Exposé: Volumetric Path Tracing inside the Unity Game Engine Extended

Research Lead: *Prof. Dr. Daniel Roth* **Type:** *Bachelor Thesis Exposé*
Date: *19.05.2025* **Supervisor:** *Constantin Kleinbeck* **Author:** *Lucas Denhof*

I. Project Abstract

In this thesis, the VolRen[3] volume path tracer will be ported into the Unity game engine. The goal is to make it seamlessly connect to the rest of Unity's systems. Additional rendering effects such as depth of field are planned to be implemented, and once optimized, it will be investigated whether a variant can be made for rendering in real-time, for example in VR. Finally, it will be compared with other state-of-the-art volumetric renderers in rendering speed and output quality.

II. Introduction

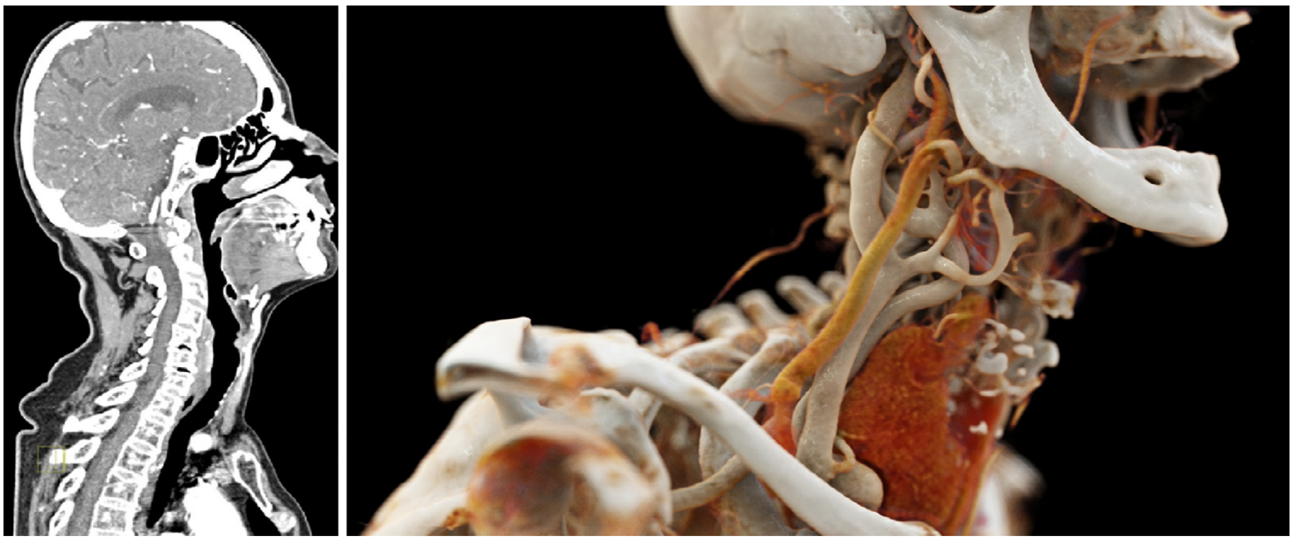
Fig. 1: Left: Original CT data. Right: Cinematic rendering of the same dataset.[1]

For a long time, doctors have been using two-dimensional slices of MRI scans to glean information out of the injuries their patients have suffered. However, this approach is flawed: We are three-dimensional creatures, and so are our injuries. The absence of an extra dimension is limiting doctors and their diagnoses.

Why not leverage the vision we have evolved with, and use realistic three-dimensional renders to create more intuitive and helpful tools for doctors? Such photorealistic renders will aid with our natural object recognition and depth perception, helping doctors reach the correct diagnosis. These images may also prove to be more intuitive for people who aren't familliar with interpreting 2D scans, such as young or inexperienced doctors, or even the patients themselves.

As it stands, two-dimensional renderers have been used for decades and many doctors have familiarised themselves with these slices. Three-dimensional renderers have also existed for some time, but they didn't always look photorealistic. Only in recent years have graphics cards, that are capable of rendering complex ray-traced 4K scenes in realtime or even faster, become so ubiquitous. The quality and interactivity of renderers powered by these graphics cards warrents investigating photorealistic renderers as a supplement, or alternative to the currently used two-dimensional renderers.

This thesis aims to investigate the feasibility of using volumetric path tracing to create three-dimensional renders. We seek to integrate them into the Unity game engine, allowing for easy, cross-platform, and real-time use in a variety of applications, perhaps even VR.

Standard medical formats for CT and MRI scans, such as DICOM, are to be easily imported, and the Transfer Function, which maps color and opacity, should should be freely editable in the build.

By doing all this, we wish to provide an simple but powerful tool for the medical world. Our hope is to provide an easy stepping stone for others who wish to do research in this area.

III. Related Work

Hofmann and Evans [4] created the structure for a volumetric path tracer which was implemented in VolRen in OpenGL, and it is the core for much that will be built in this paper.

Giesse [2] implemented the VolRen in the Unity game engine with room for further research. His work will be continued in this paper.

Lavik [6] created UnityVolumeRendering, where inspiration will be taken from the successful integration of a ray marching algorithm in Unity.

Comaniciu et al. [1] explore various techniques to improve medical imaging. Among other things, they discuss their Cinematic Rendering renderer which creates impressive images using Monte Carlo path tracing.

Kroes, Post, and Botha [5] show the approach they used for Exposure Render, which uses Monte Carlo ray tracing to render volumes, and thereby allows a large variety of effects to be easily and flexibly implemented.

Ljung et al. [7] report on the state-of-the-art research on transfer functions, which provide a conversion to show the data inside of a volume as color and opacity.

IV. Approach

This thesis plans to build off of and continue Giesse's work[2] in porting the VolRen[4] renderer into the Unity game engine.

After conducting sufficient research into the current state-of-the-art shaders, VolRen will be finished being ported into Unity. Bugs and other issues that were identified by Giesse, and that we will identify, will be fixed. Examples include problems with the chunk boundaries, extending the RGB output with e.g. specular, and making data importing and changing of the transfer function work in the build, and not just in the editor.

Once it has been properly integrated into all of Unity's systems, new effects will be added, such as depth of field or denoising.

Finally, the implementation will be optimized, and it will be investigated whether a simplified version can be made for realtime rendering. The goal would be to get the best possible quality in the shortest possible time, likely sacrificing some rendering effects for speed, and trying out optimizations like denoising, biased rendering, and deduplication.

Our solution will then be compared with various other state-of-the-art volumetric renderers. It will be measured how long it takes our renderer to converge on an image in frames and samples. Various scenes of different complexity will be used, and benchmarking will be done with and without the rendering effects that were implemented. It will be calculated how accurate the first noisy samples are, compared to the converged image, by looking at the peak signal-to-noise ratio (PSNR) and the structural similarity index measure (SSIM).

Since no ground truth is available, we cannot accurately use PSNR or SSIM to calculate how physically accurate the renderer is, and therefore we cannot numerically compare it to other volumetric renderers. A mere visual comparison with these other renderers will have to suffice, along with the rendering time.

V. Goals/Objective

- Finish porting VolRen into Unity.
- Make it interact well with all of Unity's systems, such as:
  - Rendering
  - Occlusion
  - Lighting
  - Colliders
  - Physics
  - The inspector/scene view
  - etc.
- Fix bugs in, and expand upon the existing implementation.
- Add new features, such as depth of field or denoising.
- Optimize the existing implementation.
- Investigate whether a simplified version can be made for realtime rendering or virtual reality.
- Add importing of standard files for medical scans during runtime.
- Add an editor for transfer functions in the build.
- Benchmark and compare with state-of-the-art.

VI. AI Usage

Over the course of writing this document, QuillBot has been used in order to correct various minor spelling and grammar mistakes.

ChatGPT-4o has also been used to display possible completions of sentences when looking for a certain word but unable to find it, or to help learn about some LaTeXcommands.

Otherwise, **no** other artificial intelligence tools such as ChatGPT, Grammarly, GitHub Copilot, DeepL, DeepSeek, or similar have been used.

References

[1] Dorin Comaniciu et al. "Shaping the future through innovations: From medical imaging to precision medicine". In: *Medical Image Analysis* 33 (Oct. 2016), pp. 19–26. ISSN: 13618415. DOI: 10.1016/j.media.2016.06.016. URL: https://linkinghub.elsevier.com/retrieve/pii/S1361841516300962 (visited on 06/01/2025).

[2] Simon Giesse. "Volumetric Path Tracing inside the Unity Game Engine". In: ().

[3] Nikolai Hofmann. *nihofm/volren*. original-date: 2023-03-17T00:23:49Z. June 4, 2025. URL: https://github.com/nihofm/volren (visited on 06/12/2025).

[4] Nikolai Hofmann and Alex Evans. “Efficient Unbiased Volume Path Tracing on the GPU”. In: *Ray Tracing Gems II: Next Generation Real-Time Rendering with DXR, Vulkan, and OptiX*. Ed. by Adam Marrs, Peter Shirley, and Ingo Wald. Berkeley, CA: Apress, 2021, pp. 699–711. ISBN: 978-1-4842-7185-8. DOI: 10.1007/978-1-4842-7185-8_43. URL: https://doi.org/10.1007/978-1-4842-7185-8_43 (visited on 06/01/2025).

[5] Thomas Kroes, Frits H. Post, and Charl P. Botha. “Exposure Render: An Interactive Photo-Realistic Volume Rendering Framework”. In: *PLoS ONE* 7.7 (July 2, 2012). Ed. by Xi-Nian Zuo, e38586. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0038586. URL: https://dx.plos.org/10.1371/journal.pone.0038586 (visited on 06/01/2025).

[6] Matias Lavik. *mlavik1/UnityVolumeRendering*. original-date: 2019-02-11T16:16:35Z. May 30, 2025. URL: https://github.com/mlavik1/UnityVolumeRendering (visited on 06/01/2025).

[7] Patric Ljung et al. “State of the Art in Transfer Functions for Direct Volume Rendering”. In: *Computer Graphics Forum* 35.3 (June 2016), pp. 669–691. ISSN: 0167-7055, 1467-8659. DOI: 10.1111/cgf.12934. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.12934 (visited on 06/01/2025).