# Depth-of-Field Rendering for Point-Based Object

**J. Křivánek**

krivanek@fel.cvut.cz

Department of Computer Science and Engineering
Czech Technical University in Prague
Karlovo náměstí 13, 121 35 Praha 2, Czech Republic

IRISA - INRIA Rennes
Campus de Beaulieu
35042 Rennes Cedex, France

The ability to render the depth-of-field effect is an important feature of any image synthesis algorithm. Depth-of-field increases the naturalness of the image, since both optical systems in cameras and in human eye have lens of final aperture and do not produce perfectly focused images. Depth-of-field is also an important depth cue that helps humans to perceive the spatial configuration of a scene and is also important in stereoscopic image generation.

Depth-of-field is inherent to camera models with lens of finite aperture, however in computer graphics the most commonly used camera model is the *pinhole camera,* where the lens is supposed to be infinitely small. The pinhole camera model produces images that are in sharp focus in any distance from the viewer. This can be a desired feature (e.g. for technical visualizations), but often a more realistic 3D image including depth-of-field is required.

The effect of depth-of-field is, that out-of-focus points in 3D space form circular patterns (circle of confusion) in the image plane. The most common, *post-filtering* algorithms for depth-of-field rendering [2], generate the depth-of-field in a post-processing step: first a pinhole-camera image is generated and then a post processing step turns the out-of-focus pixels into the circles of confusion. This algorithm suffers from the *intensity leakage* (*e.g.* blurred background leaks into focused object in foreground) and does not handle the *partial occlusion* (visibility of objects changes for different positions on the lens of final aperture). Several algorithms that solve those problems exist, but they are currently slow, especially for large amounts of depth-blur.

We present a new, fast depth-of-field rendering algorithm for point-based objects. The *point-based graphics* has recently gained a lot of research focus. In point-based modeling the 3D objects are represented by a cloud of points in 3D space with no connectivity information among them. The reasons for using a point-based representation are manifold. For example the point-based representation is a natural output of the 3D scanning devices that are becoming to a widespread use not only in the research, but also in practical applications. Also the rendering is faster for points than for triangles if the modeled shape is highly detailed.

There are more approaches to render the points as continuous surface. Our algorithm builds on top of *surface splatting* [3,4]. In this method, every point of the point-based representation of a 3D objects is displayed as a "splat" or "footprint" on the screen. The splats overlap in screen space and the weighted averages of splats' contributions to different pixels are used for *high quality texture antialiasing*. Another features of surface splatting that we can directly profit from are order-independent transparency and edge antialiasing.

Our algorithm generates depth-of-field by a filtering, as the post-filtering algorithms do. However, we filter the individual splats before producing the final image. This actually decouples the visibility determination from the depth-of-field rendering and allows us to handle the partial occlusion and to avoid the intensity leakage. The algorithm also allows for depth-of field rendering in presence of transparent surfaces.

To speed-up the depth-of-field rendering, we use the concept of level-of-detail. We profit from the fact, that the level-of-detail hierarchies prefilter the surface representation for coarser levels. When a large amount of filtering is required to produce the depth-of-field, we use the coarser level-of-detail hierarchy level, thus saving the computational resources. With this technique the speed of the algorithm is practically independent of the amount of depth-blur. This is a big advantage over the existing algorithms, where the rendering speed drops down with the square of the aperture diameter used for simulating the depth-of-field effect.

In our work [1], we present a mathematical analysis extending the surface splatting to include the depth-of-field rendering ability. We also present an analysis allowing to use the level-of-detail as the means for depth-of-field rendering, including the criterions for level-of-detail selection. We give the implementation of the algorithm, and a discussion of practical issues arising from the implementation as the normalization of the splats' contributions, surface reconstruction from the depth-blurred splats and shading. The timings obtained from the implementation confirm the independence of the rendering time on the amount of depth-blur.

The drawbacks of the algorithm are high sensitivity to the regularity of sample positions on the surface of point-based object and occasional artifacts due to the incorrect surface reconstruction.

As a future work we want to implement the depth-of-field rendering algorithm for volume rendering, which should be a straightforward extension of the algorithm we developed for point-based surfaces. We also want to develop a specialized point-processing tool, which could be used for normalization of point-based objects.

**References:**


[1] KŘIVÁNEK, J. - ŽÁRA, J. - BOUATOUCH, K.: *Fast Depth of Field Rendering with Surface Splatting.* Submitted to Computer Graphics International, 2003.

[2] POTMESIL, M - CHAKRAVARTY, I.: *A Lens and Aperture Camera Model for Synthetic Image Generation.* SIGGRAPH '81 Proceedings, 1981.

[3] ZWICKER, M. - PFISTER, H. - VAN BAAR, J. - GROSS, M.: *Surface Splatting* SIGGRAPH 2001 Proceedings, 2001.

[4] ZWICKER, M. - PFISTER, H. - VAN BAAR, J. - GROSS, M.: *EWA Splatting.* IEEE Transactions on Visualization and Computer Graphics, 8(3):223-238 2002.



*This research has been supported by MŠMT grant No. MSM 2159/2002.*