How to write a Ray Tracing Gem article

2018-06-13, v1.0

Tomas Akenine-Möller
NVIDIA Research

Abstract

This document describes how to write an article for submission in the *Ray Tracing Gems* book series. General recommendations will be discussed that may help increase the chance of acceptance and provide a coherent look of the book.

1 Content

See our website https://developer.nvidia.com/raytracinggems for the Call for Papers (CFP). We encourage article submissions on the following topics (taken from the CFP):

- Basic ray tracing algorithms (intersection testing, spatial data structures, etc.)
- Effects (shadows, reflections, ambient occlusion, etc.)
- Non-graphics applications (for example, audio)
- Reconstruction, denoising, and filtering
- Efficiency and best practices
- Ray tracing API and design
- Rasterization and ray tracing
- Global illumination
- BRDFs
- VR and AR
- Deep learning

However, other notable work related to ray tracing is welcome as well. Novel article content naturally is welcome, but we also hope to get submissions on useful tips and tricks that may be well-known in the graphics community, but not necessarily published before. The article should ideally be important and useful to many people, both in the industry and in academia. We are on a tight time schedule, and so we request that you submit highly polished articles (e.g., high-quality figures and images and text that has been proofread many times). Please reach out if you have any problems or questions.[1]

2 General Guidelines

We want to provide a coherent book, so it is important that we all use the same notation and writing principles. This document was written in LaTeX using a custom class file called `rtg.cls`, and your submission should be written using that setup.[2] The best way to format your article is to download the LaTeX-files for this document and start by changing the `sample.tex` and `references.bib` files. See https://www.latex-project.org/get/ for instructions how to download and install LaTeX on various platforms. TeXstudio (https://www.texstudio.org/) is a nice multi-platform editor, but you can choose anyone you like.

Use rendered images and performance numbers or diagrams to convince the reader that your work is worthwhile. Timing should be measured in milliseconds (or seconds) instead of frames per second, and compare both to ground truth images (when applicable) and to other algorithms' images.

In the following subsections, we discuss details of different types of notation and things (e.g., pseudocode) that can be included in the document.

[1] `raytracinggems@nvidia.com`
[2] Note, however, that the final layout and look of the template may still change.

2.1 Equations and Math

Let the equations be numbered, so it is easy to discuss an equation with another person. Equations should be referenced like this: See Equation 1, for example. A fantastic formula is summarized as

$$s = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}. \tag{1}$$

Vectors should be denoted by bold lower-case letters, e.g., $\mathbf{v}$, and matrices by bold upper-case letters, e.g., $\mathbf{M}$. Scalars are lower-case, italicized letters, e.g., a and v. Points are uppercase, e.g., P. The components of a vector are accessed as

$$\mathbf{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^\intercal, \tag{2}$$

where the latter shows the vector transposed, i.e., so a column becomes a row. Note that to simplify writing text with vectors, we also accept $\mathbf{v} = (v_x, v_y, v_z)$, i.e., where the scalars are separated by commas, which indicates that it is in fact a column vector shown transposed. We use column vectors by default, which means that matrix-vector multiplication is denoted $\mathbf{Mv}$. The components of a matrix are acccessed as

$$\mathbf{M} = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_2 \end{pmatrix}, \tag{3}$$

where $\mathbf{m}_0$, $i \in \{0, 1, 2\}$ are the column vectors of the matrix.

For normalized vectors, we use the following short-hand notation:

$$\widehat{\mathbf{d}} = \mathbf{d}/\|\mathbf{d}\|, \tag{4}$$

i.e., if there is a line over the vector, it is normalized. A transposed vector is denoted $\mathbf{x}^\intercal$. Try to use the following commands for vectors, matrices, normalized vectors, and transpose:

Notation	LaTeX code
$\mathbf{v}$	`\vc{v}`
$\mathbf{M}$	`\mx{M}`
$\widehat{\mathbf{v}}$	`\nvc{v}`
P	`\pt{P}`
$\mathbf{v}^\intercal$	`\transpose{\vc{v}}`
$\mathbf{M}^\intercal$	`\transpose{\mx{M}}`
Ω	`\setOfDirsOnSphere`
ω	`\dirOnSphere`,

(5)

where the two last rows should be used for solid angles, i.e., Ω is a set of directions on the unit sphere and ω is a vector in the unit sphere. Finally, note that the cross product between two vectors is written as $\mathbf{a} \times \mathbf{b}$ and their dot product is $\mathbf{a} \cdot \mathbf{b}$.

2.2 Figures, Images, and Tables

All figures, images, and tables should be referenced in the text, and have informative captions. An example table is shown in Table 1, and in Figure 1, we show an example of a PDF illustration inserted into the document.

Illustrations and images should use CMYK color mode. We recommend using PDF and jpg (max quality level) as the main formats, since those support CMYK. Figure 2 shows a screenshot from our work.

	A	B	C
1	something	else	here
2	numerous	important	things

Table 1: In this table, we show several great things.

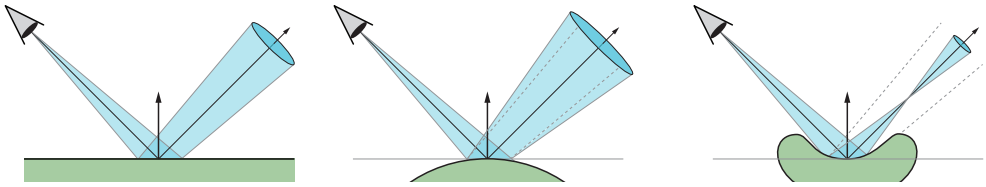

Figure 1: Left: planar reflection. Middle: convex reflection. Right: concave reflection. As can be seen, all types of reflections reflect.

Figure 2: An image rendered using several different types of rays. *(Image courtesy of Petrik Clarberg, Jon Hasselgren, Jacob Munkberg, and Chris Wyman).*

2.3 Code

Code can help understanding a lot, but articles may also become long if used excessively. As a general rule, use code or pseudocode only when you think it is important for the understanding of the article. Everyone is encouraged to submit supplemental code as well, for example, using Falcor [3]. Here is a bit of example code.

```
struct Ray
{
    float3 origin;
    float3 direction;
};

float mydot3d(float3 u, float3 v)
{
    return mult(u[0],v[0]) + u[1]*v[1] + u[2]*v[2]; // dot product in 3D
}

float mydot2d(float2 u, float2 v)
{
    return u[0]*v[0] + u[1]*v[1]; // dot product in 2D
}
```

2.4 References

A good rule for references is that it should always be possible to remove the reference and after that, the sentence should still be complete. As an example, do not write: This was shown in [2]. Instead, write: This was shown by Bako et al. [2]. Make sure your references are complete, e.g., they have page numbers and conference names. Do not use abbreviations, for example, "ACM Trans. Graph." but rather "ACM Transactions on Graphics." Here is an HPG reference [6], a SIGGRAPH paper [2], a GDC presentation [1], a book [4], an in-collection reference [7], and an IEEE paper [5].

Also, try to make sure that your references will stay alive, i.e., avoid volatile references, such as web pages that may not last long.

2.5 Finally

You may want to use the https://github.com/erich666/chex_latex tool to check for basic problems in your submission.

Good luck with your submission—we're looking forward to it!

Acknowledgement

Thanks to Petrik Clarberg for reading and comment on the first draft and to Eric Haines for dozens of nit-picking comments.

References

[1] Andersson, J. 25 Major Challenges in Real-Time Rendering. In *Beyond Programmable Shading, SIGGRAPH Courses* (2012).

[2] Bako, S., Vogels, T., Mcwilliams, B., Meyer, M., Novák, J., Harvill, A., Sen, P., Derose, T., and Rousselle, F. Kernel-predicting Convolutional Networks for Denoising Monte Carlo Renderings. *ACM Transactions on Graphics 36*, 4 (2017), 97:1–97:14.

[3] Benty, N., Yao, K.-H., Foley, T., Kaplanyan, A. S., Lavelle, C., Wyman, C., and Vijay, A. The Falcor Rendering Framework, July 2017.

[4] Fairchild, M. D. *Color Appearance Models*, 2nd ed. John Wiley & Sons, 2005.

[5] Heckbert, P. S. Survey of Texture Mapping. *IEEE Computer Graphics and Applications 6*, 11 (1986), 56–67.

[6] Kaplanyan, A. S., Hill, S., Patney, A., and Lefohn, A. Filtering Distributions of Normals for Shading Antialiasing. In *High Performance Graphics* (2016), pp. 151–162.

[7] Watson, A. B. Temporal Sensitivity. In *Handbook of Perception and Human Performance*, K. R. Boff and L. Kaufman, Eds., vol. 1. John Wiley & Sons, 1986, pp. 6:1–6:43.