# Depth of Field Simulation in Computer Graphics

(Source: self made)

**Ibraheem Alhashim**
Department of Computer Science
Portland State University
Portland, Oregon
ibraheem@pdx.edu

**ABSTRACT**: A major part of the amount of realism in a rendering engine is the engine's camera model. Simulating real camera models can produce desired realistic effects such as depth of field (DOF). Several methods are currently used to simulate DOF but they are either computationally expensive methods that produce realistic results or inaccurate but fast methods. This paper includes a list of reasons behind simulating DOF, the methods used for generating the effect, and a survey of fast methods that can be used within real time applications.

## 1. INTRODUCTION

Research in the field of computer graphics has always been focused on achieving the maximum level of realism. The realism of modeling a real life object is limited by the artists capabilities where as the quality of the images produced is determined by the rendering engine. For a rendering engine to achieve heigh levels of realism it should incorporate not only realistic models of lightning and shading but also simulate special effects such as DOF blurring.

DOF is defined as "the range of distances, in front of and behind the point of focus, where the eye perceives the image to be sharp" [1]. The effect is noticeable in human vision as well as in cameras. Blurring occurs outside the point of focus as a result of a finite DOF. The next section of this paper discusses the importance of adding DOF as part of the rendering process.

## 2. THE NEED FOR DEPTH-OF-FIELD

The phenomena of DOF blurring gives a computer generated scene a more natural and realistic appearance. Objects in computer generated (CG) scenes are commonly rendered to all look as if they are in focus which results in an unnatural image. Adding DOF as part of the rendering

process adds an extra step towards realism. DOF also adds depth cue information for the viewer to help determine distances between the scene's objects [2]. Another area where including a depth of field effect is important is with physically-based rendering algorithms. These systems require the comparison of generated images with real life data to be accurate. Such systems help with studying the transport of light rays and lens behaviors [3].

The capability of adding a DOF effect into a CG cinematic scene allows cinematographers to achieve artistic images. The effect is essential in producing CG images that try to achieve the look and feel of film [6]. Also, applications such as special effects and augmented reality requires accurate blending of real life images with CG scenes. In film and television scenes may require the addition of CG objects or characters onto a real life video. Since cameras are used to acquire such clips, the images are limited by a finite DOF depending on the physical lens in use. In order to blend the two parts, the CG part should simulate the DOF effect appearing in the real part. The same blending techniques can be used to achieve an increased level of realism in augmented reality where images include both virtual reality and real-world live video elements.

In video games, a graphics engine designer spends a large amount of effort working on simulating real life effects in their product. The more the game's graphics simulate such effects the better it appeals to the customers. Current video games graphics engine list the ability to simulate DOF as part of the the engine's main features. An example of such engine is CryEngine2 developed by Crytek and was released in 2007.

Perhaps an unconventional use of DOF blurring is in graphical user interfaces and information visualization. Blurring cretin elements on the user's screen can have the desired effect of leading the user's vision towards important elements. Kosara et al. presented a new technique called Semantic Depth of Field [5]. Their method assigns different focus values based on features instead of distance. This approach allows the user to pick parts in a visualization of data to be in focus and blur out irrelevant data. Figure (1) shows an example of the technique. Current similar implementations of this idea can be seen in a modern operating system's GUI such as Apple Mac OS X, Microsoft Windows Vista, Compiz Fusion for the X Window system, and Sun's looking glass project.

## 3. SIMULATING DOF

Common 3D rendering techniques use the pinhole camera model. In this model all objects in the scene are in focus regardless of their distance from the viewing point [4]. Rendering all objects in the scene as being in focus and sharp negatively impacts the sense of realism in the viewer's perception.

To overcome the limitation of the pinhole camera model, computer graphics researchers worked on simulating the thin lens system [2][4][6]. In this model we specify a focal length $f$, for the lens, in which given a distance $v$ between the lens and the image plane, and a distance u between

an object and the center of the lens and we can get the thin lens equation [4]:

$$\frac{1}{u}+\frac{1}{v}=\frac{1}{f}$$

If this equation is satisfied, objects at distance $u$ are rendered in focus. Light rays reflected from objects closer or beyond $u$ intersect the image plane as circles. These circles are called Circles of Confusion (CoC). The diameter of these circles on the image plane is computed by the formula:

$$CoC=\left|D\cdot f\,\frac{(u-z)}{z\cdot(u-f)}\right|$$

Where $D$ represents the diameter of the lens, and $z$ is the total distance between the object and the view point. Figure (2) shows the idea behind the thin lens model. Simulating the rays projected to the image plane as CoC would result in the blurred parts of the image, thus representing the DOF blurring.

There are a number of algorithms used in rendering engines that have been developed to generate DOF effects. Some algorithms have the advantage of being fast and hence are more suitable for real time systems. Other algorithms work toward achieving an accurate simulation of DOF blurring. These are used in situations where the quality of the generated image is more important than the time it took to render.

One type of the methods developed to generate depth of field effects uses the idea of distributed ray tracing [4]. In this method, a number of camera rays are traced through each pixel on the image plane from a slightly different reference point. The final color of a single pixel would be the average of the intensity of the distributed rays. The method produces the most realistic DOF results since it models light rays transport properties. However, due to the amount of ray tracing involved the method is not used in real-time systems.

Another approach is the layered depth method [4]. In this method the 3D scene is divided into several layers based on which pixels share the same distance to the camera. These layers are then rendered separately and blurred to generate the DOF effect and the final image would be the combination of these layers. The method may result in artifacts especially with scenes containing objects that stretch to several layers. Such object may experience discontinuity at the edges of theses layers.

DOF can also be generated by rendering the 3D scene multiple times (image accumulation) [2] [4]. In this method the scenes are rendered using the common pin-hole camera model. In each rending pass the plane in focus remains the same and the center of projection gets slightly translated. To produce the final image, these layers are appropriately combined together. Similar to the distributed ray tracing method, this method generate accurate and realistic DOF effects.

The method is computationally expensive since the scene is rendered multiple times only to achieve the DOF effect.

The most widely used method today for fast DOF rendering is post-process filtering. Postemsil et al. pioneered the idea of generating DOF effects using post-processing [2]. The idea behind the method is to render the image using the pin-hole camera model and then applying filters to blur out parts of the image. During the rendering process a $z$ value (depth) is calculated for each pixel. All sample points would then represent CoC with a diameter determined by the CoC equation and by using the calculated $z$ values for each pixel. Intensities of the pixels in the final image are calculated using the weighted average of the CoCs that intersect with the pixel. These methods have the advantage of being independent of the scene's complexity [4].

However, due to the amount of calculations required, post-process filtering methods tend to be slow [2]. Also, filtering techniques usually result in intensity leakage problems. A leakage problem occur when parts of the blurred foreground or background objects leak outside their area into the objects that are in focus. Solutions to this problem are usually computationally expensive. Another problem with post-processing is when information about the visibility of an object across a limited lens aperture is not available.

Research efforts towards applying DOF effects in real-time inspect methods that implement the idea of post-processing with approximations of the filters applied. The following section surveys a number of recently proposed algorithms that maybe used with real-time systems.

## 4. FAST DEPTH OF FIELD ALGORITHMS

Applications including virtual reality, video games, and interactive visualization require that DOF effects get generated in real-time. This task requires that pixel intensities gets calculated in speeds that reach 30 frames per second or more. Current fast DOF algorithms usually suffer form intensity leakage that can be distracting to the viewer. The goal of achieving more accurate DOF outputs require large amounts of calculations that can be accelerated using modern graphic processing hardware.

Bertalmio, Fort, and Crespo introduced an original anisotropic diffusion Partial Differential Equation (PDE) that can take advantage of modern graphics card capabilities [1]. Their method uses a PDE to perform accurate depth-dependent diffusion on a rendered image from a 3D scene rendered using the standard pin-hole camera model. The proposed equation uses a PDE to perform diffusion:

$$\frac{\partial I}{\partial t}(x, y, t) = \nabla \cdot (g(x, y, t) \nabla I(x, y, t))$$

The implementation of their equation is strongly related to the characteristics of the Cg

programming language. Operations written in Cg are performed entirely in the Graphics Processing Unit (GPU). Most of today's graphics cards are equipped with a powerful GPU that is designed to perform a large number of graphical calculations in real time. Taking advantage of hardware processing units allow for such algorithms to be used in real-time systems. In this algorithm each pixel requires similar processing and, therefore, it would take advantage of a parallel hardware architecture. Another advantage of this implementation is in the fact that all the data required for generating the DOF effect can be stored in GPU memory. This reduces the amount of bus traffic needed and making it possible to run the entire algorithm in the GPU. The paper describes the bottleneck of the algorithm as being the texture-lookup operations of the 2D image. With this limitation, however, the algorithm is still much faster than direct computation of Gaussian blurring [1]. The performance of their algorithm reached 4.9 frames per second for 20 iterations with 1K screen resolution using a NVIDIA GeForce FX 5950 Ultra graphics card (introduced in 2003). Figure (3) shows the stages of calculating the DOF effect.

Zhou, Chen and Pullen presented another method that also uses the power of the GPU to simulate DOF in real-time [4]. Their algorithm uses a two-pass filter similar to using a Gaussian filter. The first pass applies a vertical filter to each pixel while sampling and weighting neighboring pixels. The result of this first pass is then passed as the input to the second pass. The second pass perform horizontal sampling. The results of both passes are then blended and renormalized to produce the final image. The advantage of having a two pass filter is the ability to sample more pixels than a one pass filter.

The weights applied to each pixel in the filter is decided by three functions. The first function is the overlap function. This function finds the relation between CoCs and the center of a pixel and if the two overlap that pixel would be included in the calculations. The second function that affects the filter weight is the light intensity function. The method uses a simple uniform spread of intensity across the CoC. The third factor is the intensity leakage control function. This function adds a limitation on the weight based on the distance from the camera and the focal plane resolving leakage issues. The final weight of the pixel intensity is the product of the three functions mentioned:

$$W(P)=O(r_p)\times I(r_p)\times L(z_p)$$

Where $O$ represent the overlap function, $I$ is the light intensity function, and $L$ is the leakage control function. The authors stated that their method computes the circles of confusion accurately using existing depth buffer information. They also claimed that the algorithm is well suited to a GPU implementation since it takes advantage of vector-processing and parallelism capabilities of the GPU. Disadvantages of this method include limitations with scenes having large depth of field and problems related to transparency.

Mulder and Liere presented a different approach for simulating DOF by combining accuracy and speed using a hybrid model. Their algorithm is based on two techniques. The first is an accurate

high resolution technique for the center of attention. In this technique the diameters of the CoC are approximated (discretized) to pixel sizes and their borders are calculated. The border of the CoC is a list of pixels that reside within the circles but are not covered by a smaller CoC. Pixels are located within a CoC if their centers lies inside the circle. The result of this technique is based on a series of different pixel intensities blended together to form the final image. This technique makes use of fast texturing hardware, parallelism, and uniform intensity distributions over the CoCs.

The second technique of the Mulder and Liere algorithm involves a high speed low accuracy approximation filter. It is based on Gaussian pyramids, a fast way to create reduced size low-passed filtered images. A Gaussian pyramid is constructed from the original image by computing weighted average values within a 5 by 5 area. To generate the DOF effect, two pyramids are needed. The source image of the first pyramid has all the pixels closer to the camera rendered with alpha values set to 1. The second pyramid represents pixels that are further away than the focus plane. Each of these two images are stored as a 2D texture. To generate the final image the algorithm renders, front to back, texture mapped polygons using the front pyramid textures. The pixels that are in the focus plane are then blended in. The last step is to blend in, back to front, texture mapped polygons using textures from the back pyramid. This technique helps create fast highly blurred effects compared to convolution filters [2]. A disadvantage of this technique is its vulnerability to the intensity leakage problem.

In order to achieve both accurate and fast simulations of DOF the two techniques presented by Mulder and Liere can be combined. The combined algorithm defines a center of attention volume (CAV) located inside the viewing volume. The center of the CAV is the object currently in focus. The high resolution technique is applied inside the CAV where as the fast approximation technique is applied in the outside. This insures that accuracy is provided in the most important areas.

Another DOF simulation approach, designated towards film production, was developed by Kass (Pixar Studios), Lefohn, and Owens (U.C. Davis) [6]. Their algorithm approximate depth of field computations for interactive film preview. The quality of this approximation should be sufficient to allow cinematographers to modify and approve aperture settings. The idea behind the introduced algorithm is to compute DOF blurring by simulating the heat diffusion equation. The image intensities provide a heat distribution that can be used to produce a DOF effect. Blurring is based on the size of the circles of confusion. For a large CoC the thermal conductivity is high resulting in more blurring whereas a zero sized CoC represent an insulator and, therefore, the object will be rendered as in focus. The algorithm uses the input image to get the initial heat distribution and then integrate the heat equation through time to obtain the desired blurred result.

Other algorithms uses the elliptical weighted average (EWA) surface splatting to simulate DOF. An implementation by Křivánek et al. uses an algorithm that does not suffer from intensity leakage and can handle rendering transparent surfaces [7]. The algorithm also features the idea of

using level-of-detail to determine the amount of blurring applied. This algorithm differers from the rest with the fact that it blurs individual splats using low-pass filters instead of blurring the image itself. The effect is accelerated due to the use of different level-of-detail depending on wither or not the surface is in focus making the speed of the algorithm independent of the amount of blurring required. This algorithm, however, faces similar challenges of the previous ones including speed and accuracy. Figure (5) shows an example of DOF rendering with transparency using this algorithm.

## 5. LIMITATIONS

In applications like film and TV, where the time needed to render the final CG scene is part of the production, DOF is simulated using the accurate methods. These include distributed ray tracing and multi-pass rendering. Current hardware cannot supply the computational power demanded by these algorithms to render in real-time. This is also true for most post-processing DOF methods.

The approximation algorithms surveyed in this paper produced promising results though their performance have not yet reached the scope of acceptable real-time rendering. However, current graphic cards vendors are developing fast GPUs and efficient architectures that would allow in the future similar methods to perform at higher frame rates. With the increase of clock and memory speed the requirements for fast filtering and texture look ups would not be an issue. Hardware architecture may also include special processing units for applying blurring filters. Similar work has already been done on adding special hardware features to accelerate shaders and full scene anti-aliasing on graphic cards.

Virtual reality applications face another challenge when simulating DOF. Accommodation and convergence are strongly linked in human viewing and by using head mounted display devices the relationship will be violated causing eye strains [2]. Mulder and Liere suggested that simulating DOF and constructing head mounted displays with eye tracking systems might restore the convergence-accommodation cue and help prevent eye strains.

## 6. CONCLUSIONS

Depth of field effects help achieve realism and improve the user's perception of the 3D scene. There are many applications that can benefit from simulating DOF including film, augmented reality, video games, graphical user interfaces, and virtual reality. This paper included a description of the common algorithms used today to simulate the effect. The algorithms ranged from fast, yet inaccurate, approximations to slow computationally expensive accurate simulations. Most of the current research focuses on post-processing methods since they require less computational work than multi-pass rendering. Several methods to achieve real-time performance have been proposed with promising results that suggest reaching real-time speeds in the near future.

## ACKNOWLEDGMENTS

Thanks to my friend Kristi Smith for proofreading the paper and suggesting some changes.

## REFERENCES

[1] M. Bertalimío, D. Sánchez-Crespo. Real-Time, Accurate Depth of Field using Anisotropic Diffusion and Programmable Graphics Cards. In *Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04),* p.767-773, September 06-09, 2004.

[2] J. D. Mulder and R. van Liere. Fast perception-based depth of field rendering. In *VRST 2000,* pages 129–133, 2000.

[3] C. Kolb, D. Mitchell, P. Hanrahan. A realistic camera model for computer graphics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, p.317-324, September 1995.

[4] T. Zhou, J. X. Chen, M. Pullen. Accurate Depth of Field Simulation in Real Time. In *Computer Graphics Forum 26 (1)* , 15–23, 2007.

[5] R. Kosara, S. Miksch, H. Hauser. Semantic Depth of Field. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, p.97, October 22-23, 2001.

[6] M. Kass, A. Lefohn, and J. Owens. Interactive Depth of Field Using Simulated Diffusion on a GPU. Technical report, Pixar Animation Studios, 2006.

[7] J. Krivanek, J. Zara, and K. Boutouch. Fast Depth of Feld Rendering with Surface Splatting. In *Proceedings of Computer Graphics International 2003*, 2003.

# APPENDIX

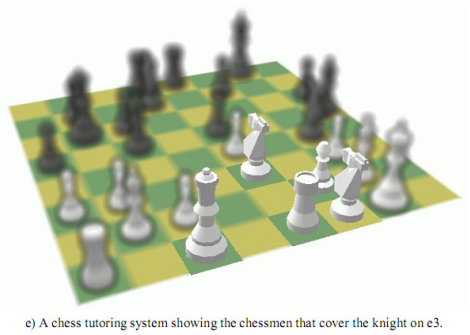


*Figure 1: An example of the SDOF technique [5]*

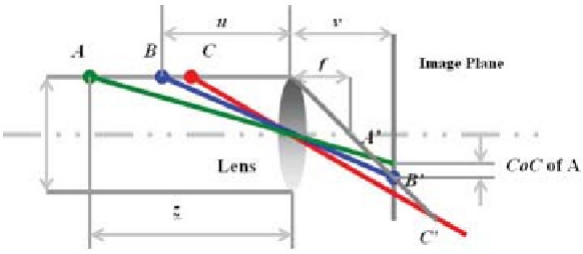


*Figure 2: The thin lens model [4]*

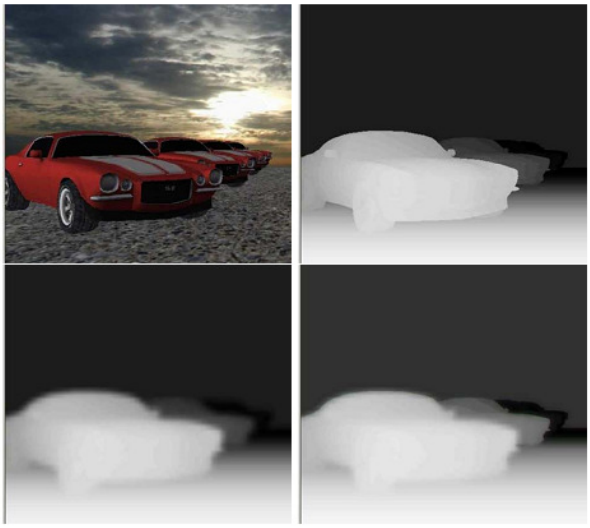

*Figure 3: Stages of preprocessing the scene [1]*

*Figure 4: Top: original pinhole image. Bottom: image with simple DOF [6]*

*Figure 5: Example of ODF rendering with semitransparent surfae. Left: no DOF. Middle: DOF is on and transparent mask is in focus. Right: the male body is in focus, the mask is out of focus.[7]*