with specifications. Another motive for external post-processing is the option of analysis procedures, many of which are not supported by the ray tracer itself.
- Pre- and post-processing governed by scripts, which operate as clients, such that the ray tracer controls them as a server. Such architecture provides a good alternative for optimizing a system, especially when the integrated optimization tools from the ray tracer are not suitable.
- Or else done the other way around, where the ray tracer calls up specialized programs for tasks it cannot solve. A classic example for this is ZEMAX: the user first creates dynamic link libraries (DLLs) in the programming language C and then binds them to ZEMAX. Whenever adequate programming knowledge is provided, the functionality of the ray tracer improves greatly. Other programs such as ASAP allow for such function extensions, too.

Generally, most ray-trace software providers tend to integrate more and more functionality into their subsequent software versions to offer even more (pre-built) solutions. This surely makes sense.

Originally published in German in Photonik 3/2012

Clearly, the functionality of ray tracers is enhanced through combination with special programs, and such combinations are increasingly attractive and efficient in sophisticated projects. There are numerous advantages once you create the structure of a combined process. Software adaptions can be done by the user when necessary, no longer dependent upon the development plans of the ray-tracer producers. Furthermore, the user gains command of a greater wealth of experience and solutions towards the creation of scripts and programs than any software producer is able to offer. For every standard programming language there are extensive software libraries, such as the GAMS Index (gams.nist.gov), the Netlib Repository (www.netlib.org), or also SciPy (www.scipy.org), which provide powerful numerical algorithms.

3 Software selection and application

Which software is best suited to an application? You could devise your own C++-programs, but this route presents the problem of insufficient flexibility. In most cases use of a standard scripting language or software that supports scripting enables a better result. MathLab and Mathematica are popular for in-house projects because both are powerful programs. In multi-partner collaboration projects, problems with compatibility arise, because all partners now have to use several commercial software products, as well as use them together. Our experience has found it better to revert to available scripting languages. We suggest the scripting language Python [1] for several reasons:

- Python is freeware.
- Python offers all structural elements of modern programming languages, yet is easy to learn.
- Python provides extensive software libraries from the start. The additional modules NumPy and SciPy offer a high-quality library of numerical algorithms for optimization, statistical data evaluation and many other applications.
- Many scientists publish their work in Python over the internet, enabling access to a variety of individual concepts.

Below two applications are introduced, where coupling ray tracers with specialized programs yields advantages. The examples, while deliberately kept easy, nonetheless show the basic idea of the coupling approach. We use ASAP and one self-developed ray tracer. The core idea here is precisely that this coupling method also works with most of the other ray tracers.

3.1 Circular light guides

Light guides with various outcoupling surface shapes are frequently used as aesthetical design elements, especially in the automotive industry. The circular light guide shown here was adapted from the daytime-running light example in the automotive sector. This light guide consists of a torus with bent off ends; the light is coupled into the guide from LEDs (**figure 1**). The outcoupled light eminates from a ring having prism-shaped indentations at the side opposite to the observer (in figure 1 below). Gaps and working angles can be adjusted individually for each prism; those gaps and angles affect both the strength and the direction of the light emission. The optimization task for this structure (figure 1 above) is varying the gaps of the prisms, as well as their angles as function of the position on the light guide, to achieve highest efficiency and homogeneity of the luminance (figure 1 below). The geometry is rather simple, yet has to be completely parameterized to enable automated optimization.

The optimization uses a Python script in the simulation of the light guide. The first step is the creation of a parameterized model in ASAP. Only one half of the light guide has to be modelled due to symmetry of the system. In our experience Bezier polynomials – whose coefficients we adjust in the optimization – are well-suited to parameterize the position of the prisms angles and the prisms distances.

To rate the quality of the light guide, we define a merit function to evaluate the emitted light flux together with the homogeneity of the light distribution.

The result will always be noisy as the simulation was performed with ray tracing; Furthermore, the prisms have only a limited size. Both cause fine scaled inhomogeneities of the light distribution. This statistical noise, however, is only an artefact of the simulation and the granularity of the prisms is not relevant here because the eye of the spectator is not able to resolve those small structures from a distance. We expand the light distribution along the light guide into Legendre polynomials in order to evaluate the interesting large scale homogeneity [2]:

$$L(x) = \sum_{l=0}^{N} a_l P_l(x) \qquad \text{(Eq.1)}$$

Here, L is the luminance (at normal observation), x the (properly scaled) location on the light guide and P_l are the Legendre polynomials. The coefficients a_l are best determined with a modified Monte Carlo simulation process called "Importance Sampling" [3] directly from the rays which are calculated by the ray tracer. Only a_0 would be different from zero for ideal homogeneity; all other coefficients "measure" the deviation from the perfect condition. The series is truncated at the relatively low order of N ~5 – 6 to largely filter out the aforementioned fine scaled inhomogeneity.

In our experience, one needs considerably less rays when this trick is used in Python. This same trick is not easily done within a ray tracer. The use of Python requires less calculation time than the "normal" method, often employed by the ray tracer. We use "Simulated Annealing" for optimization; almost ready-to-use Python-scripts are available as freeware via the Internet for either this method or other optimization algorithms, which are as good as "Simulated Annealing". Herewith, the ray tracer – ASAP in our case – will only be used as "Server" for its main purpose; everything else will be handled by the "Client" Python.

The result of the optimization is very clear: **figure 2a** shows the distinctly structured light intensity distribution of the initial geometry, which also greatly mitigates toward the centre of the light guide. The same can be said for the luminance (**figure 3a**). After the optimization, only small structures disrupt the irradiance (figure 2b) and also a homogenous luminance can be achieved (figure 3b).

This procedure can also be used for significantly more difficult light guides and basically works with all current ray tracers.

3.2 Diffusor design

One requirement in backlights and light-guides is a uniform luminance at the exit surface. To achieve this result, it is necessary to use not only light guiding elements like reflectors, lenses, and

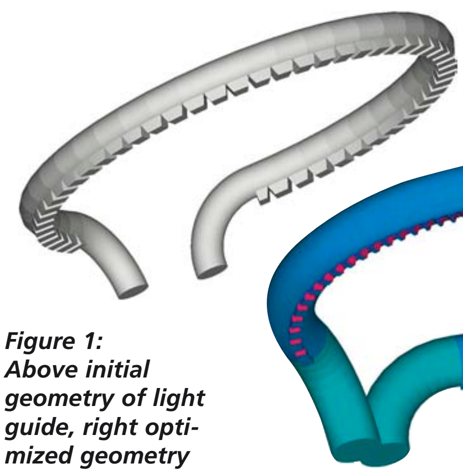
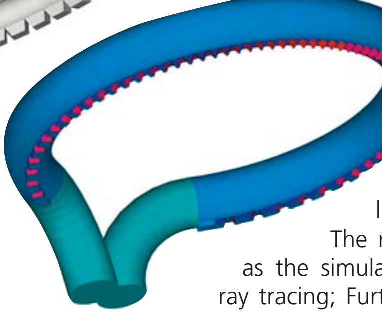



Figure 1: Above initial geometry of light guide, right optimized geometry

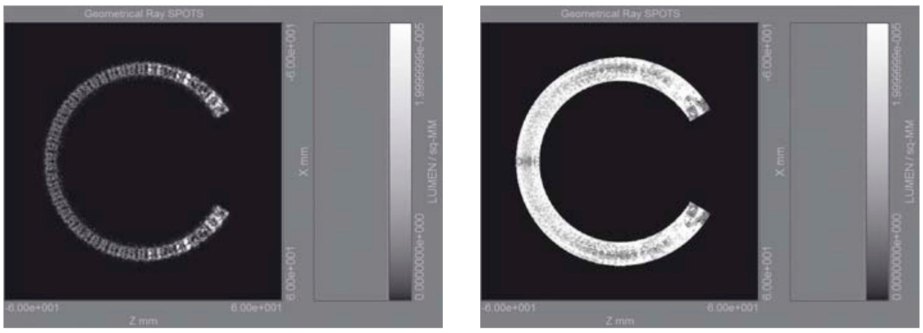


Figure 2: Illuminance distribution of light guide, a) initial geometry, b) optimized geometry

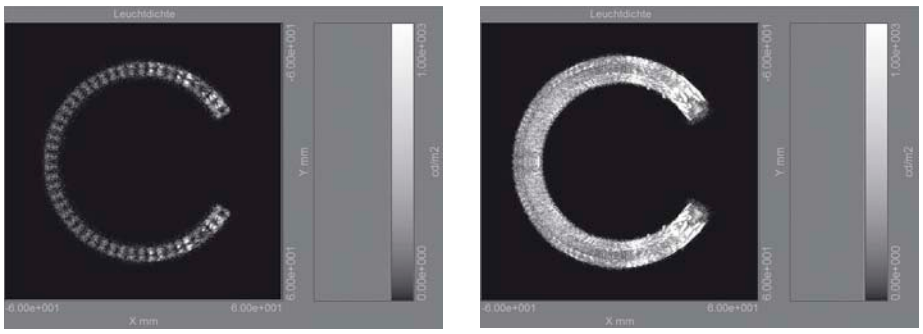


Figure 3: Luminance picture of lightguide, taken with a virtual luminance camera, a) initial geometry, b) optimized

Fresnel-structures, but often also diffusers that homogenize the light with respect to its directional distribution. Apart from using diffuser sheets to spread light, there is a significant increase in use of volume scattering materials, where small particles are embedded in the matrix material. Its parameters are the diffuse and the total transmission coefficient – the latter also includes the direct, unscattered light – as well as the angular distribution of the transmitted light.

These parameters are dependent on both the material thickness and the refractive index of the matrix and the particles, as well as their size distribution and concentration. If the particles are assumed spherical, it is then possible to compute the optical properties of the volume through Mie-theory and radiative transfer theory ("forward calculation"). Most of the commercial ray tracers are able to do that (see e.g. [4]) as well as self-developed software, which can deliver maximal flexibility in modelling the diffusion, though mostly restricted to simple geometry.

The more interesting question is whether it is possible to reverse the simulation direction ("backward calculation"): Is one able to deduce the composition/recipe of the material when the scattering intensity and transmission of the diffuser are known? Frankly, it is not always possible and the computed recipe is ambiguous in most cases. The latter fact offers advantages as certain freedom in design is gained: Users are in the position to evaluate the possible designs for producibility, the availability of the materials, and the production costs, and then select the best.

To explore the design scope, in **figure 4** we examined the influence of different size distributions, concentrations and refractive indices of the particles on the scattering intensity. We assumed PMMA for the matrix and 4 mm thickness of the material. In general, large particles result in a strong forward directed scattering while smaller particles broaden the scattered light distribution. An increase in the contrast of the refraction index (e.g. using TiO_2) highly raises the scattering to large angles. The simulation of the diffuser was realized in different ways:

1) As a model in completely in ASAP
2) A self-developed program computes the so-called phase function of the diffuser in the framework of Mie-theory. An ASAP-model reads this phase function through a special DLL and performs the simulation.
3) Complete modelling with self-written program

While each of these approaches leads to the same results, maximal flexibility can be acquired through the second approach, and the quickest is the last. Computing the phase function in our own software also has the benefit that the results can be read by different ray tracers (e.g. SPEOS); this way a library of phase function can be built which is entirely software independent. The algorithm to find the optimum recipe (concentration, size distribution, and refractive index of the particles) was solely implemented in Python. Merely small changes are required at the interfaces to the ray tracer to implement each of the three aforementioned simulation approaches.

The simulation is done as shown in **figure 5**. The input parameters are passed to a program which computes the phase function. This function will then be read from the ray tracer. The calculated angular distribution will be compared to the nominal distribution and rated with a suitable merit function. The recipe will then be fitted (through new input parameter sets) until the nominal value and actual values coincide within a predefined tolerance.

Figure 6 shows a Gaussian distribution as the desired distribution for the transmission with a width of 25° (FWHM) as well as

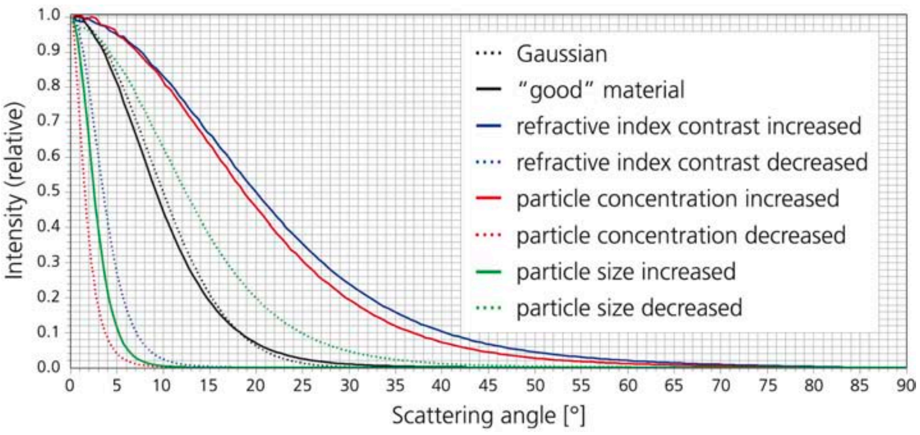


Figure 4: Scattering intensity normalized with the power of the incident beam at perpendicular incidence for different diffuser compositions

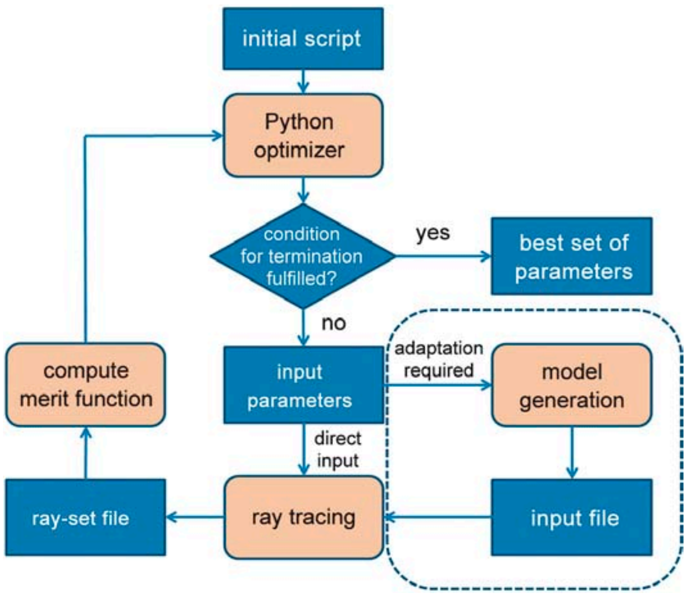


Figure 5: Flow-chart of optimization process, blocks with beige background colour can be interchanged

the best fit that has been achieved with the simulation model. The transmission amounts to 90% and no unscattered light gets transmitted. This result can be reached by using relatively large particles (diameter >20µm), while the refractive index differs only slightly from the matrix. We note that it is not possible to find a recipe for any desired distributions (e.g., a distribution with a double peak). A volume scattering material as diffuser is in this case unsuitable.

4 Conclusion

We presented the strategy to use commercial ray tracers as basically interchangeable compute engines and otherwise use special programs or scripts for their management and higher-level tasks. This is a promising way to solve non-standard problems as well as sophisticated design and analysis challenges. As a matter of fact there is market niche for such "secondary software", which can considerably increase the performance and flexibility of commercial ray tracers.

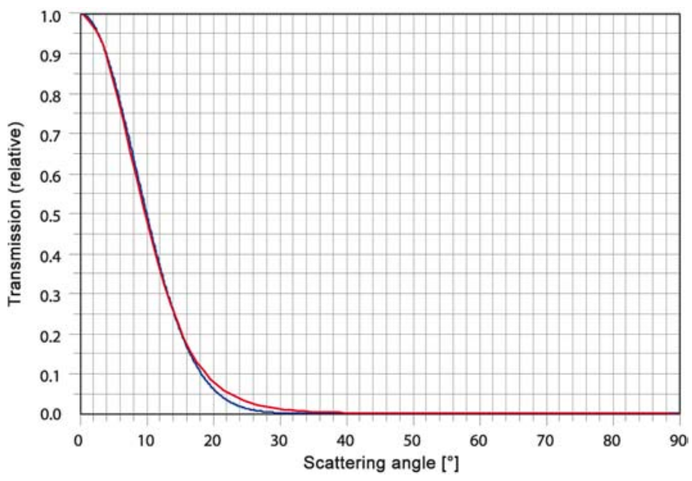


Figure 6: Desired light distribution of a diffusor (thickness 2 mm, perpendicular incidence) compared to the light distribution of an optimized realistic diffuser model

Literature:

[1] *www.python.org*
[2] *en.wikipedia.org/wiki/Legendre_polynomials*
[3] *en.wikipedia.org/wiki/Importance_sampling*
[4] B. Michel, P. Holcomb, *Simulation of light scattering in human skin*, Photonik international 1/2008, p. 84-87

Author contact:

Dr. Bernhard Michel
Managing Director
Hembach Photonik GmbH
Finkenstr. 1-3
91126 Rednitzhembach, Germany
Tel. +49/9122/8899490
Fax +49/9122/8899499
eMail: bm@hembach-photonik.de
Internet: www.hembach-photonik.de

Monika Kroneberger — eMail: mk@hembach-photonik.de
Robert Hermann — rh@hembach-photonik.de