Real-Time Lens Blur Effects and Focus Control

Figure 1: *Example images rendered in real time by our method. We achieve near-accurate depth-of-field effects, including lens aberrations (e.g., spherical aberration, left). The efficiency of our method makes it well-suited for artistic purposes and we support complex simulations like tilt-shift photography (middle). Further, our system offers an intuitive control of depth of field and we extend the physical model (middle right) to achieve an expressive, yet convincing result (right, where the background statues are still focused).*

Abstract

We present a novel rendering system for defocus-blur and lens effects. It supports physically-based rendering and outperforms previous approaches by involving a novel GPU-based tracing method. Our solution achieves more precision than competing real-time solutions and our results are mostly indistinguishable from offline rendering. Our method is also more general and can integrate advanced simulations, such as simple geometric lens models enabling various lens aberration effects. These latter are crucial for realism, but are often employed in artistic contexts too. We show that available artistic lenses can be simulated by our method. In this spirit, our work introduces an intuitive control over depth-of-field effects. The physical basis is crucial as a starting point to enable new artistic renderings based on a generalized focal surface to emphasize particular elements in the scene while retaining a realistic look. Our real-time solution provides realistic, as well as plausible expressive results.

1 Introduction

Real cameras have an aperture through which light falls on an image plane containing receptors to register an image. For a sharp image, a small aperture is preferable, but then less light would hit these sensors and diffraction becomes an issue. Using a larger aperture in combination with a lens, 3D points at a certain *focal distance* are projected to a single point on the sensors, while other points map to a *circle of confusion* (COC) [Potmesil and Chakravarty 1981]. This latter effect leads to blur and only within a certain distance range, the *depth of field* (DOF), the image is crisp. DOF is a crucial component for realistic rendering and dramatically improves photorealism and depth perception [Mather 1996]. It also has become an important aspect for semantic purposes by drawing attention to certain elements while maintaining a realistic look.

In this paper, we present an efficient solution to approximate the image capturing process by considering not only aperture, but also aspects of the lens interaction itself. We approximate optical aberrations, which is a unique feature for real-time approaches. Usually these are considered an artifact, but they are crucial for realism and allow us to reproduce many features often employed in artistic lenses (Fig. 1,left). To achieve these effects our algorithm needs a certain generality that is also illustrated by support for specialized configurations, e.g., tilt-shift photography where lens and image-plane no longer align (Fig.1, middle). Our work allows to interactively explore this large variety of possibilities and even outperforms standard competing DOF methods. Our goal is to enable artists and designers to enhance, emphasize and layout a scene or animation using our simulations to better match their intentions. For this direction, efficiency is an important aspect, but we further propose physical and non-physical possibilities to control the various effects intuitively. In particular, we are concerned with focus, as it is the one of the most crucial components in this context. Our interface enables even novice users to produce convincing results (Fig.1, right).

Precisely, the contributions of our paper are as follows:

- An efficient algorithm for DOF and lens effects;
- An interactive and intuitive focus control system;
- A generalized DOF method for *expressive* rendering.

The rest of this paper is structured as follows: We review previous work (Section 2), before discussing our DOF model and rendering algorithm (Section 3). Many optical aberrations come directly from the simulation and we motivate their use (Section 4). We illustrate our method for focus control and extend it to expressive rendering (Section 5). Finally, we discuss and present performance results (Section 6), before concluding (Section 7).

2 Previous Work

Many techniques exist to generate focal imagery in computer graphics, but the results were often of low quality, or far from real-time. The lack of high-quality interactive DOF might be one of the reasons why little work addressed DOF control, despite the increased general awareness that previsualization and control are crucial for productions [Ragan-Kelley et al. 2007]. Instant feedback is even more central for non-physical solutions and, in fact, DOF is well-suited for abstraction: one can guide perception, enhance areas of interest (e.g., person in a crowd), emphasize elements, reduce the general complexity of a scene (making it more understandable), or achieve dramatic appearances when exceeding the physical boundaries. Previously, interactive solutions usually failed to reproduce realistic results, provided only a small range of parameters with limited possibilities, and generalizations lacked plausibility. We obtain a more general almost-accurate simulation, and a physical basis for artistic effects.

Depth-of-Field Rendering Most real-time DOF methods postprocess a single image shot from the center of the lens. Filters are used to approximate COCs at each pixel [Rokita 1996; Riguer et al. 2003; Scheuermann 2004; Bertalmío et al. 2004; Earl Hammon 2007; Zhou et al. 2007; Lee et al. 2009b] leading to high performance, but also artifacts such as *intensity leakage* (foreground leaks in the background). Anisotropic filtering [Bertalmío et al. 2004; Lee et al. 2009b] can only address this issue partially. Alternative scatter methods [Potmesil and Chakravarty 1981] transform pixels into COC sprites, but the necessary back-to-front blending makes it applicable mostly for offline rendering [Demers 2004].

In general, a single image cannot give information about geometry hidden from the lens center which has a large impact on the final image. One way of hallucinating missing information is to split the single image into layers according to depth and extend the colors on each layer into the hidden regions [Barsky et al. 2002; Kass et al. 2006; Kraus and Strengert 2007; Kosloff and Barsky 2007]. However, such extrapolation does not reflect the true scene information and can lead to overly blurred and incorrect results, especially for out-of-focus foreground elements. Further, fusing layers via alpha blending is a coarse approximation. Only for separate objects, such approaches work well. Usually, discretization artifacts arise that can only be mitigated with special image processing [Barsky et al. 2005], information duplication [Kraus and Strengert 2007], or depth variation [Lee et al. 2008].

Multiview accumulation can be used to treat visibility correctly, via ray tracing [Cook et al. 1984] or multi rendering [Haeberli and Akeley 1990]. Since each scene drawing induces a heavy cost, these methods are usually inappropriate for real-time use. Further, the accumulation buffer method [Haeberli and Akeley 1990] forces several constraints on the ray directions, making it difficult to extend the solution to more general lens models. A recent method [Lee et al. 2009a] combines elements of both. A single render step derives a layered representation on which an image-based raytracer is executed. The algorithm is efficient and achieves quality comparable to accurate solutions. However, for our expressive scenario, where high anisotropic COCs can occur, the method shows reduced performance. Many layers are needed to bound the error, increasing memory consumption and making rendering artifacts more common. Our approach works in the same spirit, but is more efficient (even for standard lenses), better adapted to expressive purposes, scales better for smaller amounts of layers, and incorporates advanced lens effects that no other real-time solution provides.

User Control, Semantics and Generalized DOF Creating images from 3D models imposes certain challenges. Instead of directly interacting with the appearance of an object, as is the case for painting, the final result is defined indirectly via rendering parameters. With the increased complexity of physical simulations, there is a need to provide intuitive controls over these parameters in order to ease the realization of particular results, especially for the increasing number of novice users. Further, there is a tendency to extend physical models while maintaining a plausible outcome.

Nowadays, intuitive interaction for lighting design is common and a survey can be found in [Patow and Pueyo 2003]. But other areas have been explored, such as highlights and shadows [Poulin and Fournier 1992; Pellacini et al. 2002], camera placement [Gleicher and Witkin 1992], materials [Pellacini and Lawrence 2007], or indirect illumination [Schoeneman et al. 1993; Obert et al. 2008]. Usually, the physical simulation acts as an entry point for the artist to refine the appearance. Similarly, we allow both; a simple interaction for defining physical and physically-inspired effects.

In the context of DOF and lens effects, little work exists. Kosara [2001] proposed a semantic DOF rendering. The work proves the potential of a controlled DOF, but is a purely 2D process, making results clearly unrealistic. Bousseau [Bousseau 2009] used filtering methods on lightfields to replace the aperture effect for DOF. It abstracts filtering, but not the optical system and offers no local focus control. Kosloff [2007] specifies blurring degrees for 3D points and uses anisotropic diffusion, but the outcome also lacks plausibility. Our approach delivers often-convincing results. We support almost-accurate physical simulations and address dynamic scenes. In particular, we enable a large variety of DOF blur that is crucial for artistic purposes. E.g., we support tilt-shift photography, but avoid costly rendering methods [Barsky and Pasztor 2004] and offer real-time feedback coupled with an intuitive control. Tilt-shift photography allows a misalignment between the image and focal plane. It allows us to focus planar elements not perpendicular to the lens and has, recently, received much attention as it can produce a miniature look via its strong off-focus blur (Fig. 1, middle).

3 Realistic Real-Time Lens Blur

In this section, we explain the model we employ and present our efficient rendering algorithm for DOF and lens effects.

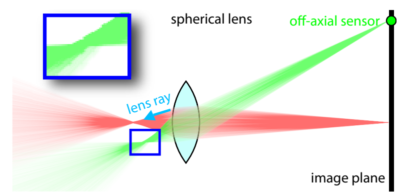

Figure 2: *Simulation of a spherical lens. For most lenses, rays do not converge exactly in a point, especially at off-axial sensors.*

Model The purpose of a lens is to refocus ray bundles on the image plane. Depending on its area of application, the lens' shape is designed accordingly [Smith 2004]. Designers do often rely on path tracing to predict the lens qualities by tracing rays from an image sensor through the lens system into the scene. Our simulation uses the geometric shape and refractive lens properties and captures optical aberrations (Section 4) that have previously been neglected in real-time methods. Further, we do not assume ray coherence in form of a perspective-projection center which many previous approaches needed for efficiency.

Assumptions The most prominent element is the refraction when rays enter and exit the lens according to Snell's law. For our real-time approach, we ignore diffraction effects and assume that rays are solely refracted exactly twice when traversing the lens and travel along straight paths inside the lens (Fig. 2).

3.1 Our Rendering Algorithm

Our algorithm works in two steps. First, we derive an image-based layered representation of the scene using a modified depth-peeling strategy. Second, for each sensor (pixel) we trace several rays. We compute the interaction with the aperture and lens and then use a ray tracing method to find the scene intersections. For the final image, the ray contributions are accumulated.

Layer Construction via Efficient Depth Peeling We avoid testing the lens rays against the actual scene and, instead, derive a layered image-based representation via depth peeling [Everitt 2001] from the lens' center. Depth peeling is a multi-pass technique: each pass *peels* off one layer of the scene. I.e., in the i^{th} pass, each pixel captures the i^{th}-nearest underlying surface by culling all geometry closer than the z-buffer of the previous pass. The termination of the peeling is detected via occlusion queries.

Faster peeling exist [Liu et al. 2006], but to accelerate our ray tracing step, it is more crucial to reduce the amount of layers. We use two important observations. First, those layer pixels that cannot be reached by any lens ray do not need to be extracted. Second, a depth-peeled representation is point-sampled at the pixel centers which leaves room for interpretation of the actual geometry.

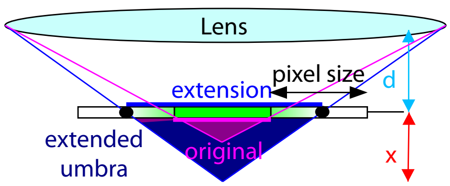

Figure 3: *We can use an offset during depth peeling to omit surfaces hidden behind already extracted pixels. Extended pixels lead to more occlusion by exploiting point-sampling ambiguities.*

Given a pixel P captured by a ray r through a pixel center, P blocks some region in space from all lens rays (Figure 3). If one thinks of the lens as a light source, this space corresponds to the shadow *umbra*. No lens ray can intersect any sample captured by r inside this umbra. Hence, during depth peeling, instead of culling against the depth of the previous pass, we offset the previous depth by the distance r travels inside the umbra. Especially for distant pixels, this solution reduces the amount of extracted layers significantly. Culling can be improved further by virtually extending a captured pixel to the neighboring pixel centers - depth peeling such a scene delivers the same layers. Our result remains artifact free, when assuming silhouette pixels to be extended in this way during our ray tracing. It can be shown that with our umbra method and a standard camera (FOVY=30 deg., dNear=1m, dFar=100m, lens radius = 9mm), 10 layers are always enough (independently of the scene). In practice, 3-7 layers occur. Considering larger connected regions did not result in a performance gain.

Computing Lens Rays Our raytracing starts on a sensor from where many rays are shot. These are blocked according to the aperture and transformed via the lens into a *lens ray* that is then tested against the scene layers. The lens is defined by two height-field surfaces - one for each side and we can combine the intersection test of the aperture and the first lens surface. At each intersection point, the ray is refracted according to the lens' refraction coefficient. The cost of this step is negligible with respect to the remaining algorithm. Alternatively, for algebraic surfaces we can solve the intersections analytically. E.g., spherical lenses are common in reality due to their relatively cheap physical construction.

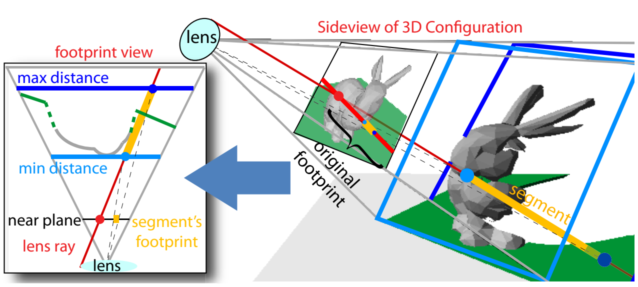

Figure 4: *Instead of searching along the entire ray, we can clamp the ray by min/max depth extents. The process can be repeated for the resulting segment.*

Efficient Intersection Test For the moment, let's assume a single depth layer and a lens ray to test for intersection with this layer. Naively, this involves stepping over all pixels underneath the 2D projection of the ray in the layer's image plane, which we call *footprint*. If the footprint is large, the intersection test is costly. We can reduce it by computing the minimum and maximum depth value of the layer (e.g., via mipmapping). As intersections can only happen within this depth range, we can clamp the original ray to these extents. The resulting 3D segment has a smaller footprint (see Fig.4) and less pixels need consideration. Similarly, given the min/max depth underneath the new footprint, we can repeat the process to further narrow down the search region. After a few iterations, the remaining pixels are tested one by one to find the intersection.

Reducing the search region per ray is costly. Instead, we treat all lens rays in parallel. This implies two challenges, addressed hereafter: Deriving a bounding footprint for all lens rays in a depth interval and computing the min/max values in a footprint region.

Bounding the Footprint For a thin-lens model [Potmesil and Chakravarty 1981], the footprint of all rays for a given depth d is the *circle of confusion* (COC). To bound the footprint for a depth interval $[d_1, d_2]$, it is enough to take the maximum of the COCs at d_1 and d_2. Simple closed-form solutions [Potmesil and Chakravarty 1981] make the computation efficient.

For a geometric lens, apart for particular cases, closed form solutions are complex. Nevertheless, we deal with a finite number of lens rays and each ray can easily be clamped to a given depth range. Thus, to bound the footprint of all rays for a depth interval $[d_1, d_2]$, we intersect each lens ray with planes at distance $d1$ and $d2$. We collect the intersection points and compute a bounding quad in image space that we use as an approximate footprint. Given this bounding quad, we compute the underlying min/max depth values (as detailed hereafter) and repeat the process: we clamp all rays and compute a new bounding quad. Three iterations are a good trade-off between gain and cost of this step. Although it might sound expensive, our ray tracing is data-bound, leaving room for such arithmetic computations. The shown examples evaluate all lens rays, but, in practice, 1/4 th is usually sufficiently accurate.

Computing Min/Max Values Given a fooprint, we use N-buffers [Décoret 2005] to determine the minimum and maximum of the covered values. N-Buffers are a set of textures $\{T_i\}$ of identical resolution. T_0 is the original image and a pixel P in T_i contains the minimum and maximum value of T_0 inside a square of size $2^i \times 2^i$ around P. We cover the footprint rectangle using four texture lookups, corresponding to overlapping squares [Décoret 2005]. The hierarchical N-Buffer construction (T_{i+1} uses T_i) is fast, but N-Buffers are memory intensive. Our solution is to use a mipmap texture and an N-Buffer applied to a downsampled version ($1/8^{th}$ resolution) of the original layer. The memory gain and construction speedup correspond to a factor of eight, while small regions can be sampled efficiently using the mipmap texture or the original image.

There is one catch: Some pixels, especially in later layers, can be empty and do not capture any information and need to be excluded during the min/max N-Buffer construction. To mark missing data, we use the depth value zero: Before depth peeling, we clear the z-buffer to zero and, during the peeling step, we exclude the output depth zero. Consequently, it indicates that no data was output.

Multi-Layer Packing We accelerate multiple layer treatment by packing four depth values into a single RGBA texture directly after the peeling step. It allows us to scan four layers in parallel. These layers share one N-Buffer, where T_0 is set to the per-pixel minimum and maximum of these layers. The intervals are still narrowed down quickly because lens rays are almost perpendicular to the image plane. Finally, we test all four depth values (recovered by a single

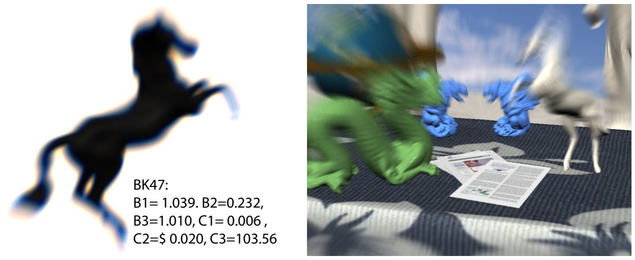

Figure 5: *Examples of lens aberrations - Chromatic aberration (left) for BK7 and Curvature of Field combined with a tilt shift lens focussing on the text on the table (right).*

lookup) simultaneously while stepping along the segment. Once the closest intersection is found, the corresponding color is retrieved.

No layer can be skipped because depth peeling does not order primitives globally. But, it gives a local order in each pixel. Hence, once a pixel is empty, it remains empty for all following layers leading to large empty zones which are detected efficiently by the N-Buffer (zero meaning missing data).

4 Optical Aberrations

Our geometric lens model captures many optical aberrations, which are present in any real camera and particularly visible when the aperture is fully opened. Although, some manufacturers try to counterbalance aberrations by employing lens sets, their simulation is crucial for realistic rendering and artistic effects (some lenses, like LensBaby$^{\text{TM}}$ have exaggerated aberrations to provoke a certain appearance). We will review three cases (for more examples, we refer to [Smith 2004]) that we consider of interest due to their strong effect and relatively common usage in artistic shots.

Spherical Aberration In contrast to a theoretical thin lens, spherical lenses do not perfectly focus all sensor rays in a single focal point. Usually, rays are more strongly bend on the border than the lens' interior (Prentice's Rule). Biconvex and Aspheric lenses (like the cornea in our eye) can reduce the effect. Visually, spherical aberration manifests in a general blur and discrepancy of sharpness and brightness of the image's center in comparison to the boundary. This allows us to derive a softer appearance and make the observer focus on central elements. Further, halos appear around strong highlights, visible for Bokeh, can be used to drive attention, as well as to define a general mood (Fig. 5).

Curvature of Field Curvature of field projects a focal plane to a curved (nonplanar) image. Rays at a large angle see the lens as if it had a smaller diameter but higher power. The image of the off axis points moves closer to the lens. This type of curved lens surfaces is very common in many real lenses, especially telescopes. The effect is that images are clearly focused in the center, but lose focus towards the boundary. Compared to spherical aberration, the blur is more anisotropic and is often used to suggest the past, dreams, or (for stills) velocity (Fig. 5).

Chromatic Aberration The refraction index of a lens usually depends on the wavelength of the incoming light. Often invisible, it can result in colored halos around objects (Fig. 5).

We use an empirical equation proposed by Sellmeier [1871]:

$$n(\lambda) = \sqrt{1 + \frac{B_1\lambda^2}{\lambda^2 - C_1} + \frac{B_2\lambda^2}{\lambda^2 - C_2} + \frac{B_3\lambda^2}{\lambda^2 - C_3}}, \tag{1}$$

where B_i, C_i are material coefficients, λ the wavelength, and n the refractive index. The benefit of this model, originally for a thin lens, is that we can benefit from large material data bases. Physical (e.g., Borosilicate crown glass (BK7) - a typical lens material) or non-physical results are possible (Fig. 5). For real-time performance, we compute the RGB color channels separately, assuming the wavelengths 650 (R), 510 (G), and 475 nm (B).

5 Focus Control

This section presents our algorithm to intuitively control lens blur. We present algorithms for physically-based lenses, but also extend DOF beyond the physical definition by allowing varying lens parameters for each image point. In this context, we also allow control over the previously presented aberrations for abstraction purposes. The values are temporally adapting to a sparse user input.

User Interface As previously mentioned, controlled focus allows us to guide an observer to certain locations, emphasize objects, or create a special mood. For example, it might be of interest to always keep an object defocused in order to not reveal any details, while other elements of the scene stay constantly in focus. Changing the focal distance manually for each frame is a tedious process. In our interface, a user controls DOF at a high level by attaching attributes, like focused or defocused, directly to the scene and by keyframing them over time. A click on the screen defines a *focus point*. Internally, we store its barycentric coordinates and triangle to support animated geometry. For each focus point one can specify DOF parameters (most prominently the amount of blur) and influence weights. Based on this input, the camera parameters are optimized to reflect the intended definitions for the current view. Per default, we exclude constraints outside the view frustum, but allow an artist to specify otherwise. We also increase the influence of nearer constraints to avoid ambiguities and use temporal interpolation to achieve temporal coherence.

5.1 Defining Focus for Standard Lens Models

We will first describe how to optimize several common lenses, before addressing non-photorealistic rendering.

Thin-Lens Model Given the focal length F, the focal distance d is defined by the distance between image plane and lens u via: $u = Fd/(d-F)$ for $d > F$. A single defined focal distance directly defines u a real camera. A weighted least-square fit is used for several constraints.

Spherical Lens Spherical lenses can be processed similarly, by computing the focal length via the lensmaker's equation: $\frac{1}{F} \approx (n-1)\left(\frac{1}{R_1} - \frac{1}{R_2} + \frac{n-1d}{nR_1R_2}\right)$, where R_1 and R_2 are the lens radii, d the thickness, and n the refractive index.

Tilt-Shift Photography For *Tilt-shift photography* the camera's image plane is tilted with respect to the lens, hereby tilting the focal plane. The effects are very interesting (Fig.1), but the nonintuitive relation between tilt and focus can make it difficult to operate the device, especially for animated scenes. On the contrary, we derive a least-square focal plane from the focus points. The focal plane is then automatically transformed into an image plane tilt [Merklinger 1996], making the process simple to control. Care is needed when the focal plane aligns with the view vector. We avoid this physical impossibility by limiting the plane normal.

5.2 Expressive Depth of Field Control

Our system also allows for non-physically-based local parameter definitions, while standard lens models are restricted to global definitions. For artistic purposes, this is particularly interesting to enhance certain regions. For example, locality is important when different emphasis is to be put on objects residing at the same distance. The depth-of-field itself is controlled via a focal surface that continuously interpolates the focal-depth constraints in screen space, as well as the depth-of-field extent in form of a single size value. Temporal continuity is achieved again by smoothing the surface and depth of field over time.

Focal Surface and DOF Interpolation The focal surface definition is based on a moving-least squares (MLS) solution. For this, around each focal point kernel functions are defined. In practice we use weights of $\alpha 1/d^{\beta}$, where d is defined in terms of screen space distance for each focal point. For a given pixel we minimize the error functions by weighting the desired parameters at focus points according to these weight functions. The parameters α, β give control over the strength and extent of the influence region of each focus point. The importance weight, $alpha$, is used for finer control, since the best-fit surface is not necessarily identical to the designer's intention. β controls the range affected by the nearby control points. As β increases, local behavior becomes narrow. In the limit, discontinuities could be reintroduced, but in practice values of $\beta = 2$–4 work well. We experimented with different kernels and got comparable results. Nevertheless, we found that it is important to work with singularities at $d = 0$ to ensure a perfect interpolation of the control point itself. The surface evaluation is executed entirely on the GPU and allows us to define the parameters in in each pixel.

Our system also supports focus points that indicate out-of-focus regions. The user simply specifies the offset with respect to a potential focal plane. In such a way we can achieve a controlled defocus. In practice, this performs very intuitively, but problems occur when such constraints overlap. In order to avoid this effect, we reduce the influence of the corresponding kernel functions according to the distance and slowly fade out their contribution, when they become invisible. As before, this can be counteracted by the artist and because our system delivers efficient feedback, it is possible to keyframe the behavior differently and have immediate feedback. In order to avoid always ensure a smooth behavior, all elements are smoothly interpolated over time.

5.3 Expressive Lens Effects

We further add the effects of aberrations in a non-physical way with simple parameters. Originally the lens shape would influence these effects, but in practice, this process would involve much expertise and is far from an intuitive framework. Instead, we decided to simulate the aberrations with more intuitive parameters.

Spherical aberration is caused by varying distances that light travels inside of the lens. Often we observe that rays on the periphery of the lens are bend more strongly than those in the center. In other words, we can approximate this effect by increasing the refractive power of the lens which we allow via a simple spline definition. In practice, this effect is most valuable for Bokeh effects and we further provide the possibility to directly assign weights to the rays in order to achieve a certain shape of Bokeh.

Curvature of field is an effect, that arises because the focal plane is associated to a curved image plane. In other words, if the sensors were on this image plane, the resulting rendering of an object on the focal plane would be sharp. To simulate this effect, we can let the user define an offset surface for the image plane. Again, we use a smooth MLS interpolation and write the resulting deformation in a buffer. As all points on this surface share the same focal distance, it implies that the lens appears to have a different focal length. We adapt rays traversing this surface accordingly to take this effect into account.

Finally, chromatic aberration can be controlled directly via different refraction coefficients for the different color channels. Hence, it remains basically unchanged.

We found that these definitions allow a very intuitive interaction with lens effects and the video demonstrates how easily parameters can be adjusted to achieve complex appearances.

6 Results and Discussion

Our test system is a Pentium Core2Quad 2.83GHz with a GeForce 285 GTX graphics card. We evaluated the performance of our system with various test scenes (Table **??**). We compared our results to a recent state-of-the-art algorithm [Lee et al. 2009a] with two different settings. In these scenes, the method our competitor needed 16 layers to be artifact free. Nevertheless, we added timings for 8 layers to achieve a fairer comparison. In the latter case, artifacts were readily visible. Our method produced artifact-free images using only 4 layers. The quality is similar to reference renderings which we show by reporting signal-to-noise ratios (PSNR) and structural similarity (SSIM) [Wang et al. 2004]. Please realize that the scenes are of high complexity (the smallest example has 98K triangles, the largest 935K). In all cases, our method resulted in real-time performance between 100 and 30 Hz.

	Pre	RT	our	comp.8/16	ref.	error
Town (98K)	4	6	10	15.3(1.5)/26(2.6)	125(12.5)	
Angels (407K)	7	17	24	75.2(3.1)/122(5.1)	213 (8.9)	36.94db/0.97
Table (935K)	16	15	31	81.3(2.6)/125(4.0)	641(20.7)	34.97db/0.95

Table 1: *Comparison (100 lens rays, 800x600): We give timings in ms for reprocessing (Pre) -NBuffer + DepthPeeling-, raytracing (RT), and total pipeline of our algorithm (our) using 4 layers. Further, we indicate timings of a competitor [Lee et al. 09] (comp.), using 8 and 16 layers (with acceleration factors given in parentheses). Finally, we give a quality measure (Error) with respect to a reference rendering by evaluating PSNR and SSIM [Wang et al. 2004].*

For us, four layers, is usually enough for a reference-like result. This is an interesting feature and not valid for uniform decompositions [Lee et al. 2009a]. Our depth peeling can be slower than a single-pass decomposition [Lee et al. 2009a], but our rendering method is more cache efficient, treats multiple layer simultaneously, uses less and more-predictable arithmetic operations. In consequence, we achieve equivalent or better quality with a strong speedup. We also do not miss hidden fragments within layers and avoid temporal popping for geometry crossing many layers. Our memory consumption is generally lower than in [Lee et al. 2009a] because, in practice, our modified peeling resulted at most in seven layers (less than half of standard depth peeling) for realistic scenes. The use of N-Buffers for skipping *and* intersection tests gives sublinear performance due to the sparsity of higher layers. For 8 layers, the second four only cost $50 - 25\%$.

To illustrate the expressive spectrum of our system, we mimicked specialized lenses like LensBaby™ or Spiratone Portragon. The effectiveness of our interface is best demonstrated in the accompanying video. In particular, the instant feedback is crucial to judge where supplementary constraints are needed. Dramatic effects can be achieved in a few clicks and the results look convincing, even under animation.

7 Conclusion and Future Work

We presented a novel real-time lens-blur rendering system exceeding previous methods in performance and quality. We introduced the first real-time system that manages many of the lens aberration effects. The latter are an important component for artistic photography and, consequently, we presented a simple system to control depth-of-field blur. We further extended this control to non-physical results, that appear plausible, yet give a larger variety of possibilities to designers. We illustrated the flexibility of our system on several complex examples.

In the future, we would like to extend our user interface to further facilitate the interaction. One direction is to adopt painting metaphors and we would like to investigate new possibilities of temporal interpolation. In theory this is already possible, but we would like to integrate an event-driven control. Such a system could be useful in a game to trigger focus elements for particular objects and integrate it into a game engine to test how well users can be guided via our focus design.

References

BARSKY, B. A., AND PASZTOR, E. 2004. Rendering skewed plane of sharp focus and associated depth of field. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, ACM, New York, NY, USA, 92.

BARSKY, B., BARGTEIL, A., GARCIA, D., AND KLEIN, S. 2002. Introducing vision-realistic rendering. In *Proc. Eurographics Rendering Workshop*, 26–28.

BARSKY, B., TOBIAS, M., CHU, D., AND HORN, D. 2005. Elimination of artifacts due to occlusion and discretization problems in image space blurring techniques. *Graphical Models 67*, 584–599.

BERTALMÍO, M., FORT, P., AND SÁNCHEZ-CRESPO, D. 2004. Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards. In *Proc. 3D Data Processing, Visualization, and Transmission*, 767–773.

BOUSSEAU, A., 2009. Non-linear aperture for stylized depth of field. SIGGRAPH 2009 - Technical talk.

COOK, R. L., PORTER, T., AND CARPENTER, L. 1984. Distributed ray tracing. *Computer Graphics 18*, 3, 137–145.

DÉCORET, X. 2005. N-buffers for efficient depth map query. In *Proc. Eurographics*, 393–400.

DEMERS, J. 2004. Depth of field: A survey of techniques. In *GPU Gems: Programming Techniques, Tips & Tricks for Real-Time Graphics*, R. Fernando, Ed. Addison-Wesley Professional, ch. 23, 375–390.

EARL HAMMON, J. 2007. Practical post-process depth of field. In *GPU Gems 3*, H. Nguyen, Ed. Addison-Wesley, ch. 28, 583–606.

EVERITT, C. 2001. Interactive order-independent transparency. *White paper, nVIDIA 2*, 6, 7.

GLEICHER, M., AND WITKIN, A. P. 1992. Through-the-lens camera control. In *SIGGRAPH*, 331–340.

HAEBERLI, P., AND AKELEY, K. 1990. The accumulation buffer: Hardware support for high-quality rendering. *Proc. ACM SIGGRAPH*, 309–318.

KASS, M., LEFOHN, A., AND OWENS, J. 2006. Interactive depth of field using simulated diffusion on a GPU. Tech. rep., Pixar Animation Studios.

KOSARA, R., MIKSCH, S., AND HAUSER, H. 2001. Semantic depth of field. In *Proc. IEEE Symp. Infomation Visualization*, 97–104.

KOSLOFF, T., AND BARSKY, B. 2007. An algorithm for rendering generalized depth of field effects based on simulated heat diffusion. In *Proc. Computational Science and Its Applications*, 1124–1140.

KRAUS, M., AND STRENGERT, M. 2007. Depth-of-field rendering by pyramidal image processing. In *Proc. Eurographics*, 645–654.

LEE, S., KIM, G. J., AND CHOI, S. 2008. Real-time depth-of-field rendering using splatting on per-pixel layers. *Computer Graphics Forum 27*, 7, 1955–1962.

LEE, S., EISEMANN, E., AND SEIDEL, H.-P. 2009. Depth-of-Field Rendering with Multiview Synthesis. *ACM Trans. Graphics 28*, 5, 134:1–6.

LEE, S., KIM, G. J., AND CHOI, S. 2009. Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. *IEEE Trans. Visualization and Computer Graphics 15*, 3, 453–464.

LIU, B., WEI, L.-Y., AND XU, Y.-Q. 2006. Multi-layer depth peeling via fragment sort. Tech. Rep. MSR-TR-2006-81, Microsoft Research, June.

MATHER, G. 1996. Image blur as a pictorial depth cue. *Biological Sciences 263*, 1367, 169–172.

MERKLINGER, H. M. 1996. *FOCUSING the VIEW CAMERA*. Nova Scotia.

OBERT, J., KRIVÁNEK, J., PELLACINI, F., SÝKORA, D., AND PATTANAIK, S. N. 2008. icheat: A representation for artistic control of indirect cinematic lighting. *Comput. Graph. Forum 27*, 4, 1217–1223.

PATOW, G., AND PUEYO, X. 2003. A survey of inverse rendering problems. *Computer Graphics Forum 22*, 4 (Dec.), 663–687.

PELLACINI, F., AND LAWRENCE, J. 2007. Appwand: editing measured materials using appearance-driven optimization. *ACM Trans. Graph. 26*, 3, 54.

PELLACINI, F., TOLE, P., AND GREENBERG, D. P. 2002. A user interface for interactive cinematic shadow design. In *SIGGRAPH*, 563–566.

POTMESIL, M., AND CHAKRAVARTY, I. 1981. A lens and aperture camera model for synthetic image generation. *Proc. ACM SIGGRAPH 15*, 3, 297–305.

POULIN, P., AND FOURNIER, A. 1992. Lights from highlights and shadows. In *SI3D*, 31–38.

RAGAN-KELLEY, J., KILPATRICK, C., SMITH, B. W., EPPS, D., GREEN, P., HERY, C., AND DURAND, F. 2007. The lightspeed automatic interactive lighting preview system. *ACM Trans. Graph. 26*, 3, 25.

RIGUER, G., TATARCHUK, N., AND ISIDORO, J. 2003. Real-time depth of field simulation. In *ShaderX^2: Shader Programming Tips and Tricks with DirectX 9*, W. F. Engel, Ed. Wordware, ch. 4, 529–556.

ROKITA, P. 1996. Generating depth-of-field effects in virtual reality applications. *IEEE Computer Graphics and its Application 16*, 2, 18–21.

SCHEUERMANN, T. 2004. Advanced depth of field. In *Game Developers Conference*.

SCHOENEMAN, C., DORSEY, J., SMITS, B., ARVO, J., AND GREENBERG, D. 1993. Painting with light. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 143–146.

SELLMEIER, W. 1871. Zur Erklärung der abnormen Farbenfolge im Spectrum einiger Substanzen. *Annalen der Physik und Chemie 219*, 272–282.

SMITH, W. 2004. *Modern Lens Design*. McGraw-Hill Professional.

WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Processing 13*, 4, 600–612.

ZHOU, T., CHEN, J., AND PULLEN, M. 2007. Accurate depth of field simulation in real time. *Computer Graphics Forum 26*, 1.