HOLOGRAPHIC APPROACH FOR AN EYE SIMULATION

Lukas Novosad
Czech Technical University in Prague, Faculty of Electrical Engineering, Czech Republic
novosluk@fel.cvut.cz

Remo Ziegler
LiberoVision AG, Zürich, Switzerland
ziegler@liberovision.com

ABSTRACT

Virtual scenes used in state-of-the-art computer games and animated movies appear as realistic as possible by using the latest graphics algorithms and hardware available. This work aims to improve the realism in the way the scene is perceived by the user. Usually a pinhole camera is placed into the scene and the projected image is presented to the viewer. We want to improve the image which is presented to the user by simulating an eye, which is divided into different layers. Each one of these layers provides individual features. The approach is based on wave optics and is able to simulate effects of refraction, diffraction, high-dynamic range lighting, and depth-of-field. This simulation is implemented by using an NVidia CUDA device with its GPGPU capabilities and unified shader architecture. Our framework offers a simple interface which only requires access to the frame buffer and depth buffer. As a consequence it may be plugged into existing engines in a straightforward manner as a simple extension.

KEYWORDS

rendering, GPU acceleration, screen space effects

1. INTRODUCTION

1.1 Motivation

Animated movies and computer games entertain us in our daily lives. The perceived quality of these has improved significantly over the last few years. On the one hand, much effort has been put into photorealistic rendering. This includes very detailed models for lighting calculations, e.g. accurate BRDFs, taking into account not only diffuse and specular lighting, but also phenomena such as light refraction, diffraction, and scattering. On the other hand, the interaction between the modeled objects and elements such as gases and liquids in the scene is supported by numerous physically-based simulations. This makes water pearl off on materials like wax or wooden boxes can be destroyed such that the wood splinters into all directions in a realistic fashion. One important factor for this improvement is the fast development of graphics hardware. This enables algorithms and techniques, which needed seconds, minutes, or even hours of computational time for a single frame, to achieve real-time behaviour nowadays. Furthermore, the exploration of completely different approaches, which break up the standard graphics pipelines such as DirectX and OpenGL with their triangle-based projection and rasterization, require new hardware yet to be invented. However, the eye through which the user sees the virtual scene or interacts with it is almost always simulated as a pinhole camera. This corresponds to an evaluation based on geometric optics, which is the simplest way to treat light. It models light as rays or particles that move in straight lines. The scalar diffraction theory models light as a real-valued function of time and space. Maxwell showed that light is in fact an electromagnetic wave, which is a real-valued function of space, time, and direction. This is described in the vector diffraction theory. Finally, the most accurate model of light is defined by quantum optics and was introduced by Einstein with the help of quantum mechanics. We implement an eye simulation which is based on the scalar diffraction theory to achieve a higher level of realism as how virtual scenes may be rendered.

1.2 Related work

The ways how virtual worlds are perceived are improved by shader programs and post-processing effects. Shader programs can, for example, change the geometry of objects in the scene before they are shown to the viewer. Characters can be skinned such as in (James & Twigg, 2005). Furthermore, shaders can work on the object surfaces and materials to alter the lighting evaluations. One example for this would be classical bump-mapping (Blinn, 1978) or modern parallax-mapping (Tatarchuk, 2006) which adds artificial depth to textures. Another good example is the subsurface-scattering effect for human skin as described in (Koenderink et al., 1996).

Post-processing effects then again work on the rendering pipeline's output data. Consequently, the output of a post-processing effect is an altered frame buffer image. These effects often change the way how a scene is visually perceived by the user. One of the most popular post-processing effects is high-dynamic range (HDR) lighting. Intensive work was done by (Debevec et al., 1999) and by (Ward & Simmons, 2005). (Ledda et al., 2004) model the eye adaptation over time based on physiological data. Since common displays have a limited range of luminance values, input data from sources with a broader range must be fitted into the screen's displayable range. We use the method introduced by (Reinhard et al., 2002) to tone map our input data and found that this produces good results. However, there are displays which provide a significantly larger range of intensities values than the currently available displaying devices. An HDR displaying device is presented in (Seetzen et al., 2004), for example. Another post-processing effect is motion blur as in (Potmesil & Chakravarty, 1983), (Brostow & Essa, 2001) or depth-of-field blur as in (Hillaire et al., 2007). Our eye simulation corresponds to a post-processing effect.

There has been intensive research in the area of the human visual system, and how it may be simulated. Very interesting is the work done by (Deering, 2005) where electromagnetic waves are refracted through a four layer eye model. A synthetic retina model is used where each of the photoreceptor cones is modeled individually. The goal of our simulation is different, since we focus on flexibility and performance, while the retinal model focuses on accuracy instead. The way how glares and bloom effects may be created is, for example, discussed in (Spencer et al., 1995) and (Kakimoto et al., 2004). Similar to (Kakimoto et al., 2004), we can simulate the effect of eye lashes which are modeled as a grating where light is diffracted. While we simulate the first-person view through an eye, there is also work on how the human eye can be rendered. In (Lam & Baranoski, 2006) a layered model is used with each layer having a different refractive index based on its tissue.

Two new rendering pipelines are introduced in (Zwicker et al., 2002) and (Ziegler et al., 2007). (Zwicker et al., 2002) present a pipeline which is completely based on point primitives and incorporates various editing capabilities for 3D point-sampled geometry. The holography rendering pipeline introduced in (Ziegler et al., 2007) provides the possibility to set up a scene and evaluate it using wave optics. It may record holograms and reconstruct them by propagating the reconstructed light wave and rendering an image from the evaluation of the wave function by using a virtual camera. The final image is evaluated at an aperture, which can be modified in size and shape.

1.3 Contribution

Our work aims to improve the way how a scene can be perceived by the user. As previously mentioned, a game or movie scene is usually presented to the user by placing a camera into the scene and setting the appropriate parameters such as field of view, near plane, and far plane. The perspective projection onto this placed camera is what the user sees on his displaying device. This approach basically corresponds to the scene being evaluated with geometric optics and a pinhole camera. We extend this approach by introducing a post-processing effect which evaluates the output received from the frame buffer and depth buffer, and create an altered frame buffer as a result. As a consequence, the eye simulation is very flexible and can be integrated into existing frameworks as a simple extension. Our evaluation uses wave optics instead of geometrical optics. This ultimately leads to an improved level of realism. The idea of using wave optics in computer graphics was first posed in (Moravec, 1981). But the hardware available at that time was not capable to achieve reasonable performance not to speak of real-time behaviour. Recently more research was done in this area. The holography pipeline introduced in (Ziegler et al., 2007) contributes to this development. Parts of this pipeline are used to implement our eye simulation.

2. EYE SIMULATION

Similar to (Deering, 2005) and (Kakimoto et al., 2004) we have chosen a layered approach for our design. This offers the possibility to add new features or modify existing ones when new technologies become available. We require only the frame buffer and the depth buffer as input to our eye simulation. These buffers may be generated by any engine, no matter how detailed and complicated the calculations in this engine are. The eye simulation works on these input buffers and creates a new image as output, which is presented on a displaying device. This leads to a simple and flexible interface, which eases the integration into existing frameworks. We assume that the scene is mainly composed of diffuse materials and we have limited information about the scene since we provide a screen-space effect. Figure 2.1 serves as an illustration. Finally, we integrated our eye simulation into the *HoloPipeline* introduced in (Ziegler et al., 2007).

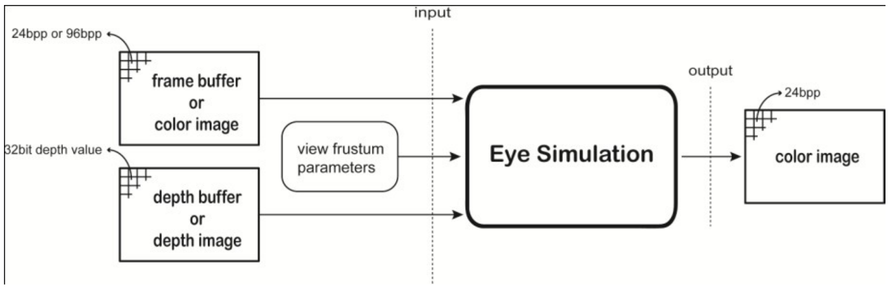

Figure 2.1: The I/O interface to the eye simulation

2.1 Scalar diffraction theory

We provide a short introduction into the scalar diffraction theory and describe monochromatic waves. The reader should be familiar with basics in wave propagation such as diffraction, interference, and superposition. For more detailed information refer to (Goodman, 1968).

The relations between the electric field E and the magnetic field H are described by Maxwell's equations. If the medium of propagation is homogeneous and there are no boundary conditions present, the vector components can be treated individually. The error introduced by this separation is negligible at a distance of several wavelengths from the point of interaction between the light and the material. Since we evaluate objects in our eye simulation which are rather far away from the used aperture, we use the scalar diffraction theory. As a consequence, we will only consider monochromatic waves from now on, e.g. we treat each color channel separately in our eye simulation with the three corresponding wavelengths λ_r, λ_g, and λ_b.

A monochromatic plane wave can be described as

$$U(P) = Ae^{i\boldsymbol{k}\cdot\boldsymbol{r}+\varphi} \qquad (2.1)$$

where A is the initial amplitude at the origin P_0 and φ the corresponding initial phase. $\boldsymbol{k}$ is the wave vector defined as $\boldsymbol{k} = k * (\alpha, \beta, \gamma)$, where k is the wave number and (α, β, γ) is the direction of propagation as a unit vector. Finally, $\boldsymbol{r}$ is defined as $\boldsymbol{r} = P - P_0$.

A spherical wave expands in all directions with the same velocity and the wavefront at any point in time corresponds to a sphere. The amplitude decreases inversely proportional to the distance between the point source and the wavefront due to the law of conservation of energy. A spherical wave has the following form

$$U(P) = A\frac{e^{(ikr+\varphi)}}{r} \qquad (2.2)$$

where the distance r is the length of the vector from the source to the point of evaluation $r = \|P - P_0\|_2$.

2.2 Eye layers

Each eye layer implements one of the physical eye properties. We model the eye layers as thin optical elements. As a result the effects applied by our eye layers correspond to phase shifts and amplitude modulations. All eye layers except the base layer can be activated or deactivated as required. The eye we use in our simulation is divided into five different layers. In order to improve the performance and simplify the processing complexity of our eye simulation, all eye layers have the same dimensions and are oriented parallel to each other. Figure 2.2 illustrates the eye structure which we use for our simulation.

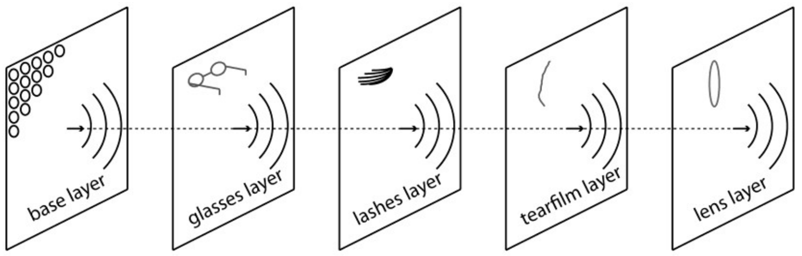

Figure 2.2: The eye structure used for the simulation

The base layer evaluates the whole holographic scene, which in our case consists of point sources. Each point source emits a spherical wave which influences the whole base layer. Figure 2.3 depicts this setup for a simple object.

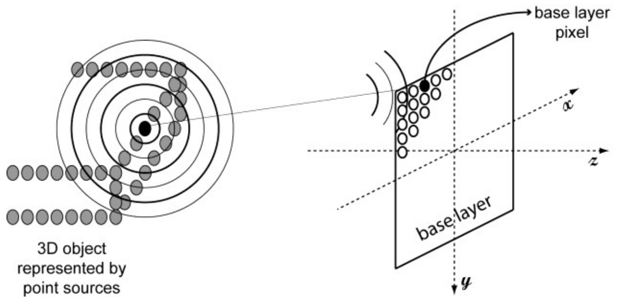

Figure 2.3: Each point source emits a wave a can potentially influence all base layer pixels

If the character viewing the 3D scene is wearing glasses, the glasses layer provides the necessary functionality by altering the phase of the incoming wavefront. This shapes the waves in order to appear in focus.

Eye lashes have a large influence onto the human visual system. They diffract the incoming light waves and we add this glare effect to our eye simulation. The glare effect is stronger with high intensities of the incoming waves. We use similar textures for our eye lashes layer as in (Kakimoto et al., 2004). More specifically, we perform a convolution, by transforming both the wavefront and the eyelashes texture (see Figure 2.4) into Fourier space, multiplying them, and finally transforming them back from the Fourier space.

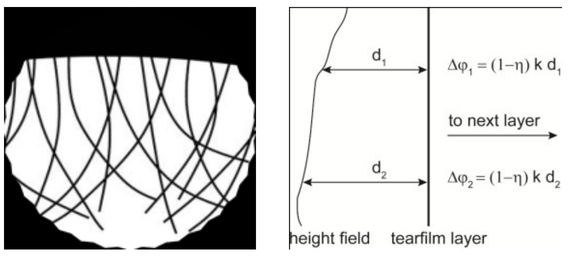
d_1
$\Delta\varphi_1 = (1-\eta)\ k\ d_1$
to next layer
d_2
$\Delta\varphi_2 = (1-\eta)\ k\ d_2$
height field
tearfilm layer

Figure 2.4: The left side shows the texture for the eyelashes and the right side illustrates the phase delay of the tearfilm

The tearfilm layer simulates the tearfilm which covers the cornea of the eye. It should be able to create effects such as tears or blood sprinkles, which could, for example, be used for a computer game, or if the model is improved, even for animated action movies. Currently, the layer uses a height field to represent the tearfilm. This field can be either static or dynamic, depending upon whether the eye simulation is used for still images or for dynamic scenes. Since we treat our layers as thin optical elements, this height field introduces a phase delay $\Delta\varphi$ to the propagated wavefront as depicted in Figure 2.4, where k represents the wave number, η is the refractive index of the tearfilm liquid, and d corresponds to the height field value.

The final layer, the eye lens, can be used to focus a specific area, which leads to a limited depth-of-field and variable focal length. We use a simple lens which is provided by the *HoloPipeline*.

2.3 Processing

The processing starts by reading in the frame buffer data and creating the appropriate number of point sources. This means that for an input consisting of *width*height* pixels, we create an equal amount of point sources. Additionally, we need the depth buffer data if exact depth-of-field simulation is required. These point sources are evaluated at the base layer and the resulting wavefront is propagated through the other eye layers. In order to reduce speckle noise we perform several iterations. For each iteration we randomize all the point source phases and evaluate them anew. Then, the newly created wavefront is again propagated through all the active layers. We sum over all these iterations and calculate the average. Speckle noise is a static interference pattern present in images based on coherent light. Two propagating waves are called coherent, if the change in one of the waves allows predicting the change in the other wave. Holographic approaches are based on coherent light.

We have decoupled the resolution which we use for the output image from the resolution of the input data. This means that it is possible to use small input data sizes while keeping the output resolution high. This leads to a smaller amount of generated and evaluated point sources, which in turn results in a better performance and reduction of speckle noise. However, the quality of the output image decreases since the smaller amount of point sources also leads to a lower precision.

2.3.1 Point source creation

In order to create a point source, we need several components. We obtain the color from the corresponding frame buffer location, and set the amplitude to a constant value which is the same for all point sources. For bright pixels in HDR images, however, we set the amplitude to a high value in order to create glare effects. Furthermore, we assign a random phase to the point source. The phase is chosen random because we need to simulate a diffuse object hologram in order to create an image with the *HoloPipeline*. Finally, we need to specify the location of the point source. With the corresponding depth buffer value and the additional view frustum data we are able to reproject each pixel from the frame buffer to the former world space coordinates. And this is exactly the position where we create the point source. In case no depth buffer is provided to the eye simulation, a constant depth value can be used for all the point sources.

2.3.2 Wave-based evaluation

After creating all the point sources, we start the wave-based evaluation in the base layer by processing all the spherical waves emitted by the point sources. For each point source, we evaluate its influence on all pixels of the base layer according to Equation 2.2. This is illustrated in Figure 2.3. Besides the point source amplitude, phase, and color, we also need the distance from the point source to the point of evaluation on the base layer, which can be evaluated from the reprojected point. The resulting image is propagated to the next active layer, which in turn applies its own effect to the wavefront. This process continues until all active layers have been passed. It is important to note that the base layer is also made out of point sources, whereas the other layers are modelled differently, for example by textures.

The wavefront is propagated from layer to layer by a method called angular spectrum propagation. If the wave field given at a layer is Fourier transformed, the frequency components can be regarded as plane waves propagating away from this layer. These plane waves are described by Equation 2.1. The wave field across any other parallel plane can be evaluated by summing up the propagated plane waves of the angular spectrum. The plane wave can be propagated easily by shifting the phase of each plane wave. For a more detailed description refer to (Ziegler et al., 2007).

It is possible to extend this simple version in order to handle non-parallel planes of different size. However, the fact that we use equally sized layers which are all parallel to each other improves the propagation performance since no padding or resampling must be done on either the source or target plane. Furthermore, no rotation is required for either plane. When all active layers have been passed and processed, the image is evaluated at the virtual camera in the *HoloPipeline*. In case of an HDR input, the image is then tone mapped according to (Reinhard et al., 2002). The result is a color image with 8bit precision per color channel.

3. RESULTS & DISCUSSION

In order to illustrate the LDR data processing we rendered the Utah teapot, which is lit by a directional light source. Table 3.1 shows the times measured for the teapot renderings. The timings required to apply the individual layer effects range from one to ten milliseconds, depending upon the resolution and the effect. They provide a small contribution to the total runtime. We realized that the main bottleneck is given by the time, which is needed to evaluate all the point sources at the base layer. The system we used to create our results was a Pentium IV 660 3.6 GHz with 2GB DDR2 Ram and a GeForce 8800GTX.

Table 3.1: CPU time measurements for the Utah teapot

Rendering size	# Iterations	Activated layers	Time [s]
256x256	16	base	39
256x256	16	base and lashes	58
256x256	16	base, lashes, and tearfilm	75
512x512	16	base	245
512x512	16	base and lashes	314
512x512	16	base, lashes, and tearfilm	382
512x512	64	base	975
512x512	64	base and lashes	1269
512x512	64	base, lashes, and tearfilm	1563

Because our GPU version for the evaluation of the point sources uses a CUDA kernel, the evaluation is faster than the CPU version. However, it is not as fast as we expected. Furthermore, it was only possible to use the CUDA kernels for a rendering size of 256x256 point sources. This is due to the GDI watchdog timeout. As a consequence, any CUDA kernel may only run for a maximum time of five seconds, which is the GDI watchdog timeout on Windows XP systems. There are ways to circumvent this limitation but unfortunately we did not have the resources to do so. We provide a comparison between the CPU and GPU version in Table 3.2. The speed up factor depends on the resolution and the number of black pixels present in the input image because black pixels are not evaluated by the CPU whereas the current GPU version does not make any distinction.

Table 3.2: Comparison between the CPU and the GPU version of the point source evaluation procedure for the Utah teapot example

Rendering Size	Type	CPU version [ms]	GPU version [ms]
128x128	LDR	197	44.4
128x128	HDR	1020	45.5
256x256	LDR	973	702
256x256	HDR	4328	703

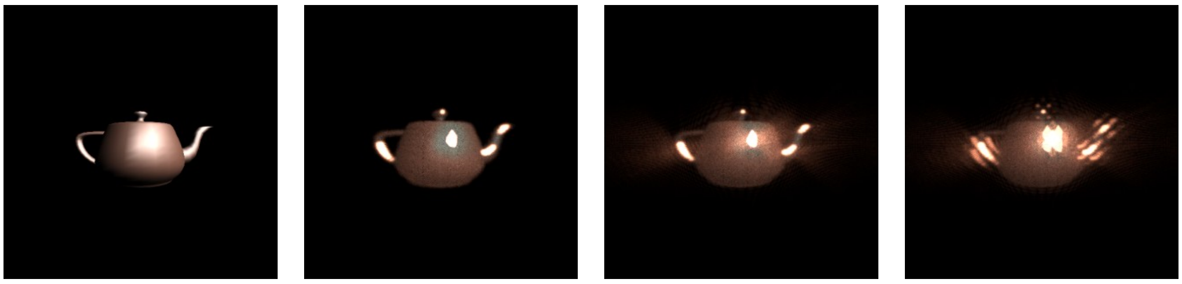

Figure 3.1: Utah teapot, 512x512 input & output, 64 iterations. From left to right: input image - base layer - base and lashes layer - base, lashes, and tearfilm layer

Figure 3.1 shows the renderings of the Utah teapot. The pictures illustrate different layer combinations. The first image shows the initial rendering done by DirectX, which provides the input to our eye simulation. We can see glare effects if the lashes layer is used and a perturbation effect is added by the tearfilm layer. The regular pattern is due to the simple, sinusoidal height field we use at the moment.

3.1 HDR eye simulation

To test our tone mapping implementation, we used an .hdr image without depth map and we set the depth to a fixed value for all the point sources in the holographic scene. Again, we rendered different combinations of activated eye layers which are depicted in Figure 3.2. Because of the tone mapping procedure, the specular highlights on the car front are not altered by the eye lashes since their intensity is too low compared to the overhead light bulbs. The distortions, called coma-aberration, which can be well seen in the car example, are due to the fact that we used a simple lens model. The time measurements made with two different numbers of iterations and rendering sizes are shown in Table 3.3. We used the CPU version of the point source evaluation procedure.

Table 3.3: Time measurements for the HDR car example

Rendering size	# Iterations	Activated layers	Time [s]
256x256	16	base	153
256x256	16	base and lashes	171
256x256	16	base, lashes and tearfilm	191
512x512	64	base	5031
512x512	64	base and lashes	5453
512x512	64	base, lashes and tearfilm	5645

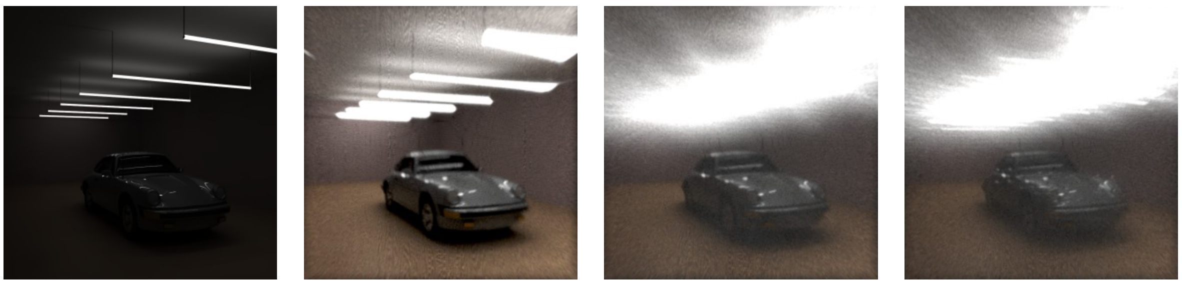
Figure 3.2: HDR car, 512x512 input & output, 64 iterations. From left to right: Initial .png image – base layer – base and lashes layer – base, lashes, and tearfilm layer

4. CONCLUSION AND FUTURE WORK

We proposed an eye simulation based on wave optics. This improves the camera model used for most rendering engines, where usually geometric optics is used to evaluate the scene. Our model is based on the scalar diffraction theory and the final eye simulation consists of four layers; we did not implement the glasses layer. The approach provides a simple and general I/O interface, and may be easily integrated into existing frameworks. The output we generate simulates effects such as depth-of-field, diffraction, and high dynamic range lighting, and the results appear physically plausible. However, we were disappointed by the final performance of our eye simulation because we expected to achieve a far better performance by using CUDA and the GeForce8 GPU series form NVidia. As a consequence of the poorly improved performance, we dropped the idea of integrating our eye simulation into an existing real-time rendering engine. For this, more work has to be done in order to improve the performance of our eye simulation.

As for future work, one straightforward extension to our eye simulation would be new layers such as a retinal layer, for example. The simulation of the tearfilm layer could be improved by using a fluid simulation and the wavefront could be refracted through the liquid instead of using the thin optical element approximation. Other types of waves could be incorporated, for example, to simulate a temperature dependent vision. It would also be interesting to test our eye simulation on a more powerful device like an SLI-system or an NVidia Tesla GPU. Another exciting improvement would be to investigate the possibility to use the vector diffraction theory. This would enable us to simulate effects of light polarization. With this, new filters could be generated, which could, for example, highlight objects emitting a vertically polarized light wave. The sky emits vertically polarized light waves, for example. Finally a user study should be carried out in order to evaluate the efficacy of our approach. This would show whether our approach is useful or if the pinhole camera model is more favored by the user.

ACKNOWLEDGEMENT

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under research programs LC-06008 (Center for Computer Graphics) and MEB-060906 (Kontakt OE/CZ), and the Grant Agency of the Czech Technical University in Prague, grant No. SGS10/289/OHK3/3T/13.

REFERENCES

Blinn, J., 1978. Simulation of wrinkled surfaces. *SIGGRAPH Proceedings of the 5th annual conference on Computer graphics and interactive techniques*. New York, USA, pp. 286-292.

Brostow, G. J. and Essa, I., 2001. Image-based motion blur for stop motion animation. *SIGGRAPH Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. New York, USA, pp. 561-566.

Debevec, P. et al., 1999. Fiat lux. *SIGGRAPH Electronic art and animation catalog*. New York, USA, p. 133.

Deering, M. F., 2005. A photon accurate model of the human eye. *ACM Trans. on Graphics,* Vol. 24 No. 3, pp. 649-658.

Goodman, J. W., 1968. *Introduction to Fourier Optics*. Roberts & Company Publishers, San Francisco, USA.

Hillaire, S. et al., 2007. Depth-of-field blur effects for first-person navigation in virtual environments. *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*. New York, USA, pp. 203-206.

James, D. L. and Twigg, C. D., 2005. Skinning mesh animations. *ACM Trans. on Graphics,* Vol. 24, No. 3, pp. 399-407.

Kakimoto, M. et al., 2004. Glare Generation Based on Wave Optics. *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*. Washington, USA, pp. 133-142.

Koenderink, J. J. et al., 1996. Bidirectional Reflection Distribution Function Expressed in Terms of Surface Scattering Modes. *Proceedings of the 4th European Conference on Computer Vision - Volume II*. London, UK, pp. 28-39.

Lam, M. W. and Baranoski, G. V., 2006. A Predictive Light Transport Model for the Human Iris. *Computer Graphics Forum*, Vol. 25, No. 3, pp. 359-368.

Ledda, P. et al., 2004. A local model of eye adaptation for high dynamic range images. *AFRIGRAPH Proceedings of the 3rd intern. conf. on Comp. graphics, virtual reality, visual. and interaction in Africa*. New York, USA, pp. 151-160.

Moravec, H. P., 1981. 3D graphics and the wave theory. *SIGGRAPH Proceedings of the 8th annual conference on Computer graphics and interactive techniques*. New York, USA, pp. 289-296.

Potmesil, M. and Chakravarty, I., 1983. Modeling motion blur in computer-generated images. *SIGGRAPH Proceedings of the 10th annual conference on Computer graphics and interactive techniques*. New York, USA, pp. 389-399.

Reinhard, E. et al., 2002. Photographic tone reproduction for digital images. *SIGGRAPH Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. New York, USA, pp. 267-276.

Seetzen, H. et al., 2004. High dynamic range display systems. *ACM Trans. on Graphics,* Vol. 23, No. 3, pp. 760-768.

Spencer, G. et al. 1995. Physically-based glare effects for digital images. *SIGGRAPH Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. New York, USA, pp. 325-334.

Tatarchuk, N., 2006. Dynamic parallax occlusion mapping with approximate soft shadows. *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. New York, USA, pp. 63-69.

Ward, G. and Simmons, M., 2005. JPEG-HDR: A Backwards-Compatible, High-Dynamic Range Extension to JPEG. *Thirteenth Color Imaging Conference*. Springfield, USA, pp. 283-290.

Ziegler, R. et al., 2007. A Framework for Holographic Scene Representation and Image Synthesis. *IEEE Transactions on Visualization and Computer Graphics,* Vol. 13, No. 2, pp. 403-415.

Zwicker, M. et al., 2002. Pointshop 3D: An interactive system for point-based surface editing. *SIGGRAPH Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. New York, USA, pp. 322-329.