Physically Based Lens Flare Rendering in “The Lego Movie 2”

Erik Pekkarinen
Animal Logic
erikp@animallogic.ca

Michael Balzer
Animal Logic
michaelba@animallogic.ca

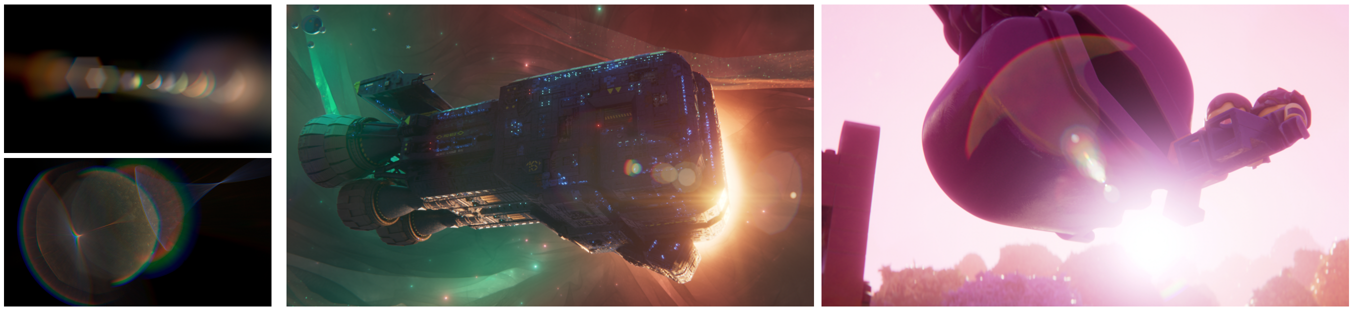

Figure 1: Standalone lens flares demonstrating occlusion and lens dirt effects generated by our approach (left); lens flares in “The Lego Movie 2: The Second Part” © 2019 Warner Animation Group (center and right).

ABSTRACT

We present our approach for incorporating realistic lens flare rendering in a production renderer based on a previously presented physically based lens simulation technique [Hullin et al. 2012]. We describe the approximations and sampling techniques behind efficient lens flare rendering, in addition to introducing flexible artist controls and workflows for this purpose. Using “The Lego Movie 2: The Second Part” as a case study, we show that these approaches are efficient and work well in a production environment.

CCS CONCEPTS

• **Computing methodologies → Lens simulation**.

KEYWORDS

camera, lens flare, rendering, artist workflow

ACM Reference Format:
Erik Pekkarinen and Michael Balzer. 2019. Physically Based Lens Flare Rendering in “The Lego Movie 2”. In *DigiPro ’19: The Digital Production Symposium (DigiPro ’19), July 27, 2019, Los Angeles, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3329715.3338881

1 INTRODUCTION

Lens flares are image artifacts generated by light being reflected within a camera’s lens system before hitting the image sensor. Since all physical lens systems generate flares to some extent, their presence is a norm in films, and consequently is expected in realistic computer generated imagery. They are even introduced artificially, for example, to give shots a more interesting or dramatic look. Traditionally, phenomenological approaches are used to synthesize lens flares, but more recent techniques are physically based and model light interaction with individual lens system elements [Hullin et al. 2011].

The technique by [Hullin et al. 2012] models lens interactions using polynomial systems obtained as Taylor expansions of the analytic ray-surface intersections at the optical axis. Here, individual polynomials depend on the physical properties of the lens elements, such as their dimensions and wavelength dependent IORs, and get combined into systems based on the actual lens arrangements inside a camera. Using this to transform rays is more efficient than naïve ray tracing approaches, especially since multiple systems can be composed and then truncated to the desired degree. This makes the evaluation cost per light path independent of the number of lens elements while keeping the approximation error roughly constant.

The efficiency of this approach combined with uncompromised physical accuracy makes it ideal for production rendering, where the real-time rendering constraints of many alternative approaches do not apply and a photorealistic result is the primary goal. A C++ implementation of this technique by its authors is available as the *Polynomial Optics Toolkit.* The presented lens flare renderer is built around this library.

2 LENS FLARE LIGHT PATHS

Lens flares are generated by light paths involving reflections. Only light paths with an even number of reflections can travel from the front lens to the image sensor. The Fresnel equations tell us that typically less than 1 % of transmitted radiance is reflected at any lens surface. This results in the radiance of flares going down rapidly with their reflection count, and thus considering light paths with only two reflections is a reasonable approximation. These reflections are obtained as the 2-combinations of all the material interfaces. For example, in a lens system with 29 material interfaces this results in 406 light paths.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
DigiPro ’19, July 27, 2019, Los Angeles, CA, USA
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6799-8/19/07.
https://doi.org/10.1145/3329715.3338881

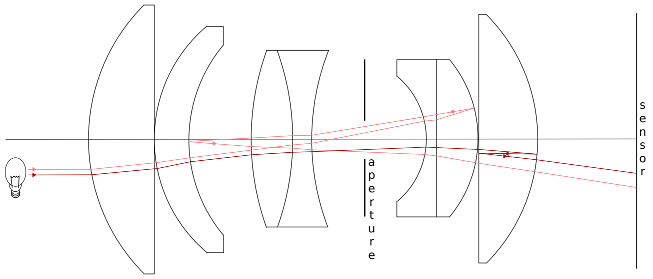

Figure 2: Two rays from different lens flare generating light paths.

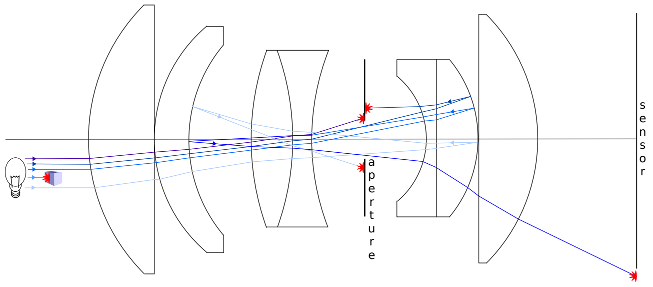

Figure 3: Rays from the same light path that are occluded by scene geometry, the camera housing and the aperture.

Figure 2 shows example rays from different light paths with two reflections. Each of these light paths consists of three transmission segments. Given the systems for the individual lens elements from the Polynomial Optics Toolkit, we combine them to get the transformations for each segment. These systems are then combined to obtain the transformations for the complete light paths.

Due to occlusion, not all rays from lens flare generating light paths reach the image sensor. As illustrated in Figure 3, rays originating from a light source may be occluded by geometry outside of the lens system, by the camera housing or by the aperture. We handle the first case by raycasting against the scene geometry, while the other two cases are handled by testing the ray positions against the image sensor and the aperture geometry. For aperture occlusion testing, we evaluate up to three additional systems per light path, one for each transmission segment passing the aperture.

We use the Fresnel equations to estimate the radiance carried by a light path. In addition to the material dependent part, we include the angle dependent Fresnel factor. Since the contributing rays are nearly parallel to the optical axis after passing the front lens, and computing the angle dependent factor at every interface is expensive, we estimate this with the Schlick approximation [Schlick 1994] at the front lens.

3 SAMPLING

We sample lens flares by connecting a random sample on a light source with a random sample on the front lens. For this given entry point and direction to the lens system, the evaluation of the light path transformation provides us the exit point and direction at the back lens. We then use this to test against the image sensor geometry, and determine the splat location of the sample in the image.

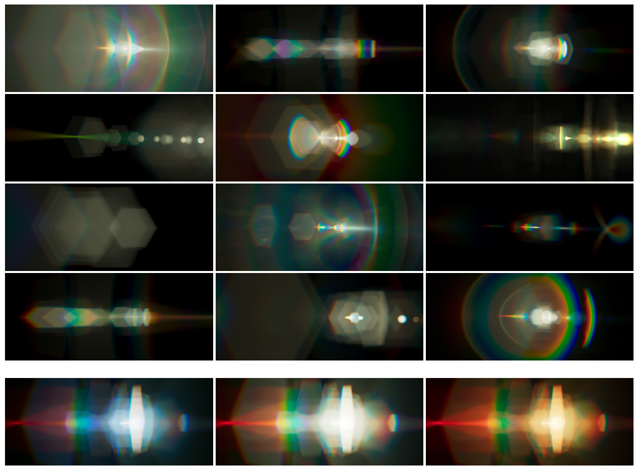

Figure 4: Different lens systems supporting anamorphic, wide-angle and multi-zoom setups (top), and different coating presets (bottom).

To improve convergence, we try to maximize the number of samples arriving at the image sensor. We observed that the distribution of these unoccluded samples is very non-uniform across both light paths and the front lens area. Because of this, we sample them both adaptively. While the light paths already have a natural partioning, the front lens is partitioned uniformly into cells. The overall sampling probability then becomes $p_s = p_{\text{lp}} \times p_{\text{cell}}$, where p_{lp} is the light path sampling probability and p_{cell} is the front lens cell sampling probability. The normalization weight of a sample that arrives at the image sensor is

$$w_s = \frac{1/p_s}{\sum_{i=1}^{N} 1/p_i}, \tag{1}$$

where N is the total number of samples hitting the sensor. In practice we found that the best convergence is obtained when the probabilities p_s are defined as the fractions of radiance carried by each subset of rays. The adaptive sampling increases the sensor hit rate for the standalone lens flare renders in Figure 1 from about 15 % to about 90 %.

To splat the unoccluded samples on the image sensor, we estimate their radiance L_s as

$$L_s = L(\lambda) \times c_{\text{lens}} \times l \times F_{\text{lp}} \times w_s, \tag{2}$$

where $L(\lambda)$ is the radiance incoming from the light at wavelength λ, c_{lens} is the lens attenuation factor, l is the Lambertian reflectance coefficient of the ray at the image sensor and F_{lp} is the light path Fresnel factor. The lens attenuates rays as $c_{\text{lens}} = c_{\text{dirt}} \times c_{\text{coating}}$, where c_{dirt} is the front lens dirt color and c_{coating} is the product of the light path lens coating colors.

4 ARTIST WORKFLOW

The lens flare renderer is part of Animal Logic's proprietary render engine. It is used in a separate render pass for selected shots and lights with prominent flares. The input scene is the same as that used for the main render pass. Additionally, lights generating lens flares are flagged. These lights can also have separate intensity multipliers for the lens flare pass.

To support a wide variety of lens systems, the renderer parses a lens system description along with a coating color description and an attenuating front lens dirt map as properties of the render camera. The used lens system descriptions are based on real-world lens descriptions. Existing render camera properties like f-stop, aperture blade count and focal length affect the lens system in a physically correct way. Figure 4 illustrates the wide range of looks that can be achieved with these input parameters.

If the main render pass uses a separate lens distortion model, it is taken into account by applying the distortion on the light samples in image space. This way the flares correctly line up with their generating light source when composited with the main render.

The renderer supports the output of individual flares as independent layers to allow their manipulation in compositing. To reduce the number of lens flare layers, it merges all flares below a given radiance threshold into a single layer.

A typical 720p lens flare preview render takes less than 10 seconds, whereas a typical 2K production quality render takes less than 15 minutes. These render times were feasible in production, as they were fractions of the main render pass times.

5 CONCLUSIONS

Our lens flare renderer was used for about 50 shots in "The Lego Movie 2: The Second Part". This represents the majority of the film's strongly backlit shots. The renderer was easily adopted by artists since it largely uses existing camera parameters and scene setups. There are strong expressions of interest for its usage in Animal Logic's future animation and VFX projects.

One limitation is that it cannot account for stochastic scattering processes and diffraction since these are not captured by the underlying Polynomial Optics Toolkit.

In the future, we would like to determine the correct sensor normalization with respect to the light intensity to make the input parameter units physical. We also intend to allow for denoising of flares by correctly capturing their sampling variance, and reduce variance in the first place by adaptively sampling multiple light sources as well as their occlusion. We also plan to add camera hoods and procedurally animated aperture shapes.

ACKNOWLEDGMENTS

Thanks go to Daniel Heckenberg, Rafiq Dandoo, Laurianne Proud'hon as well as the lighting crew and supervisors for their support and invaluable feedback.

REFERENCES

Matthias Hullin, Elmar Eisemann, Hans-Peter Seidel, and Sungkil Lee. 2011. Physically-based Real-time Lens Flare Rendering. *ACM Trans. Graph.* 30, 4, Article 108 (July 2011), 10 pages. https://doi.org/10.1145/2010324.1965003

Matthias B. Hullin, Johannes Hanika, and Wolfgang Heidrich. 2012. Polynomial Optics: A Construction Kit for Efficient Ray-Tracing of Lens Systems. *Computer Graphics Forum (Proceedings of EGSR 2012)* 31, 4 (July 2012).

Christophe Schlick. 1994. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum* 13, 3 (1994), 233–246. https://doi.org/10.1111/1467-8659.1330233