WEB-BASED LIGHT SIMULATION AND INTERACTIVE CONFIGURATION FOR LIGHTING SYSTEMS

Luca Fluri*, Manuel Riedi* and Hilko Cords
Institute of Interactive Technologies,
University of Applied Sciences and Arts Northwestern Switzerland,
Hochschule für Technik, Bahnhofstrasse 6, 5210 Windisch, Switzerland

* Equal Contribution

ABSTRACT

The current generation of lighting systems produced by lighting manufacturers can be highly configurable. Consequently, it is challenging for manufacturers to showcase the potential of configurations and the impact of light on specific environments. This issue is further compounded by the increasing prevalence of online shopping in the lamp industry, which has historically been a relatively slow-moving sector.

This paper presents an interactive, web-based configurator for complex lighting systems featuring real-time light simulation running on the client. Our work addresses the aforementioned issues directly by facilitating fast, physically accurate path tracing for interactive exploration and a novel and intuitive user interface for lamp system configurations. Consequently, our approach is highly performant due to analytical calculations for measured, physical lamp data stored in texture space for rendering and exploring static light configurations. Thus, our system can be executed on low-specification hardware and mobile devices while maintaining interactivity.

KEYWORDS

Light Simulation, Web-based Configurator, React, WebGL, Path Tracing

Figure 1 *An empty room, illuminated by a diverse array of physically accurate lights. The configuration of these lights is achieved through the utilization of our web-based configurator, while the rendering process employs our texture baking and reverse path tracing approach in WebGL.*

1. INTRODUCTION

The lighting design plays a pivotal role in shaping spaces, influencing both the aesthetic appeal and functionality of a given environment. However, traditional lighting design software (*Relux Informatik AG, 2024; DIAL GmbH, 2024*) is designed for professionals and is often complex and non-interactive. Furthermore, although this type of software is highly flexible, it is typically inappropriate for end users to access and unsuitable to be integrated within an online store. These factors present a significant challenge for lighting manufacturers attempting to market and sell intricately adjustable lamp system products online.

This paper presents a novel solution to address the aforementioned issues: a fast, web-based light simulation and configuration system for configurable lighting systems. In this way, we address two distinct problems:

a) The need for a physical-accurate simulation of real, physical lamps on the web with limited resources (*Figure 1*), and
b) the challenge of an intuitive configuration of complex lighting systems.

Regarding a), the technical core of our solution employs global illumination methods, with particular emphasis on GPU-based path tracing. We utilize real-world photometric data to calculate and visualize realistic light propagation for arbitrary configurations. To optimize performance, we chose a texture-based approach that only bakes direct and global illumination data when the scene configuration changes. This ensures fast and responsive 3D exploration while maintaining realistic lighting effects. Our method produces results that are very close to reality.

With regard to b), we address the challenge of providing an intuitive user experience (UX) when configuring complex lighting systems. Consequently, our concept bridges the gap between web-based simulation and configuration since our interactive light simulation is integrated directly into the configurator.

The benefits of our work extend beyond the enhancement of the UX. While existing online configurators are typically unable to provide users with a realistic impression of the actual lighting effects, our system offers the end user a clear understanding of real-world lighting effects according to a customized configuration. Our comprehensive solution is currently being implemented by the lighting manufacturer RIBAG (2024) in their current online shop.

2. RELATED WORK

General path tracing is a well-studied area of research. In particular, the interaction of light with surfaces has been well studied in the context of physically based rendering (PBR) and Monte Carlo algorithms (*Lafortune, 1996*). Being computationally expensive, optimizing data structures and hierarchies has been a long-standing topic of study (*Goldsmith & Salmon, 1987*). Thereby, light maps for static lights are widely used to maximize runtime efficiency without sacrificing visual quality (*Luksch, 2013; Rasmusson, 2010; Özturk, 2017*). Nowadays, path tracing research often focuses on real-time capabilities (*Lin et al., 2022; Wyman & Panteleev, 2022*). While these approaches typically facilitate high-end GPUs, the need for path tracing on the web on low-end devices has been equally increased (*Yu et al., 2023*).

To enhance the realism of web-based light simulations, a novel scripting environment for creating and interacting with ray-marched 3D graphics has been proposed, demonstrating the feasibility of implementing complex ray-tracing algorithms in web applications (*Albrigsten, 2021*). An interactive approach to indirect illumination, based on voxel cone tracing, has been developed to address the challenges of simulating global illumination and multiple light interactions in real-time on web platforms (*Crassin et al., 2011*). Furthermore, real-time platform-agnostic volumetric path tracing has been showcased in WebGL 2.0, illustrating its potential for high-quality light simulations (*Lesar, Bohak & Marolt, 2018*). Extending the capabilities of ray tracing, recent implementations with DXR, Vulkan, and OptiX show how these technologies can be adapted for web-based applications to achieve near-photorealistic rendering, significantly advancing the state of the art in web-based light simulations (*Marrs et al., 2021*).

In addition to path tracing, research is directed towards the integration of a configurator for lighting systems that incorporates realistic lighting effects. Integrating configurator systems with real-time rendering capabilities is paramount to providing users with immediate visual feedback. A high-quality parametric visual product configuration system, emphasizing the significance of advanced visualization techniques, has been described (*Segura et al., 2005*). Furthermore, the potential of implementing a photorealistic rendering system using GLSL has been highlighted, with emphasis on the physically-based environment and area lighting using progressive rendering in WebGL (*Otto, Limberger & Döllner, 2020; Vitsas et al., 2020*).

Web-based light configurators do already exist but are either limited to top-down 2D views without actual light simulation (*Lichtwerk GmbH, 2024; Light Performer, 2024*) or limit the simulation to only one lamp per 3D scene (*Prolicht, 2024*). Furthermore, these platforms often lack comprehensive customization options and user-friendly interfaces that cater to both novice and professional users (*Lichtwerk GmbH, 2024; Light*

Performer, 2024; Prolicht, 2024). While some configurators provide advanced visualization and customization features, they may require extensive technical knowledge or fail to support the configuration of complex lighting systems in real-time (*Prolicht, 2024*). There is a clear need for a light configurator that is more intuitive, accessible, and fully featured. It should combine user-friendliness with robust customization and simulation capabilities, showing significant room for improvement.

3. OUR APPROACH

Our solution is two-fold, addressing both the technical challenges of

a) realistic light simulation in web environments and the
b) practical needs of end-users in configuring lighting environments.

Figure 2 illustrates our results and seamless integration between the configurator and the light simulator, showcasing the same lighting setup. Thereby, changes made in the configurator are instantly reflected in the 3D view. Users can switch between views to compare their configurations with the simulated outcomes, ensuring that their final setup meets their expectations, requirements, and tastes.

Regarding a), we describe below a texture-based light simulation approach capable of handling direct and global illumination. Our method employs analytical rendering techniques to simulate the light in a given scene on the GPU, ensuring high realism and accuracy.

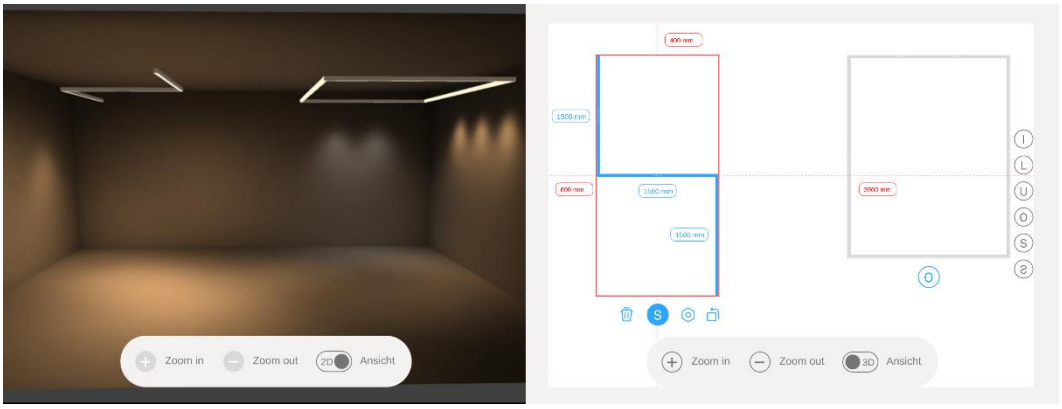

Figure 2 *A light system rendered with our pipeline in the 3D viewer (left). The same light system in the configurator view (right).*

Regarding b), our light simulation method was incorporated into an end-user-centric light system configurator. This configurator provides an intuitive and responsive interface, allowing users to manipulate and adjust lighting parameters in real-time. By integrating our light simulation method into the configurator, any alterations made by the user are promptly reflected in the visual feedback, thus creating a seamless and interactive experience.

In Section 3.1, we provide a comprehensive examination of our light simulation approach, while in Section 3.2, we address all aspects related to the configurator itself.

3.1 Light Simulation

The simulation and visualization of physical light propagation in web environments present considerable computational challenges. Our approach addresses these challenges by giving users a realistic impression of a lamp's illumination within a room while acknowledging the limitations of performing full polygonal path tracing within a web-based scenario.

Based on analytical GPU-based path tracing, our technique enables the realistic simulation of real-world lighting conditions for the specific lighting setup present in a room while ensuring appropriate performance and interactivity.

Typically, path tracers rely on specialized data structures and often leverage the GPU's processing power for efficiency. The difficulty lies in mapping complex scenes onto these GPU-friendly data structures, which can be memory-intensive due to the large amount of data processed. To address these constraints, we employ the following techniques:

a) ***Iteratively Rendered Textures***: Global illumination is progressively refined in the final image, allowing for different quality levels based on user preferences while maintaining interactivity.
b) ***Programmatic Data Representation***: Position and orientation data of lights and lighting systems are represented programmatically in shader code, allowing for shader compilation optimization. Meanwhile, the radiation characteristics per lamp are passed via texture.
c) ***Analytical Geometry***: Since the scene geometry for rooms typically consists of a limited number of polygons or even cubes, analytical methods are used for efficient lighting calculations.

The scene (i.e., room) and lamp configurations are static in the typical configuration use case. This allows for significant performance improvements by pre-calculating each wall's diffuse, view-independent light characteristics and storing them in individual wall textures. All light calculations are based on industry-standard photometry datasets, i.e., IES files (*Illuminating Engineering Society, 2019*) which incorporate the photometric characteristics of real lamps, providing an accurate physical description of any specific lamps (*Figure 3*).

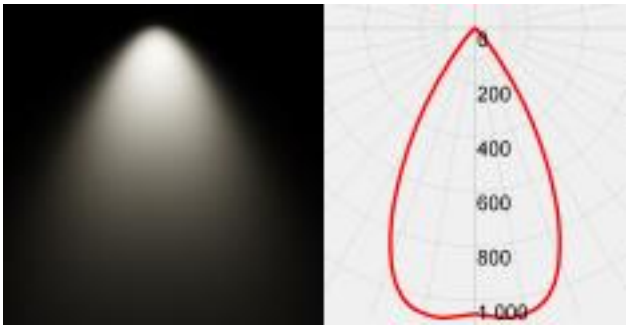

Figure 3 *Photometric characteristics of a real lamp, stored in an IES file.*

The initial step is to create direct light wall textures by sampling the IES data in relation to the position and orientation of the light source (*Figure 4.a*). For global illumination effects, these direct light wall textures are used to analytically and iteratively calculate a separate indirect light texture per wall using reverse path tracing. Using all direct lighting textures, the indirect lighting pass casts rays from the current wall in a random hemispherical direction, normal to the surface. It then calculates the intersection points with all other walls and averages all summed-up texture values at these points, giving the current pixel value of the indirect lighting texture (*Figure 4.b*). This process is repeated for all texture pixels and for all other walls. In the final step, the noisy indirect light textures are blurred, which enhances the image quality without excessive computational costs (*Figure 4.c*). The blurred indirect and direct light textures are blended while applying light maps to the room's walls (*Figure 4.d*). This approach leaves only simple geometries (in our case, the room) to render in real-time, and the light maps only need recalculation if the room or light setup changes. *Figure 5* shows the complete illumination pipeline.

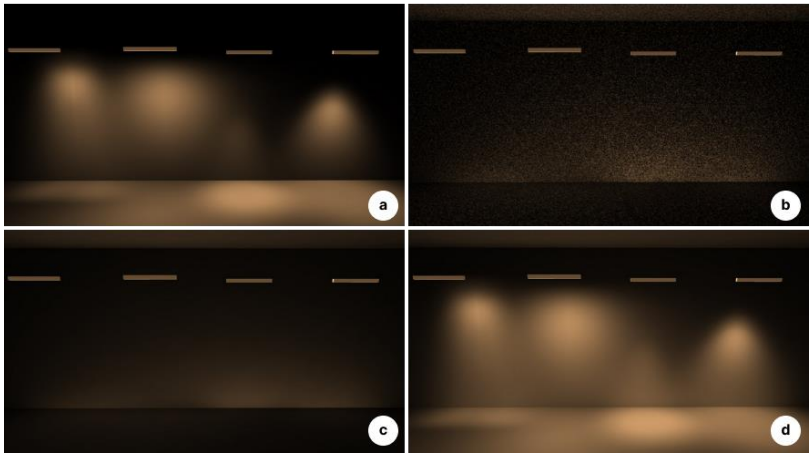

Figure 4 *All stages of our rendering pipeline combined. Direct lighting from different lights using data from IES files (a). Indirect lighting using only direct light textures (b). Blurred indirect texture (c). Final blended textures (d = a ⊕ c).*

Using analytical methods in the indirect lighting calculation provide a substantial performance advantage compared to traditional ray-intersections with polygonal geometry. This approach significantly improves rendering performance compared to intersections with complex polygonal meshes. Additionally, it allows for efficient discretization of area light sources using point lights, mirroring the approach used in professional lighting calculation software.

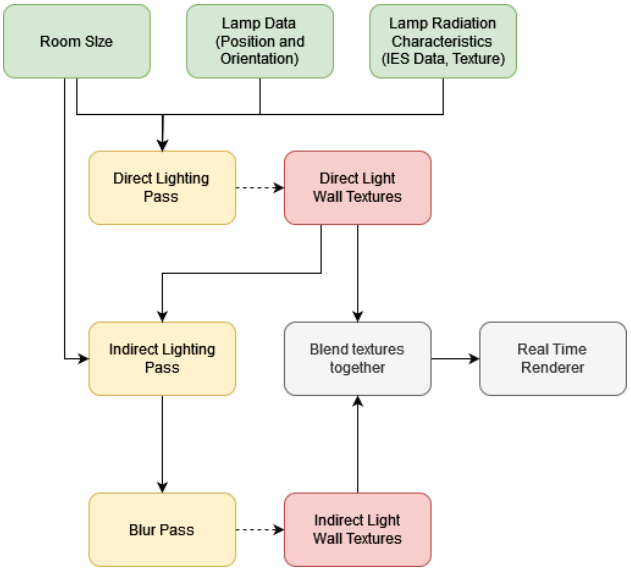

Figure 5 *Our rendering pipeline: Shader steps (yellow), static resources (green) and wall light maps (red).*

Our system dynamically generates and compiles shader programs at runtime. These programs directly encode the geometry of the room as well as lamp positions and orientations in an analytical way, significantly reducing slow memory access on the GPU and maximizing the utilization of its Arithmetic Logical Unit (ALU) for efficient computations. However, the discrete characteristics of a lamp in an IES file can be too large for efficient shader compilation or might exceed the shader's local memory. Instead, these radiation properties are passed to the shader in a texture. This ensures fast and efficient access and enables fast GPU standard lerping for interpolation.

3.2 Configurator

In addition to our light simulation, we have developed a general lighting system configurator capable of managing complex lamp configurations while facilitating data exchange with the lighting simulator. This integration allows users to view the 3D visualization of their present configuration at any time.
In order to exemplify our methodology, we employed the *Milum* system, developed by Ribag AG (*RIBAG, 2024*), as a case study. This system features long support rails that can be connected in snake-like configurations, with various light modules inserted into the rails. Nevertheless, our methodology is designed to be broadly applicable, allowing for integration with a variety of other lamp systems.

Although *React* and *TypeScript* were the principal technologies employed in the configurator, a variety of usability features were incorporated with the objective of enabling even novice users to become proficient and productive with minimal effort. In view of the considerable configuration possibilities typically associated with such lamp systems, many usability concepts were developed and integrated to conceal complexity. Map services such as Google Maps have served as a source of inspiration for our UX design, exemplifying our design approach. *Zooming* in and out reveals more or less detailed information based on the current view. Furthermore, approaches based on the principle of *always-validity* have been implemented, which prevent incorrect configurations by disallowing them from the beginning. In addition, presets have been introduced, enabling users to commence work directly from predefined lamp system configurations.

The configuration process begins with the specification of the room dimensions on the initial page. Subsequently, the main screen is presented, wherein the white area displays the room in a 2D plan view (*Figure 6*). The right navigation bar allows the user to place a specified preset lamp system in the room, while the bottom navigation options provide access to the navigation controls. The presets streamline the configuration process, simplifying the setup and making it especially user-friendly for beginners. Furthermore, a preset placed in the room can be translated and rotated as desired and it can be adjusted for ceiling distance, with the length of each support rail also modifiable.

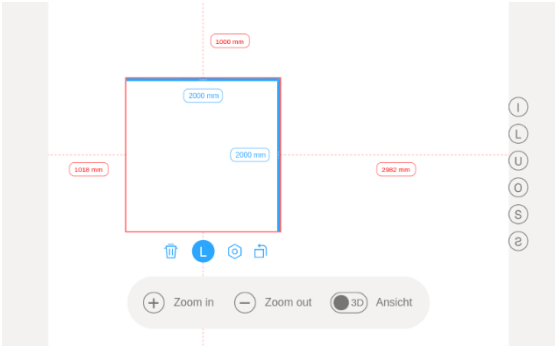

Figure 6 *A preset is displayed in the room (blue and red elements). Bottom controls allow deletion, selection, additional options, and rotation. Red input fields move the preset, while blue input fields adjust the support rail length.*

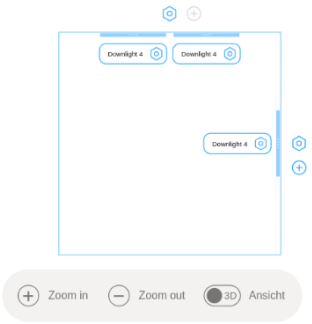

Figure 7 *Zooming in on the scene from Figure 6 reveals a more detailed light module configuration view.*

A lamp system provides extensive configuration options, ranging from support rails to light temperature. To avoid overloading the user interface, settings must be thoughtfully organized. Consequently, the controls for these UI elements are separated between the overall system and the light modules. This separation was facilitated through the stepless zooming approach mentioned earlier. Further zooming transitions the interface to the light module configuration level, thus facilitating the placement of specific light modules (*Figure 7*).

Given the highly event-driven nature of *React* and the necessity for seamless, natural animation during central zoom processes, the *React Spring* library was integrated. *React Spring* excels in crafting interactive, responsive web applications with smooth animations. Given the frequent alterations in the representation of drawn elements (e.g., enlargement or reduction in size) during zooming, significant animation effort is required. Consequently, we opted to employ the lightweight SVG graphics format to render the entire drawing area, encompassing both UI elements and all components of the lamp system. The zooming process reduces

the number of rendered elements in the same space, namely the visible window section. This space allows for the introduction of new UI elements, which remain constant in size and distance to the lamp elements throughout the zoom process.

New modules can be added by clicking the plus button located above a support rail until the space is fully occupied. This approach is in accordance with the previously mentioned *always-valid* methodology, which is designed to prevent overflow (*Figure 8*). The objective of this methodology is to prevent potential misconfigurations proactively, thereby streamlining the user's thought process.

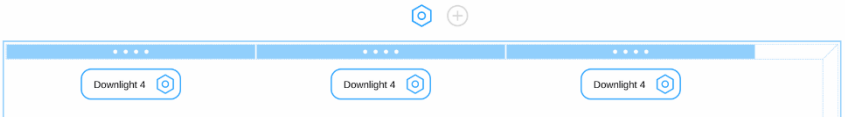

Figure 8 *Demonstration of the always-valid approach by adding light modules until there is no more space left, which deactivates the relevant button(s).*

The *always-valid* approach is also applied to light distribution options, enabling light modules to be arranged within support rails with various alignments (inside-centered, outside-centered, start or end). Furthermore, it facilitates switching between different light module types using the buttons located below each respective module. Only options that avoid clashes with existing light modules in the support rail can be selected (*Figures 9 and 10*). Finally, the light modules can be customized through the utilization of settings button, e.g., adjusting color temperature or exposure angle (*Figure 11*).

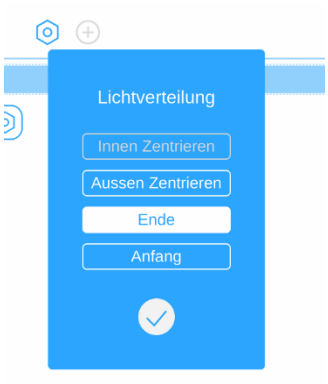

Figure 9 *The always-valid approach is applied to light distribution, disabling invalid options.*

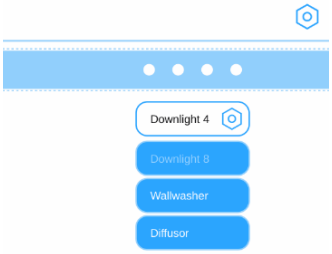

Figure 10 *The always-valid approach is used for light module changes, deactivating any invalid options.*

Figure 11 *Configuring light module settings.*

During the configuration process, regardless of the selected zoom level, the current setup can be viewed in 3D at any time by utilizing the 2D/3D toggle switch, located at the bottom of the navigation bar (e.g., Figure 7). This activates a WebGL component in the background which is responsible for the management of data exchange between the configurator and the light simulation. It retrieves the current configurator values being essential for updating the 3D rendering, which include e.g., system positions, rotations, or color temperature.

4. RESULTS & DISCUSSION

We have integrated our proposed light simulation and configurator into a web environment utilizing WebGL, React, JavaScript, and TypeScript. Our evaluation encompasses results from all major browsers, including Chrome, Firefox, Opera, and Microsoft Edge. The following sections will analyze our system's visual quality, performance, and usability.

On a PC with an Intel Iris Xe GPU and 13th Gen Intel Core i9-13900H CPU using the Chrome browser (Version 126.0.6478.183), the average rendering time for all direct textures is *26 ms*, while the indirect textures render in *0.1 ms* on average. The time required for the rendering of direct lighting textures depends on the complexity of all the lamps present within the scene. The algorithm proceeds through each lamp and gathers its radiance properties. In contrast, indirect lighting does not depend on the number or complexity of the lights in a scene. It only relies on the six wall textures created during the initial step of direct lighting. Consequently, the rendering time of indirect lighting scales with constant time, whereas the creation of direct light textures does not. Although *26 ms* for the creation of six direct lighting textures may appear to be a considerable amount of time, it should be noted that this calculation is not performed on every frame. Instead, it occurs only once after the scene setup changes, ensuring high framerates for the real-time renderer, even on low-spec machines.

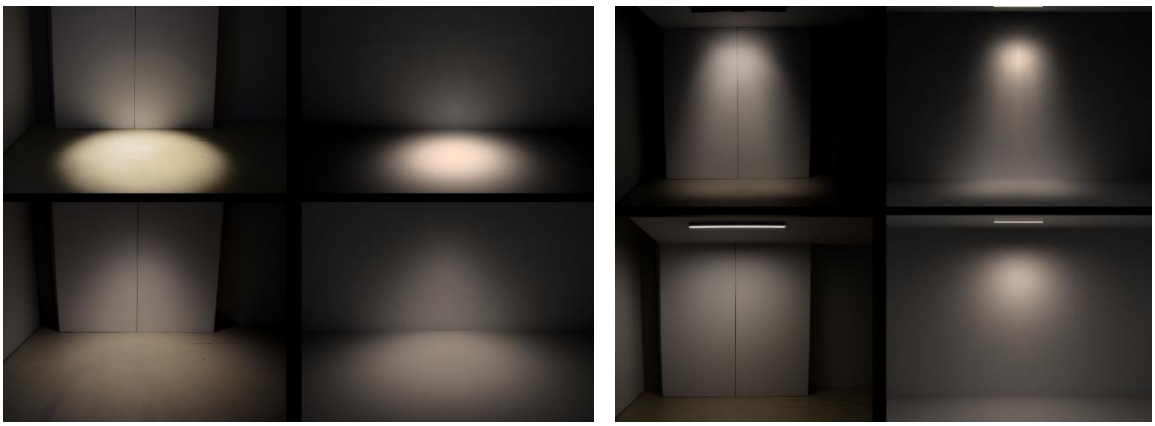

Figure 12 *Comparison between real-life pictures (left) and our results (right). Various lights are shown.*

The visual quality we achieve is comparable to that of professional simulation software and closely resembles actual photographic lighting conditions (*Figure 12*). In the comparison, the left side shows the original photographs, while the right side displays our results. The close match between the two highlights our configurator's precision and effectiveness.

A challenge in developing the configurator is integrating complex configuration options, typically found in CAD programs, into a user-friendly interface that is easily understandable for beginners and occasional users. We've observed that achieving this balance often requires a compromise between the breadth of settings and usability. However, a zooming approach can effectively distribute complexity across different levels, thereby reducing the user's cognitive load.

In principle, React was an optimal selection for our implementation due to its suitability for creating complex user interfaces, aligning with web-based requirements and dynamic usability principles. We generated most of the drawn elements through vector calculations, which proved advantageous for precisely describing geometric objects such as support rails or light modules in code. Furthermore, the integration of React Spring with seamless SVG blocks for drawing lamp system elements enabled the creation of exceptionally smooth transition animations and zooming navigation, even within a web-based environment.

5. CONCLUSION & FUTURE WORK

We presented a web-based configuration solution for lighting systems that a light manufacturing company is currently implementing (*RIBAG, 2024*). Our developed solution enables the intuitive configuration of a variety of lighting systems and provides an interactive preview of the anticipated lighting conditions. The interaction and UX concepts are based on contemporary methods and were developed through a user-centered design process. Our approach employs TypeScript, React, and WebGL, thereby ensuring an efficient implementation. Our innovative light simulation demonstrates how fast calculations and visualizations can be performed online through a render-to-texture approach in combination with reverse path tracing, which is particularly relevant for the requirements of a web-based configurator.

Future work for the light simulation might include the introduction of windows into the scene and the rendering of ambient light, which would serve to brighten the currently dark and empty rooms. A feature we have already examined is the placement of polygonal objects within the room (*Roth & Cords, 2024*). This allows users to observe the interaction between light and various objects, thereby enhancing the configurator's capability to simulate real-world scenarios and providing a more comprehensive tool for lighting design.

Regarding the configurator, our objective was to create a flexible tool capable of representing a range of lighting systems. Furthermore, additional features have been conceptualized, including a light cone indicator in the 2D view, which provides an approximate idea of the light spread before viewing the 3D rendering. Additionally, a live calculation of the current configuration's price is planned.

In conclusion, our solution meets the current needs for web-based lighting configurators and establishes a foundation for future enhancements, ensuring adaptability and comprehensive utility for both users and developers.

ACKNOWLEDGEMENT

This research was carried out in collaboration with the company Ribag AG and funded with support from the Forschungsfonds Aargau. We would like to thank Dominik Hausherr and Andreas Richner from Ribag AG for the valuable collaboration.

REFERENCES

Albrigtsen, Ø., 2021. *A web-based scripting environment for creating and interacting with ray marched 3D graphics*. Master's thesis, NTNU, Norway.

Crassin, C., Neyret, F., Sainz, M., Green, S., & Eisemann, E., 2011. *Interactive indirect illumination using voxel cone tracing*. Computer Graphics Forum, 30(7), pp. 1921–1930.

DIAL GmbH., 2024. *Dialux: The ultimate software for lighting design*. Available at: https://www.dialux.com/ [Accessed 11 July 2024].

Goldsmith, J. & Salmon, J., 1987. *Automatic creation of object hierarchies for ray tracing*. IEEE Computer Graphics and Applications, 7(5), pp. 14–20.

Illuminating Engineering Society., 2019. *IES standard file format for the electronic transfer of photometric data and related information*. ISBN: 978-0-87995-015-6.

Lafortune, E., 1996. *Mathematical models and Monte Carlo algorithms for physically based rendering.* Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven, 20(74-79), p.4.

Lesar, Ž., Bohak, C., & Marolt, M., 2018. *Real-time interactive platform-agnostic volumetric path tracing in webGL 2.0*. In Proceedings of the 23rd International ACM Conference on 3D Web Technology, Web3D '18, New York, NY, USA, June 2018, pp. 1–7. Association for Computing Machinery.

Lichtwerk GmbH, 2024. *Serio: Konfigurator*. Available at: https://serio.lichtwerk.de/de/ [Accessed 11 July 2024].

Light Performer, 2024. *Light performer configurator: Duct*. Available at: https://www.lightperformer.com/configurator [Accessed 11 July 2024].

Lin, D., Kettunen, M., Bitterli, B., Pantaleoni, J., Yuksel, C., & Wyman, C., 2022. *Generalized resampled importance sampling: foundations of ReSTIR*. ACM Transactions on Graphics, 41(4):75:1–75:23.

Luksch, C., Tobler, R. F., Habel, R., Schwärzler, M., & Wimmer, M., 2013. *Master light-map computation with linear-time complexity*. In Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D '13, New York, NY, USA, March 2013, pp. 87–94. Association for Computing Machinery.

Marrs, A., Shirley, P., & Wald, I., 2021. *Ray Tracing Gems II: Next Generation Real-Time Rendering with DXR, Vulkan, and OptiX*. Apress, Berkeley, CA.

Öztürk, B., & Akyüz, A. O., 2017. *Semi-dynamic light maps*. In ACM SIGGRAPH 2017 Posters, pp. 1–2.

Otto, P., Limberger, D., & Döllner, J., 2020. *Physically-based environment and area lighting using progressive rendering in WebGL*. In Proceedings of the 25th International Conference on 3D Web Technology, Web3D '20, New York, NY, USA, November 2020, pp. 1–9. Association for Computing Machinery.

PROLICHT, 2024. *Produktkonfigurator*. Available at: https://www.prolicht.at/ [Accessed 11 July 2024].

Roth, A., & Cords, H., 2024. *Texture-based global illumination for physics-based light propagation in interactive web applications*. In 32nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2024), pp. 97-106. Pilsen, Czech Republic.

Relux Informatik AG, 2024. *Relux: Lighting and sensor planning software*. Available at: https://www.relux.com/ [Accessed 11 July 2024].

RIBAG, 2024. *Ribag: Innovative lighting solutions*. Available at: https://www.ribag.ch/ [Accessed 11 July 2024].

Rasmussen, J., Ström, J., Wennersten, P., Doggett, M., Akenine-Möller, T., & Ericsson Research, 2010. *Texture compression of light maps using smooth profile functions*. IEEE Computer Graphics and Applications.

Segura, Á., Arizkuren, I., Aranburu, I., & Telleria, I., 2005. *Quality parametric visual product configuration systems over the web*. In Proceedings of the tenth international conference on 3D Web technology, Web3D '05, March 2005, pp. 159–167.

Vitsas, N., Papaioannou, G., Gkaravelis, A., & Vasilakis, A. A., 2020. *Illumination-guided furniture layout optimization*. Computer Graphics Forum, 39(2):291–301.

Wyman, C., & Panteleev, A., 2022. *Rearchitecting spatiotemporal resampling for production*. In Proceedings of the Conference on High-Performance Graphics, Goslar, DEU, pp. 23–41. Eurographics Association.

Geng, Y., Liu, C., Fang, T., Jia, J., Lin, E., Fu, Y., Wang, L., Wei, L., & Huang, Q., 2023. *A survey of real-time rendering on Web3D application*. Virtual Reality & Intelligent Hardware, 5(5):379–394.