Joint Virtual Reality Conference of EuroVR - EGVE - VEC (2010)
T. Kuhlen, S. Coquillart, and V. Interrante (Editors)

Improving Photon Mapping Towards an Interactive Stage Lighting Framework

T. Vierjahn[1], S. Meik[2], S. Mostafawy[1] and C.-A. Bohn[3]

[1]Department of Media, FH Duesselduorf University of Applied Sciences, Germany
[2]Lighting Department, Theatre Oberhausen, Germany
[3]Department of Mediainformatics, FH Wedel University of Applied Sciences, Germany

Abstract

In this work we present an improved rendering algorithm for the planning of stage lighting using photon mapping, since existing software implements algorithms that do not suit the lighting workflow in a theatre adequately. During theatre lighting, positional changes of light sources are occuring rarely compared to intensity and color changes: Once a light is positioned and aimed correctly, many intensity and color values are tested. Thus, intensity and color changes must be recomputed very fast whereas positional changes may take longer to be rendered. The algorithm we present provides such fast recalculations and furthermore balances idle times of user and renderer.
In our algorithm image synthesis is split into a rendering and a real-time updating process. Additionally the well-known concept of splitting the photon map into a global and a caustic map is consequently refined to light-specific photon maps and intensity buffers.
The presented prototypic system utilizes general purpose API to get further insight into and proof of the concepts. Results indicate that the proposed algorithm can efficiently be used to plan and simulate stage lighting in a theatre.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

Achieving adequate and appealing stage lighting has been a problem since the early days of ancient greek theatres: Plays had to be performed at daytimes, otherwise the stage had to be illuminated with torches or oil lamps. When theatres moved from open-air arenas to indoor stages, oil lamps and torches were replaced by candles and later by gas lamps. But the lighting department still had to do hard work: To satisfy the audience's desire "to see a shining and shimmering stage" [Lan10], many people were needed to operate lighting fixtures, control valves and special effects [Kob88]. With the advent of electrical light and mechanical remote control for electric dimmers, staff requirements were reduced; even more with modern computerized control desks and electronic dimmers.

Although the workload was reduced, the problem of achieving adequate and appealing stage lighting has not changed, since lighting systems became more complex. Thus, not only the acting is rehearsed on stage during the weeks before a premiere, but also the lighting is. During these rehearsals the stage cannot be used otherwise. Additionally there are theatres with a lighting system too complex to evaluate all settings during lighting rehearsals. Thus for cost-effectiveness these have to be carefully planned beforehand.

Computer graphics allows for realistic simulation and visualization of different lighting setups. For this purpose software is readily available: on the one hand professional modelling and rendering packages that are used in the movies, on the other hand dedicated planning software for lighting. These are very elaborate tools, but do not satisfy all of the requirements of planning lighting in the theatres efficiently: The former are tools specialized for completely different fields of work, the latter are rather intended for planning

DOI: 10.2312/EGVE/JVRC10/077-084

huge events and tours. Some being too complex, offering too many tools that are unneccessary for the use in the theatres. Some are taking too long a time for rendering. None of them matches the workflow of theatre lighting.

Consequently, in this work an algorithm was developed that matches the workflow of theatre lighting and reduces rendertimes. If used in a planning tool, it simplifies the problem of achieving adequate and appealing stage lighting.

The remainder of this paper is organized as follows: Section 2 presents related work. In section 3 the proposed algorithm is derived and additional components of the rendering system are presented. Section 4 presents the results. Section 5 concludes the paper and gives an overview over future work.

2. Related work

The rendering algorithm proposed in this work is based on Whitted's recursive ray tracing [Whi80]. The influence of direct illumination is calculated using the illumination model proposed by Phong [Pho75] in the prototypic system, but any other could be easily integrated in later versions. Photon mapping as proposed by Jensen is used to calculate the contribution of indirect lighting [Jen01]. The bounding interval hierarchy proposed by Waechter and Keller [WK06] is used as an acceleration structure for ray tracing . Primary intersections are precalculated. The used technique is an extension to the one proposed by Weghorst *et al.* [WHG84].

The introduction of light-centric intensity buffers and photon maps was inspired by the note of Lambert "the larger the number of candels, the greater the illumination" [Lam60]. A similar concept of individual photon maps represents the well-known separation of global and caustic maps proposed by Jensen. Christensen and Batali propesed an "irradiance atlas" with separate photonmaps for groups of objects [CB04] – the geometry-centric motivation for the light-centric variant presented in this paper.

Much effort has been spent to speed-up ray tracing and global illumination. The most notable work is *OpenRT* [Wal04]. Tracing packets of rays has been a common speed-up for camera and shadow rays. The application of packets to reflection and transmission rays was discussed and evaluated by Boulos *et al.* [BEL*07]. Havel and Herout utilized the SSE4 instruction set to accelerate ray triangle intersections for ray packets as well as for single rays [HH10].

Progressive photon mapping was introduced by Hachisuka *et al.* to find arbitrary accurate radiance estimates [HJ09]. This technique can be used to increase accuracy during idle times. Purcell *et al.* implemented photonmapping on a GPU [PDC*03] using compact grids. A more general GPU approach was presented by Wyng *et al.* [WWZ*09]. McGuire and Luebke presented a hardware accelerated, image based extension, reducing the costs of first and last bounces [ML09].

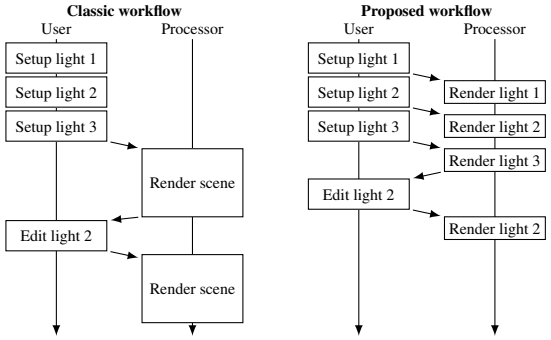

Figure 1: *Classic (left) versus proposed workflow (right): In existing rendering packages the render process is started once the lighting is set up completely. In the proposed workflow a render process is started after setting up only one single light. With the proposed method the renderer (re)calculates only the light that has changed. Thus the user can inspect and edit the result earlier.*

Existing rendering software is per-se suitable for lighting simulations. Among others are commercial products like *mental ray*® [Men] or *RenderMan*® [Pix], but also free alternatives exist like *POV-Ray*™ [Per] and *LuxRender* [Lux]. The latter is based on Pharr's and Humphrey's *pbrt* [PH04] and implements lightgroups, a similar but coarser concept as propsed herein. Renderers that allow for interactive changes are *Shaderlight*™ [Art] and *fryrender SWAP* [Ran]; interactive lighting previews for *RenderMan*® are offered by *Lpics* [PVL*05].

Specialized lighting simulation software like *grandMA 3D* [MA] and *wysiwyg* [cas10] are also available. Both offer real-time previews of the lighting. The latter offers global illumination for offline renderings utilizing radiosity [GTGB84]. A macro package for *POV-Ray*™ exists to implement theatre lighting [CSh09].

These very elaborate tools are not completely sufficient for planning theatre lighting: Some are too complex to be practical for a user from outside computer graphics industry. Some take too long a time to render, since they are specialized tools for different fields of work. None of the tools matches the workflow of theatre lighting.

3. The new rendering system

For lighting simulation with existing software packages, a user would first construct the scene, then set up lighting and finally start the rendering process. Thus, the complete scene gets rendered (cf. fig. 1 left). After some time, the result can be inspected and adjustments can be made. Eventually, the scene would have to be rerendered completely, even if only intensity or color of one single light were changed.

In this paper a different workflow is proposed: While the

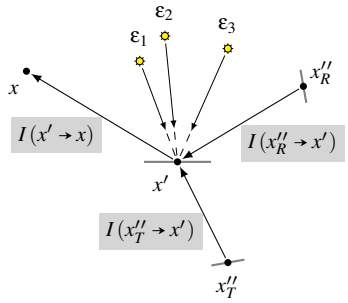

Figure 2: *Illustration of the notation used throughout this paper: The point x' is lit by 3 point lights ε_1 - ε_3. The point x''_R is visible in reflection at x', the point x''_T is visible in transmission through x'. Total intensity is leaving x' towards x.*

user is setting up the lighting, the processor is nearly running idle. Thus, the processor could render the effects of the first light while the user is setting up the second one (cf. fig. 1 right). Eventually, after having set up the last light, the user would have to wait only for the last light to finish rendering. By better balancing the workload in such a way, the user could begin to adjust lighting earlier than in the classic workflow. This new workflow requires the renderer to be capable of detecting changes of lighting and of only rendering the effects of these changes. Thus, the scene would not have to be rendered completely unless all lights were changed.

3.1. Reworking lighting model and algorithm

Let $I\left(x' \to x\right)$ be the intensity reflected off point x' towards point x and let x''_R be a point visible in reflection at x' and x''_T be a point visible in transmission through x' respectively (cf. fig. 2). Let furthermore $g\left(x' \to x\right)$ be a term that evaluates mutual visibility of x and x', with $g\left(x' \to x\right) = 1$ if x and x' are mutually visible and $g\left(x' \to x\right) = 0$ otherwise. With an ambient intensity of ε_A and m point lights, each with intensity ε_i positioned at P_i in a distance r_i to x', Whitted's shading model [Whi80] can be rewritten:

$$I\left(x' \to x\right) = g\left(x' \to x\right)\Bigg[k_A\varepsilon_A + \sum_{i=1}^{m} \frac{g\left(P_i \to x'\right)\varepsilon_i}{r_i^2} f_P\left(P_i \to x' \to x\right) + k_R I\left(x''_R \to x'\right) + k_T I\left(x''_T \to x'\right)\Bigg] \quad (1)$$

With k_T being Whitted's transmission coefficient, k_R being a similar coefficient for specular reflection and k_A a coefficient for determining the reflection of ambient illumination. The BRDF $f_P\left(P_i \to x' \to x\right)$ is estimated by Phong's illumination model [Pho75]. As Jensen suggested, only the ambient term $k_A\varepsilon_A$ will be calculated by his radiance estimate [Jen01], to integrate global illumination.

During theatre lighting, positional changes of light sources are occuring rarely compared to intensity and color changes: Once a light is positioned and aimed correctly, many intensity and color values are tested. Thus, intensity and color changes must be computable very fast whereas positional changes may take longer to be rendered.

To create a new algorithm with the proposed workflow (cf. fig. 1 right), allowing for the above constraints, let $I\left(x' \to x\right)$ (from equation 1) be separable into $I_{\varepsilon_A}\left(x' \to x\right)$ and $I_\varepsilon\left(x' \to x\right)$ in such a way that $I_{\varepsilon_A}\left(x' \to x\right)$ is summing up only the ambient terms of equation 1 to be calculated with Jensen's radiance estimate, and $I_\varepsilon\left(x' \to x\right)$ summing up the illumination caused by light sources to be calculated by recursive ray tracing.

$I_\varepsilon\left(x' \to x\right)$ can be further separated to create the basic concept of the new algorithm:

$$I_\varepsilon\left(x' \to x\right) = \sum_{i=1}^{m} I_{\varepsilon_i}\left(x' \to x\right) \quad (2)$$

with $I_{\varepsilon_i}\left(x' \to x\right)$ being the intensity reflected off x' in direction of x that originated at only one single light source ε_i.

Integrating global illumination by photon mapping leads to ε_A being Jensen's radiance estimate. Since each photon in the estimate originated at an individual light source, $I_{\varepsilon_A}\left(x' \to x\right)$ can be expressed as

$$I_{\varepsilon_A}\left(x' \to x\right) = \sum_{i=1}^{m} I_{\varepsilon_{A,i}}\left(x' \to x\right) \quad (3)$$

with $I_{\varepsilon_{A,i}}\left(x' \to x\right)$ being the intensity of the global illumination reflected off x' in direction of x that originated at only one single light source ε_i.

Most stage lights can be modelled by an infinitely small lightsource. Generally they are positioned too far away to cause noticeable penumbrae – at least in the rendered image on a computer screen. But, stage lights do not emit light homogeneously.

Next, the prerequisites for integrating real stage lights are modelled. Intensities are calculated for n color channels. Let $f_{\varepsilon_i}\left(P_i \to x'\right) \in [0,1]^n$ be a function defining the emission pattern of the i-th lightsource with regard to direction from its position P_i to a point x'. The maximum intensity $\hat{I}_i$ of a fixture can be determined from the datasheet. Furthermore a light color $c_{I_i} \in [0,1]^n$ can be determined from the correlated color temperature of the used light bulb. The intensity of a stage light can be regulated by a dimmer with a factor $v_i \in [0,1]$. Finally a color filter can be attached to the fixture modelled by $c_{F_i} \in [0,1]^n$. To rerender intensity and color changes quickly, c_{F_i}, c_{I_i}, v_i and $\hat{I}_i$ are combined to a common coefficient k_{ε_i} of the light. Let $\overline{\varepsilon_0} = 1$ be the inten-

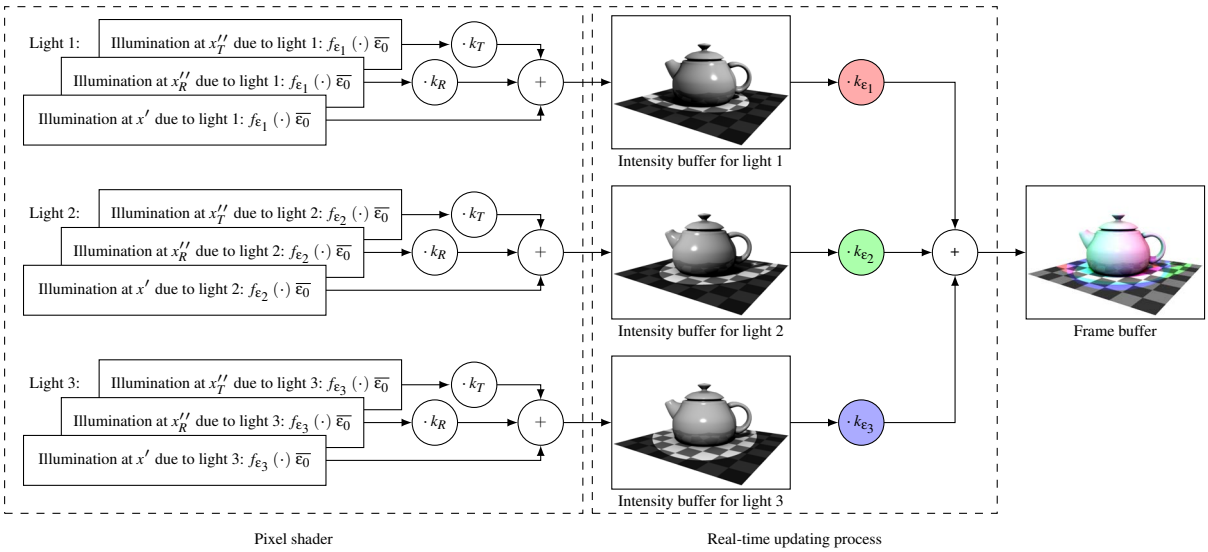

Figure 3: *Block diagram of the proposed pixel shader with an additional real-time updating process. This figure illustrates the calculation of the final illumination for three lights and one level of recursion for reflection and transmission.*

sity of a homogeneosly emitting replacement source, then

$$\varepsilon_i = c_{F_i} c_{I_i} f_{\varepsilon_i}\left(P_i \to x'\right) v_i \hat{I}_i \overline{\varepsilon_0} = k_{\varepsilon_i} f_{\varepsilon_i}\left(P_i \to x'\right) \overline{\varepsilon_0} \quad (4)$$

Since the flux of each photon in the radiance estimate is proportional to the source's intensity, equation 4 can not only be integrated into equation 2 but also into equation 3. Thus, both can be combined to create the final algorithm

$$I\left(x' \to x\right) = \sum_{i=1}^{m} k_{\varepsilon_i}\left[I_{h,\varepsilon_{A,i}}\left(x' \to x\right) + I_{h,\varepsilon_i}\left(x' \to x\right)\right] \quad (5)$$

with $I_{h,\varepsilon_i}\left(x' \to x\right)'$ being the intensity determined by ray tracing that originated only at the replacement lightsource with intensity $\overline{\varepsilon_0} = 1$ and with the same emision pattern as the i-th real light source. $I_{h,\varepsilon_{A,i}}\left(x' \to x\right)$ is the respective intensity determined by global illumination. With the definition of a maximum recursion depth it can be shown by induction that equations 1 and 5 are equal for any finite recursion depth. The proof has been left out due to space restrictions.

From equation 5 an implementation can be created that consists of a rendering step utilizing the term im brackets as a pixel shader. To do so, results have to be rendered to individual intensity buffers – one per light. Furthermore Jensen's photon maps have to be split into individual maps – again one per light. Thus, whenever positional changes of one light occur, only the photon map of this light has to be recreated and the respective buffer has to be rerendered. To allow for fast updates due to intensity or color changes, a subsequent real-time updating process is necessary that implements the sum and the coefficients k_{ε_i}. For a quick overview, figure 3 illustrates the algorithm for three lights and one level of recursion by a block diagram.

3.2. Integrating real stage lights

For real stage lights the maximum intensity $\hat{I}_i$ can be determined from the fixture's datasheet. Inside the fixtures two types of lightbulbs are typically used: Tungsten halogen and discharge (HMI) lamps. Both emit light at different intensities and with different correlated color temperatures. The former nominally with 3,200 K the latter with 6,000 K. To determine a suitable value for the light's color c_{I_i} the Planckian locus has to be evaluated for the respective color temperatures: The radiated intensity at a given wavelength can be calculated utilizing Planck's law. With suitable color matching functions [CIE09] these intensities can be expressed as a color in CIE XYZ color space. Afterwards they have to be converted to sRGB color space [SACM95] to be used by the renderer.

The intensity of theatre lights can be regulated by a dimmer: HMI lamps by mechanical shutters, tungsten halogen lamps by varying the supply voltage. Since the filament's temperature varies as the supply voltage is varied, the correlated color temperature of the emitted light and thus c_{I_i} varies. According to [FB78, Osr03] the color temperature is related to the dimmer value by $T_i \approx \sqrt[8]{v_i} \cdot \hat{T}_i$, with $\hat{T}_i$ being the nominal color temperature. Consequently, the light of tungsten halogen lamps gets redder the more they are dimmed.

A color filter can be inserted into real stage lights. The transmission spectra of these filters can be found in the fil-

ter's datasheet. Applying these spectra as filter functions to the Planckian locus leads to an sRGB color of the filtered light. The value of c_{F_i} can be determined from this color and the color of unfiltered light.

The radiation pattern of a stage light can be found in the fixture's datasheet. Typically two types of fixtures are used in the theatres: Profile or zoom spots with multiple lenses and fresnell or PC spots with one fresnell or plan-convex lens. The former can be modelled as spot lights with cone angle and penumbra angle. Both angles are variable with zoom spots, whereas profile spots have a fixed cone angle and a variable penumbra angle. The cone angle of zoom spots can be adjusted by the zoom value and by a mechanical iris. It is important to note that the maximum intensity $\hat{I}_i$ of a zoom spot varies while altering the zoom value but remains constant while opening or closing the iris. Zoom ranges, cone angles and the relationship of zoom value to intensity can be found in the datasheets.

Radiation patterns for PC and fresnell spots are a bit more elaborate than for profile or zoom spots. Fortunately these can also be found in the datasheets. Typically patterns for three focus ranges are presented: spot, middle and flood. They are modelled by piecewise-continuous Hermite polynomials in the proposed rendering framework. Thus, patterns can easily be edited to match real lights. Finally, the polynomials are interpolated with respect to the focus value of the fixture to create the radiation pattern $f_{\varepsilon_i}\left(P_i \to x'\right)$. This function is used for direct lighting as well as for sampling the photons' direction during photon emission.

3.3. Real-time updating

With the proposed pixel shader, light-specific intensity buffers are created, each storing only $\left[I_{h,\varepsilon_{A,i}}\left(x' \to x\right) + I_{h,\varepsilon_i}\left(x' \to x\right)\right]$. The remainder of equation 5 is implemented in the real-time updating process: During each display refresh, k_{ε_i} is calculated for each light from its intensity, dimmer value, correlated color temperature and filter color. The color values of the pixels are scaled according to k_{ε_i} in a suitable shader program. Furthermore, the shader program implements white balancing since the white point of the monitor does not match the dominant illuminant on a stage.

Having the color-scaled and white-balanced buffers available, these are summed up by rendering them into one common buffer with a suitable blending function. Finally this common buffer is rendered to the display's frame buffer. Again, a shader program is used that implements gamma correction as a simple means of tone mapping. Additionally, exposure values can be entered to manually adjust tone mapping for different intensities.

By integrating the real-time updating process, changes to k_{ε_i} affect the image within one screen refresh wehreas common renderers would recalculate the complete scene

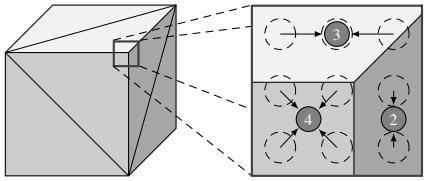

Figure 4: *Illustration of the adaptive antialiasing used in the rendering system: The pixel on the cube (marked with a square on the left) is nine-times oversampled (dashed circles on the right). Samples on the same triangle are combined to reduce the number of primary rays (filled gray circles). The weight of each combined sample is printed in white.*

3.4. Primary intersections and adaptive antialiasing

To speed up the presented rendering process, it was not only designed to rerender only those lights that changed, but also to detect whether the camera has been moved: As long as the camera remains static, the same intersection points of primary rays and objects can be used for every recalculation. Thus, an additional buffer was implemented to store primary intersections. For performance reasons – inspired by [WHG84] – these intersections are not determined by ray tracing but entirely on the GPU utilizing common API – i. e. OpenGL.

Weghorst *et al.* proposed to determine only the visible-surfaces for every pixel [WHG84]. In this work, however, intersection points, normals and triangle IDs is created during three OpenGL-rendering passes. The results are read back from the frame buffer into main memory to be used for rendering, later.

To allow for antialiasing, intersections are rendered with nine-times oversampling. Since not all of the samples are necessary to create a smooth image in the rendering process, intersections are preprocessed: For each pixel, intersections are binned according to their triangle ID. The number of bins in the resulting histogram determines the number of samples needed for the final rendering. The number of intersections in a bin detemines the weight of each sample. Positions and normals of the intersections are interpolated for the respective sample (cf. fig. 4). Thus, the number of samples and therefore the number of primary rays can be greatly reduced while preserving image quality.

Since the above antialising is based on the triangle ID only, light and shadow boundaries are not smoothed. To allow for antialiasing of the illumination, the renderer determines the gradient from the current pixel to the neighbouring pixels. If the gradient is above a certain threshold, the current and the neighbouring pixels are rerendered using all nine samples from the intersection buffer. It turned out that thresholds between 15 and 25 are a suitable trade-off between image quality and rendering speed. By antialiasing

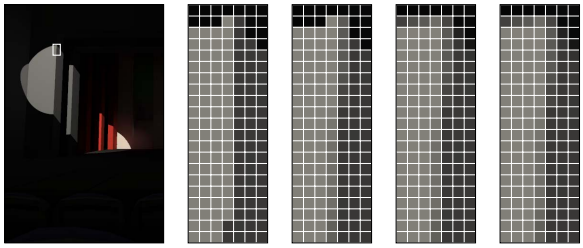

Figure 5: *Results of antialiasing: Image (a) shows a part of a rendered image. A scaled version of the highlighted region is shown on the right: rendered without antialiasing (b), rendered with antialiasing the geometry (c) and rendered with antialiasing both the geometry and the illumination (d). For comparison the region was also rendered with nine-times oversampling (e).*

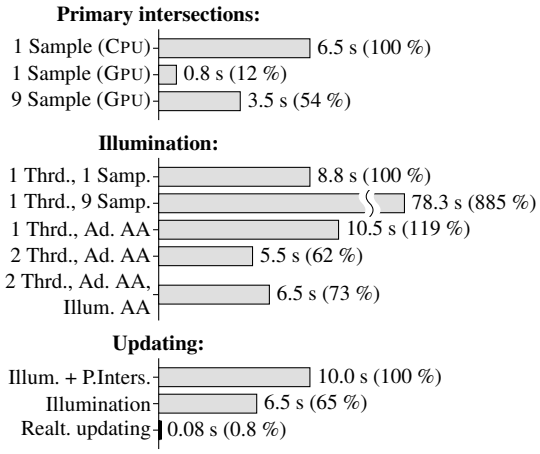

Figure 6: *Render times for calculating primary intersections (top), calculating the illumination (middle) and updating (bottom).*

the illumination, spotlight and shadow boundaries are also smoothed.

4. Results

The proposed algorithm was implemented in C++ on an *Apple MacBook Pro* with an *Intel Core 2 Duo* processor running at 2.33 GHz with 2 GB RAM using *Mac OS X 10.6*. It had an *ATI Radeon X1600* GPU with 256 MB VRAM.

To evaluate render times and the quality of antialiasing, a stage set inspired by the Cornell Box was used and illuminated by a single spot light. The scene contained 38,000 triangles. A specular material was attached to 24 percent of these. The scene was rendered at a resolution of 800 by 500 pixels with a maximum recursion depth of five. Per photon map 50.000 photons were stored with 150 photons in the radiance estimate. A part of the rendering with global illumination is shown in figure 5a.

To evaluate the rendering quality in terms of suitability for simulating theatre stage lighting, a real stage set was photographed and the 3D model of the same set was rendered with the proposed algorithm (cf. fig. 7). The model of the real stage set contained 42,000 triangles, 15 percent of them with a specular material.

4.1. Antialiasing quality

Typically, models of a stage set contain many faces that are much larger than a single pixel. Thus, to speed-up antialiasing, an adaptive method was proposed in this paper.

Figure 5a shows a part of a rendering. A scaled version of the highlighted region is shown alongside: Once rendered without antialiasing (cf. fig. 5b), once rendered with antialising only the geometry (cf. fig. 5c). It can be seen that the edge of the geometry was smoothed whereas the boundary of the lit area is not. Antialiasing both the geometry and the illumination as proposed in this work, leads to smooth geometry edges and light boundaries (cf. fig. 5d). For visual evaluation the result of nine-times oversampling is shown in figure 5e. Both results, the one of the proposed method and the one of nine-times oversampling, are very similar.

To evaluate the proposed method numerically, mean squared errors (*MSE*s) were determined for images with color values $RGB \in [0,1]^3$. The results of nine-times oversampling were used as a reference. For ten renderings an average $\overline{MSE}_{RGB} = (0.006, 0.004, 0.0004)$ was determined for the proposed method. Since images created with photon mapping expose a certain variance, the *MSE*s for pairs of images, created with nine-times oversampling, were calculated as a baseline: For ten image pairs an average $\overline{MSE}_{RGB} = (0.005, 0.003, 0.003)$ was determined. Although the average *MSE* for the proposed method was slightly higher than the baseline, no statistically significant difference could be found, utilizing a two-tailed, unpaired t-test ($t(18)_{RGB} \approx (0.78, 1.00, 1.03)$, $p_{RGB} \approx (0.45, 0.33, 0.32)$).

4.2. Render times

To evaluate the speed-ups achievable with the proposed rendering system, the scene was rendered with single sample ray tracing as a reference. At first, the intersection calculation process was evaluated separately (cf. fig. 6 top). Implementing this process on the GPU by OpenGL – as inspired by [WHG84] and specialized in this work – speeded it up by a factor of eight. Even when calculating nine samples per pixel, a total speed-up by a factor of two was achieved – a statistically highly significant speedup compared to the

reference ($t\,(18) \approx 11.3$, $p \ll 10^{-3}$). Most of the time was spent sorting and preparing the intersections for rendering.

Next, the illumination calculation process was evaluated (cf. fig. 6 middle). The calculation using one sample in a single thread was used as a reference. Full nine-times oversampling lead to a slowdown by a factor of nine. Implementing adaptive antialiasing as proposed in this paper led to a far lower slow-down: Rendering took 119 % of the reference time. This could be more than compensated for by running the algorithm in multiple threads. Running only two threads even compensated for slow-downs due to the additional integration of antialiasing the illumination. Finally the proposed algorithm runs around 1.4-times faster than the single-threaded implementation without antialiasing at all – also a statistically highly significant speedup compared to the reference ($t\,(18) \approx 47.9$, $p \ll 10^{-3}$).

Having this measures available, the recalculation speedup can be evaluated (cf. fig. 6 bottom): Rerendering one light completely was used as a reference – i.e. calculating primary intersections and illumination with adaptive antialiasing as presented. Although existing software packages implement different techniques, rerendering is based on the same algorithm (cf. section 3). Thus, this reference value corresponds to the render times of existing software packages. If position or radiation pattern of a fixture were changed, only the illumination has to be rerendered. This process ran 1.5-times faster than a complete rerendering. If only color or intensity of a light were changed the real-time updating process runs even 125-times faster than the complete rerendering and still 83-times faster than recalculating only the illumination with precomputed intersections.

4.3. Render quality

The quality of rendering a real stage set was inspected visually. Figure 7 shows photos taken in the theatre in the upper row. The lower row of the figure shows renderings of the same stage set created with the proposed system. A larger color-version of this figure can be found in the color plates section.

For figures 7a and 7d the scene was illuminated by one zoom spot *Niethammer HPZ 115 D* from front above and its virtual counterpart respectively. Both were set to 100 % intensity with no filter. For figures 7b and 7e one fresnell spot *Strand Castor* was used from front left. It was set to 100 % intensity with a *LEE* 201 filter. For figures 7c and 7f both prior illuminations were combined plus four additional HMIs *Arri Compact 2500 T* from behind above. These were set to 50 % intensity with a *LEE 136* filter.

It can clearly be seen that real and virtual illumination are very but not completely similar. However, in a spontaneous interview, users rated the quality of the images rendered with the prototypic system as absolutely sufficient for planning theatre illumination. They furthermore rated update-speed way more important than total accuracy.

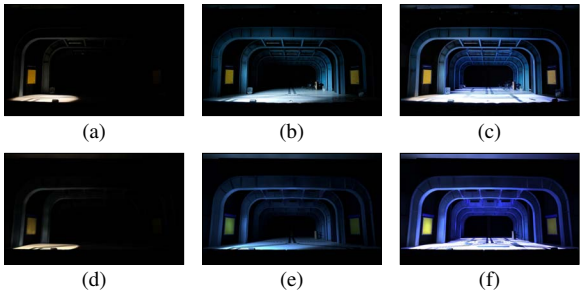

Figure 7: *Comparison of photos and renderings. Images (a) to (c) show photos taken in the theatre, images (d) to (f) show renderings created with the proposed system. (for a larger version see Color Plate)*

5. Conclusion and future work

In this paper, we presented a new rendering algorithm based upon photon mapping to be used for lighting simulation in the theatres. The algorithm was designed to match the workflow during lighting rehearsals – i. e. once a light is positioned and aimed correctly, many color and intensity values are tested. Thus, these attributes must affect the rendered image immediately. Otherwise, such a planning tool would not be useful for the head of the lighting department or a stage designer.

The implementation proved to be useful in terms of rendertimes and image quality: A complete rerendering could be speeded up by the proposed techniques to provide even antialiasing. The seperate real-time updating process allows for intensity and color changes more than 80-times faster than a recalculation of the illumination and 125-times faster than a complete rerendering. Therefore, many lghting setups can be easily evaluated before even building the set on stage. Thus, the ammount of testing to be done during rehearsals is reduced.

With the insight gained from the prototypic system, many more speedups can now be applied that are readily available in literature. Having a fast renderer available, a well-suited intuitive stage set editor is still missing. Finally, serious effort has to be spent to integrate real materials and an elaborate tone mapping step into the system to make color display more realistic. Furthermore volume scattering and participating media have to be addressed, since fog or haze are used in several theatre productions. But, the rendering speed of the presented system makes users to overlook even the slightly inaccurate color rendering or missing features.

Acknowledgment

The authors wish to thank the director Mr. Joan Anton Rechi and the production team of *Die Oberhausener Johannes-Passion* at Theatre Oberhausen, Germany, for kindly supporting this work. Special thanks to Mr. Alfons Flores, the stage designer, for making available the 3D model of the stage.

Many thanks to Mr. Mathias Leonhardt for offering his findings and implementations as fundamentals of this work.

Many thanks also to Mr. Dan Redler, curator of the Compulite-Danor Stage lighting museum in Hod Hasharon, Islael, for invaluable information.

References

[Art] ARTVPS: Shaderlight™. http://www.artvps.com/content/shaderlight/what-is-shaderlight. Last visited: 2010-07-14. 2

[BEL*07] BOULOS S., EDWARDS D., LACEWELL J. D., KNISS J., KAUTZ J., SHIRLEY P., WALD I.: Packet-based whitted and distribution ray tracing. In *GI '07: Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), ACM, pp. 177–184. 2

[cas10] CAST SOFTWARE: wysiwyg. https://www.cast-soft.com/cast/products/meetwysiwyg.php, 2010. Last visited: 2010-07-14. 2

[CB04] CHRISTENSEN P. H., BATALI D.: An irradiance atlas for global illumination in complex production scenes. In *Rendering Techniques* (2004), Keller A., Jensen H. W., (Eds.), Eurographics Association, pp. 133–142. 2

[CIE09] CIE: Commission internationale de l'eclairage: Selected colorimetric tables – cie 1931 standard colorimetric observer. http://files.cie.co.at/204.xls, 2009. Last visited: 2010-07-14. 4

[CSh09] CSHAKE: Theatresys theatrical lighting system. http://news.povray.org/povray.binaries.scene-files/thread/49c119b9@news.povray.org/, 2009. Last visited: 2010-07-14. 2

[FB78] FINK D. G., BEATY H. W. (Eds.): *Standard Handbook for Electrical Engineers*, 11 ed. McGraw-Hill, New York, 1978. 4

[GTGB84] GORAL C. M., TORRANCE K. E., GREENBERG D. P., BATTAILE B.: Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), ACM, pp. 213–222. 2

[HH10] HAVEL J., HEROUT A.: Yet faster ray-triangle intersection (using sse4). *IEEE Transactions on Visualization and Computer Graphics 16*, 3 (2010), 434–438. 2

[HJ09] HACHISUKA T., JENSEN H. W.: Stochastic progressive photon mapping. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–8. 2

[Jen01] JENSEN H. W.: *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001. 2, 3

[Kob88] KOBBÉ G.: Behind the scenes of an opera-house. *Scribner's Magazine 4*, 4 (1888), 435–454. 1

[Lam60] LAMBERT J. H.: *Photometria: sive de mensvra et gradibvs lvminis, colorvm et vmbrae*. 1760. 2

[Lan10] LANGHANS C. G.: *Über Theater oder Bemerkungen über Katakustik in Beziehung auf Theater*. Hayn, Berlin, 1810. 1

[Lux] LUXRENDER: Luxrender – gpl physically based renderer. http://www.luxrender.net/. Last visited: 2010-07-14. 2

[MA] MA LIGHTING: grandma 3d. http://www.malighting.com/control.html?\&L=2\&tx_lightpowerpdb_pi1[parent_gruppe]=233\&tx_lightpowerpdb_pi1[produkt_id]=3955\&cHash=d64bdc9dc4. Last visited: 2010-07-14. 2

[Men] MENTAL IMAGES: Mental ray® photorealistic rendering software. http://www.mentalimages.com/products/mental-ray.html. Last visited: 2010-07-14. 2

[ML09] MCGUIRE M., LUEBKE D.: Hardware-accelerated global illumination by image space photon mapping. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), ACM, pp. 77–89. 2

[Osr03] OSRAM SYLVANIA: Properties of tungsten filament lamps. OSRAM SYLVANIA: Sylvania Automotive Lighting Catalog http://www.sylvaniaautocatalog.com/sylvania/tung_fila_lamps.htm, 2003. Last visited: 2010-07-14. 4

[PDC*03] PURCELL T. J., DONNER C., CAMMARANO M., JENSEN H. W., HANRAHAN P.: Photon mapping on programmable graphics hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 41–50. 2

[Per] PERSISTENCE OF VISION RAYTRACER PTY. LTD.: Persistene of vision raytracer™. http://www.povray.org/. Last visited: 2010-07-14. 2

[PH04] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. 2

[Pho75] PHONG B. T.: Illumination for computer generated pictures. *Commun. ACM 18*, 6 (1975), 311–317. 3

[Pix] PIXAR: Renderman®. https://renderman.pixar.com/. Last visited: 2010-07-14. 2

[PVL*05] PELLACINI F., VIDIMČE K., LEFOHN A., MOHR A., LEONE M., WARREN J.: Lpics: a hybrid hardware-accelerated relighting engine for computer cinematography. *ACM Trans. Graph. 24*, 3 (2005), 464–470. 2

[Ran] RANDOMCONTROL: SWAP. http://randomcontrol.com/swap. Last visited: 2010-07-14. 2

[SACM95] STOKE M., ANDERSON M., CHANDRASEKAR S., MOTTA R.: *A Standard Default Color Space for the Internet - sRGB – Version 1.10*, 1995. 4

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004. 2

[WHG84] WEGHORST H., HOOPER G., GREENBERG D. P.: Improved computational methods for ray tracing. *ACM Trans. Graph. 3*, 1 (1984), 52–69. 2, 5, 6

[Whi80] WHITTED T.: An improved illumination model for shaded display. *Commun. ACM 23*, 6 (1980), 343–349. 2, 3

[WK06] WÄCHTER C., KELLER A.: Instant ray tracing: The bounding interval hierarchy. In *Rendering Techniques 2006: Proceedings of the 17th Eurographics Symposium on Rendering* (2006), pp. 139–149. 2

[WWZ*09] WANG R., WANG R., ZHOU K., PAN M., BAO H.: An efficient gpu-based approach for interactive global illumination. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–8. 2