VVRT: Virtual Volume Raycaster

L. v. d. Wal, P. Blesinger, J. Kosinka, and S. Frey

University of Groningen, Bernoulli Institute, The Netherlands

Abstract

Virtual Ray Tracer (VRT) is an educational tool to provide users with an interactive environment for understanding ray-tracing concepts. Extending VRT, we propose Virtual Volume Raycaster *(VVRT), an interactive application that allows to view and explore the volume raycasting process in real-time. The goal is to help users—students of scientific visualization and the general public—to better understand the steps of volume raycasting and their characteristics, for example the effect of early ray termination. VVRT shows a scene containing a camera casting rays which interact with a volume. Learners are able to modify and explore various settings, e.g., concerning the transfer function or ray sampling step size. Our educational tool is built with the cross-platform engine Unity, and we make it fully available to be extended and/or adjusted to fit the requirements of courses at other institutions, educational tutorials, or of enthusiasts from the general public. Two user studies demonstrate the effectiveness of VVRT in supporting the understanding and teaching of volume raycasting.*

CCS Concepts

• ***Applied computing*** → ***Interactive learning environments;*** • ***Human-centered computing*** → ***Scientific visualization;*** • ***Computing methodologies*** → ***Ray tracing;***

1. Introduction

Volume raycasting is one of the key methods in scientific visualization and is widely taught in visualization courses due to its prevalence and importance. The understanding of volume raycasting and the impact of different parameter settings and acceleration techniques like early ray termination can greatly be supported via visualization. Simple illustrations[†] are commonly employed to illustrate the different steps of the process (ray setup, sampling, shading, and compositing), but their 2-dimensional and static nature limits their effectiveness as an educational tool.

With Virtual Volume Raycaster (VVRT), we propose a 3D approach to visualize volume raycasting and make it widely accessible. The main demographic for the application are students of university-level (scientific) visualization courses. However, the application is built to be accessible to as wide an audience as possible, so anyone interested in volume raycasting should be able to understand and explore its underlying concepts[‡]. VVRT has been developed and evaluated in the context of two student BSc theses [Wal23, Ble24]. It builds upon our previous work on Virtual Ray Tracer (VRT) [VdlHWFK22, vWVdlHFK23], an interactive application that visualizes the ray tracing process.

The contributions of VVRT include:

- interactive visualization of the different steps involved in volume raycasting (with early ray termination);
- a live raycasting view of the volume that is consistently generated with the same process and settings that are employed in the raycasting visualization;
- user studies showing the promise of VVRT in education.

We review related work in Sec. 2 and discuss the visual mapping of volume raycasting concepts and steps in Sec. 3. Sec. 4 outlines the implementation of the live raycasting feature. An evaluation of VVRT based on user studies is given in Sec. 5. We then discuss VVRT and its pedagogical use cases in Sec. 6, and finally conclude the paper in Sec. 7.

2. Related Work

For teaching volume raycasting, academic courses and tutorials often incorporate simplified illustrations to guide students through concepts like volume integration and gradient computation for shading. However, to the best of our knowledge, no comprehensive interactive tool exists to date that is dedicated to providing students with an interactive tool to explore the overall procedure and investigate the different components. VVRT aims to address this gap. It is the result of two student theses [Wal23, Ble24] and based on VRT that visualizes the ray tracing process [VdlHWFK22, vWVdlHFK23]. VVRT extends VRT in order to teach concepts specifically related to volume raycasting.

While several other applications exist that visualize specific as-

[†] (for example the Wikipedia figure at https://commons.wikimedia.org/wiki/File:Volume_ray_casting.svg#file)

[‡] https://github.com/LukkeWal/VVRT

DOI: 10.2312/eved.20251021

pects and metrics of ray-based rendering techniques [SHP*19, SAH*16, SJL15, ZAD15], these are not primarily intended for educational purposes. The prevalence of such tools underscores the value of visualization in understanding such techniques. While there are some applications focusing on teaching ray tracing for example, these do not directly visualize the ray tracing process itself [SSM02, VGV*20].

One of the earliest examples to depict ray tracing is a set of Java Applets created in 1999 [Rus99]. A more recent application is the Ray Tracing Visualization Toolkit (rtVTK) [GFE*12]. As the name implies, rtVTK serves as a toolkit for visualizing ray tracing. Its primary objectives are to support the development of ray tracing applications and to assist with ray tracing education. However, the authors acknowledge that rtVTK may be overly complex for the latter purpose. A significant limitation is that users must integrate the toolkit with their own ray tracing application. While this approach offers versatility, enabling usage across a wide range of applications, it poses challenges in accessibility. For an educational tool aimed at a broad audience, this complexity is a notable drawback.

VRT [VdlHWFK22] addresses this gap by offering a dedicated, user-friendly educational application specifically designed to visualize the ray tracing process. VRT 2.0 [vWVdlHFK23] builds on this foundation with several enhancements and extensions: the integration and visualization of distributed ray tracing techniques [CPC84, SM09] (such as those underlying area lights and soft shadows), the incorporation of acceleration structures for spatial data [FTI86, HAM19, MSW21], and the use of gamification concepts [RW15, Nic12] to make the tool engaging and enjoyable. In a similar fashion, VVRT extends VRT with raycasting capabilities and visualizations.

3. Visualization of the Volume Raycasting Process

Volume raycasting like ray tracing turns a 3D scene into 2D images [SM09]. They both follow a ray through 3D space and consider the interactions of the ray with objects in the scene to calculate a color value for a pixel. While ray tracing calculates interactions with objects' surfaces only, volume raycasting considers a participating medium and a ray accumulates contributions from the object (the volume) at every point along its way. In this section, we discuss the visual mapping of raycasting concepts in VVRT alongside a brief introduction to volume raycasting: volume data (Sec. 3.1), ray setup and sampling (Sec. 3.2), transfer functions (Sec. 3.3), and compositing methods (Sec. 3.4). Across all steps, the visualization of the raycasting process reflects directly what is done when producing accompanying volume renderings (e.g., Fig. 1 depicts generating Fig. 2c only with a lower screen resolution, i.e., fewer rays, for the sake of visual clarity).

VVRT is accompanied by a dedicated tutorial (see the accompanying video). It walks the user through all essential functions of the tool, including changing the voxel grid (volume data), sampling parameters, the compositing function, and the transfer function. Implementation details are described in supplementary material.

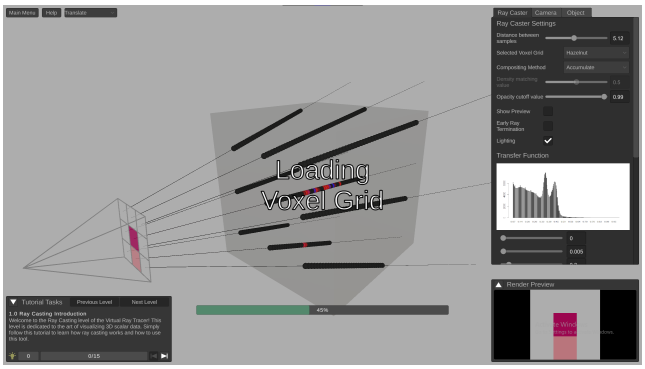
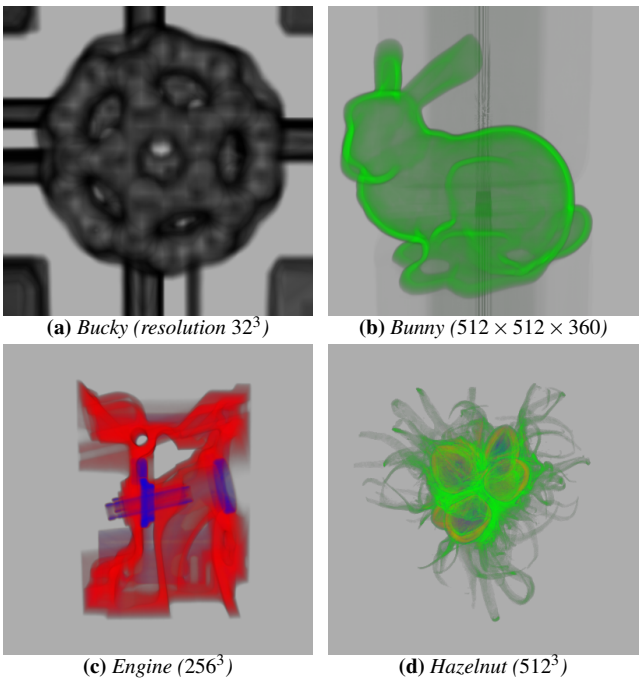


Figure 1: *Voxel grid (volume data) loading.*



(a) *Bucky (resolution* 32^3*)* **(b)** *Bunny* ($512 \times 512 \times 360$) **(c)** *Engine* ($256^3$) **(d)** *Hazelnut* ($512^3$)

Figure 2: *Volume renderings with default transfer functions.*

3.1. Volume Data

Volume data refers to a 3D domain with a scalar attribute: $\mathbb{R}^3 \to \mathbb{R}$. Throughout this work, we simply consider uniform grids to represent the data, where each cell corresponds to a volume element (a voxel). In the following, we refer to the scalar quantity as *density* ($\in [0, 1]$). In VVRT, the voxel grid is visualized using a transparent gray box by default. This means that what is inside these boxes can only be seen in a render preview window (e.g., bottom right in Fig. 1), or when rendering a high-resolution image using the render image button (Fig. 2 and our video). This is generated by a CPU-based volume raycaster implemented in the Unity framework (denoted as Unity Raycaster in the following). Our GPU-based live raycasting extension that shows continuously updated renderings instead is discussed in Sec. 4.

The four example datasets provided with VVRT are shown in Fig. 2: Bucky, Bunny, Engine and Hazelnut. The user can select a

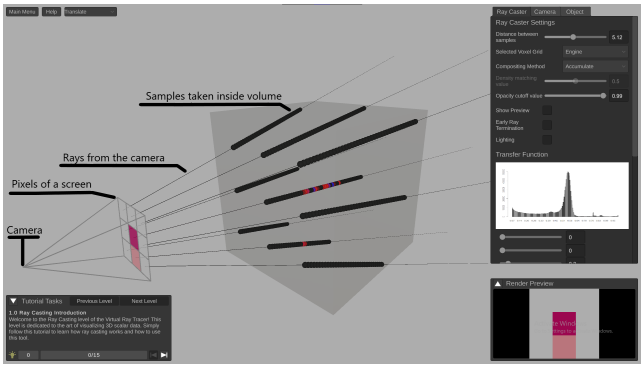
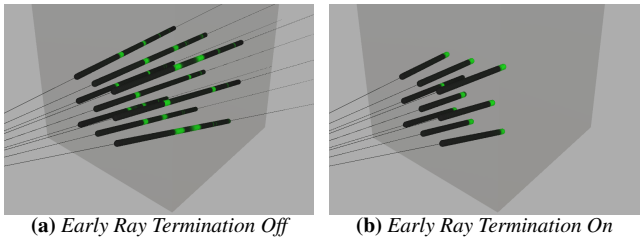


Figure 3: *VVRT scene view showing the camera, (virtual) screen, rays and samples. Middle right: Transfer function; here the default for the Engine is shown (corresponding to Fig. 2c).*



(a) *Early Ray Termination Off* **(b)** *Early Ray Termination On*

Figure 4: *With early ray termination 'on' (for **accumulate** and **first** compositing), only the samples used in calculations are shown and the ray cylinder ends at the position where the ray is 'terminated'.*

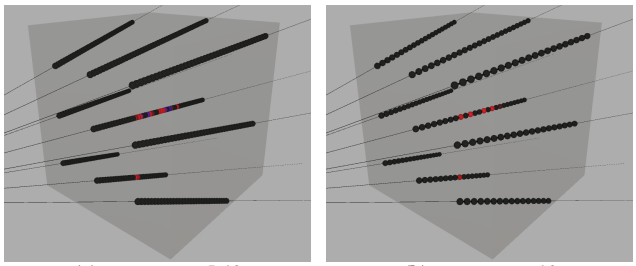
different voxel grid with a dropdown menu. As loading can take a couple of seconds, we opted to indicate the progress with a loading bar in VVRT (Fig. 1). Note that each volume comes with its default transfer function (see Sec. 3.3), which is automatically applied when loading a new volume.

3.2. Ray Setup and Sampling

Rays originate at the camera (or an observer), pass through the pixels of the screen, and determine the color of the respective pixel via (uniform) sampling while traversing through the volume. In this work, we consider front-to-back raycasting and early termination, i.e., taking samples from the start of the ray until additional samples would only have a negligible impact on the final color.

VVRT shows a scene where screen pixels are indicated via 2D rectangles and rays are depicted as cylinders starting at their corresponding pixel (Fig. 3). Samples along the rays are visualized using opaque spheres which are drawn until the point of early ray termination, where applicable. The cylinder representing the ray also stops at the respective position when early ray termination is in effect. The difference in the visualizations can be seen in Fig. 4.

The colors of the spheres and rays reflect the transfer function and the compositing method. The sampling distance along a ray can be set using a slider (Fig. 5 exemplifies different sample distances). When the 'Animate' setting is set to *true*, the samples are drawn as if taken along the rays as they travel through the grid (see the accompanying video).



(a) *Distance = 5.12* **(b)** *Distance = 10*

Figure 5: *Different sample distance settings (for the Engine).*

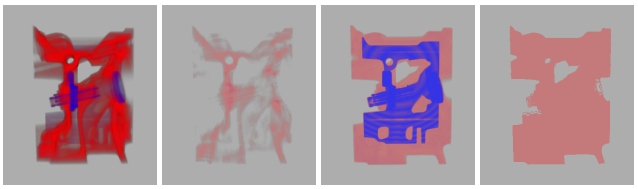


Figure 6: *Compositing Methods with Engine (no lighting). Left to right: **Accumulate**, **Average**, **Maximum**, and **First** at 0.55.*

3.3. Transfer Function

The transfer function maps the scalar values obtained during sampling to color and opacity. The color of the spheres presented in the raycasting view corresponds to the color of the respective samples (Fig. 3). In VVRT, the transfer function is visualized in the control panel where the color lookup table can be viewed and edited via sliders (Fig. 3, *center right*). To aid the user, a default transfer function is initialized every time new data is loaded. We also show the histogram of scalar values in the volume for reference (we opted to omit low density values in the histogram for clarity as all provided datasets exhibit a significant amount of 'empty' space).

3.4. Compositing Method

The (mapped) samples along a ray are finally composited into the pixel color. This is also reflected accordingly on the virtual screen in the scene in VVRT (Fig. 3, *left*).

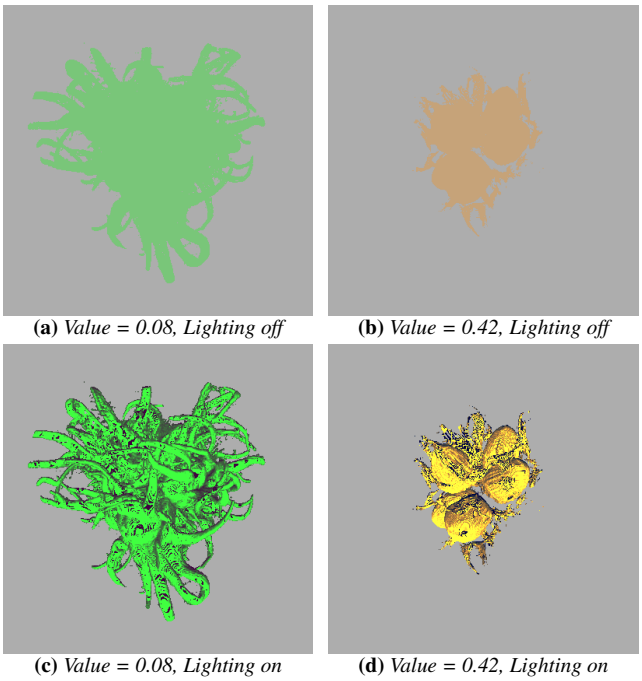
There are different ways how color can be composited from the samples. We provide the user with four compositing methods which are commonly taught in the context of scientific visualization [Tel14]. This allows to visually compare the different outcomes of the four methods (Fig. 6):

Accumulate. Incrementally incorporate the color of the next (new) sample with the composited color of all previous samples. In detail, for each sample i with (c_i, o_i) (color & opacity), we evaluate the resulting pixel color C_i and opacity O_i via $C_i = C_{i-1} + (1 - O_{i-1})c_i$ and $O_i = O_{i-1} + (1 - O_{i-1})o_i$, respectively.

Average. Take the average color and opacity of all samples.

Maximum. Use the color and opacity of the sample with the highest density value.

First. Take the color of the first sample with density equal to the density matching value (set by the user with a slider). The effects of changing this value are exemplified in Fig. 7. VVRT optionally employs Phong illumination as the local lighting model with



(a) *Value = 0.08, Lighting off* **(b)** *Value = 0.42, Lighting off*

(c) *Value = 0.08, Lighting on* **(d)** *Value = 0.42, Lighting on*

Figure 7: *The effects of the Density Matching value and Lighting in the **First** compositing method.*

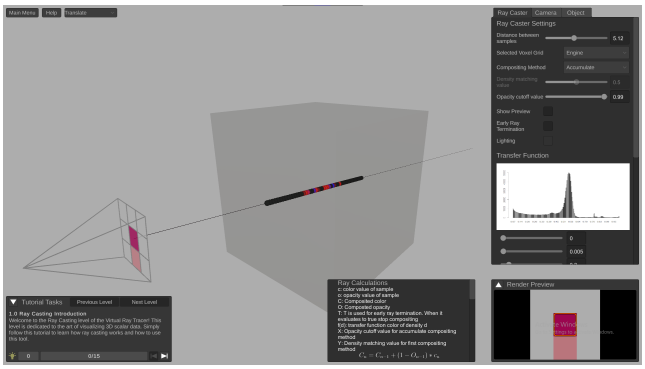
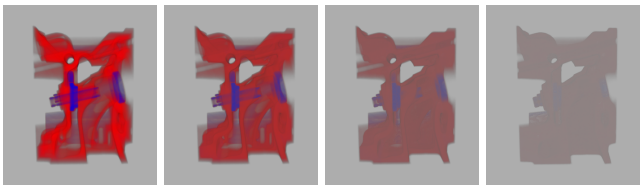


Figure 8: *The ray calculation window is located at the bottom right next to the render preview window*

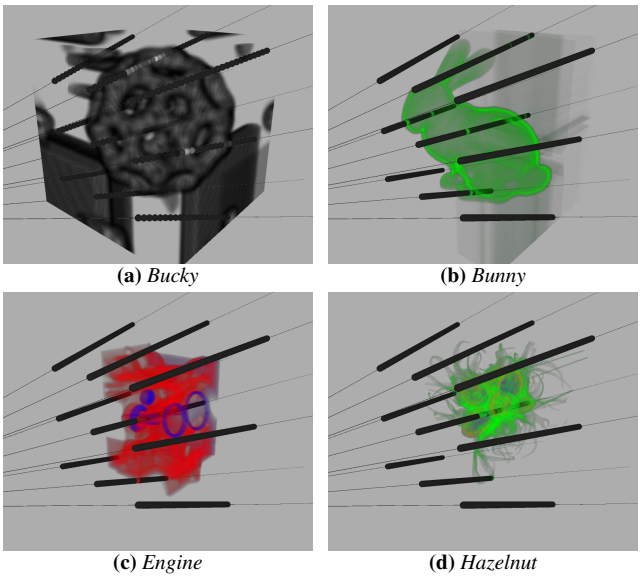
a point light source. Fig. 7 shows the impact of lighting on communicating shape and structure.

The compositing method is shown in the ray calculation window (Fig. 8). To view how the composited color is calculated, the user can click on any pixel in the Render Preview window. This will open the ray calculation window for that pixel. The exact formulas along with a description of the variables used can be viewed in this ray calculation window. If applicable, it will also display the early ray termination condition. A table showing the compositing being performed sample by sample is presented at the bottom of the window. Additionally, the tutorial walks the user through each of the compositing methods, explaining how they work and inviting the user to interact with their unique parameters, such as the density matching value for **First**.



$O_c = 1$ $O_c = 0.75$ $O_c = 0.5$ $O_c = 0.25$

Figure 9: *The effects of altering the opacity cutoff value O_c for early ray termination.*



(a) *Bucky* **(b)** *Bunny*

(c) *Engine* **(d)** *Hazelnut*

Figure 10: *Live preview of volumes in the scene view.*

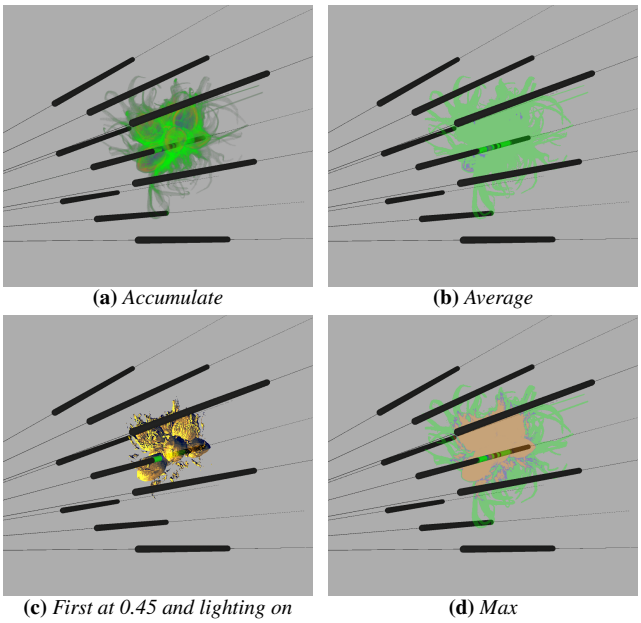
As indicated above, for **Accumulate** and **First** we employ early termination if set 'on', i.e., accumulate will stop if the opacity exceeds an opacity cutoff value, which can be set by the user through a slider in the range 0 to 1. Its effect on the resulting volume renderings is exemplified in Fig. 9.

4. Live Raycasting Extension

We now introduce the live preview feature to provide users with more immediate feedback as to the effect of the render settings that they are changing in VVRT. For this, live preview replaces the gray box—which represents the volume domain (see Sec. 3.1)—with the color-mapped volume (voxel grid) and renders it within the Unity scene view for display. For example, Fig. 10 shows different volumes and Fig. 11 compares compositing methods using live preview. The live preview rendering reflects what is generated and shown in the reference VVRT raycaster as discussed in Sec. 3, and needs to be generated at highly interactive rates.

4.1. GPU Raycasting

The gray box needs to be visually replaced by the volume with renderings generated in the same way as it is done by the Unity Raycaster and depicted in the VVRT visualization. However, the



(a) *Accumulate* **(b)** *Average*

(c) *First at 0.45 and lighting on* **(d)** *Max*

Figure 11: *Compositing method results on the Live Hazelnut.*

Unity Raycaster runs on the CPU and is only able to handle low-resolution images interactively. Note that this suffices for visualization purposes when showing rays and samples as well as the virtual screen as a large number of rays would induce significant visual clutter. Also, a few seconds are acceptable for generating a high-resolution image in this context as discussed in Sec. 3.1. In contrast, the live preview needs to run at interactive rates at high resolution in the scene view. For this, we port the same raycasting process to the GPU using a custom shader for the voxel grid object in Unity, as well as pass relevant data to the GPU (representing the volume with a 3D texture).

4.2. Blending with Geometry

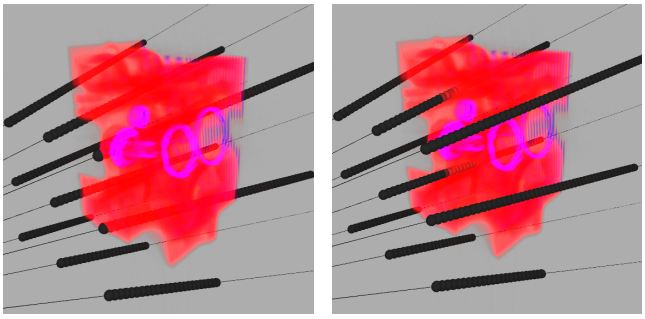


Figure 12: *Accumulate without (left) and with (right) depth testing.*

We aim to visualize the preview of the voxel grid along with the ray/sample geometry inside the volume which is generated by the Unity Raycaster such that they are blended correctly. In order for the user to see the Unity raycaster's rays and samples correctly blending in with the volume, the voxel grid shader needs to be aware of this geometry. Unity's built-in depth testing, however, is not able to interfere with the raycasting process itself and therefore produces incorrect results when objects are placed inside of the voxel grid (see Fig. 12). We thus supplement the process by accessing the main camera's depth texture and computing a vector between the depth of our ray's starting point and the depth of any geometry in the depth buffer. The length of this vector gives us the distance the ray can travel until it hits other geometry. We then place a condition in the raycasting sampling loop which terminates the ray if it exceeds the length of this depth vector. This lets the user see the ray and sample objects generated by the virtual camera together with a corresponding volume rendering (only for a different camera but otherwise the same settings) in the same view.

4.3. Synchronization

The synchronization of settings between the live preview raycasting shader and the Unity Raycaster ensures that changes made by the user in the graphical interface are immediately reflected in the live view. This synchronization is achieved through an event-driven system that updates shader parameters in real-time as users adjust settings like sampling distance, transfer functions, and compositing methods. This approach allows users to interactively tweak these parameters and see the impact on the rendered preview without needing to manually restart the rendering process.

Key parameters that are synchronized include:

- **Step Size**: Adjusting this setting updates the distance between samples along each ray, allowing users to balance between rendering precision and performance.
- **Transfer Function Parameters**: Any changes to the mapping of densities to color and opacity are directly reflected, enabling users to experiment with transfer function design.
- **Compositing Method**: Switching between different compositing approaches, such as accumulation or maximum intensity projection, is reflected instantly, giving users an immediate sense of how different methods affect the final visualization.

This real-time synchronization not only improves the user experience by making the application more responsive, but also serves an educational purpose by allowing users to understand the effects of various settings dynamically. For example, users can see how adjusting the step size impacts the level of detail in the rendered volume or how different transfer functions highlight specific features of the data. Crucially, this live preview is shown in one scene view along with our geometric representation of rays and samples corresponding to the virtual screen, reflecting user settings.

5. Evaluation

We first discuss the performance of VVRT in Sec. 5.1. For the basic version of VVRT (Sec. 3) as well as the live raycasting extension (Sec. 4) we conducted dedicated user studies to evaluate their effectiveness. All participants followed the full VVRT tutorial on their own computers without supervision; they were not compensated for their participation. We had participants who took the Scientific Visualization course, other computer science students, as well as members of the general public. The results are discussed in Sec. 5.2 and Sec. 5.3, respectively.

5.1. Performance

Interactive performance is important for VVRT. Frame rates were tested with the Bunny dataset at a sample distance corresponding to the size of a voxel, all other settings were left at their default values. Our measurements show that with modern hardware (recent Ryzen CPUs and an NVIDIA RTX 3080) we achieve the monitor's refresh rate (at 144 Hz and 360 Hz, respectively). While loading times can take up to several seconds (ranging from 0.11 s for Bucky to 3.45 s for Hazelnut on a Ryzen 5950x with an RTX 3080), it can still be considered fast enough for a user to do this during exploration.

Our only participant to report poor performance in our live raycasting user study (Sec. 5.3) used a configuration with an Intel T3400 (from 2009), 3GB RAM, an and NVIDIA GeForce 9300M GS (512MB) at a 1680×1050 resolution. This system is significantly less capable than the next-lowest hardware configuration reported with an Nvidia GTX 1060 that achieved high responsiveness. In our view, this is generally sufficient as we can reasonably expect our target audience of Scientific Visualisation students to have access to a system with a GPU with at least this performance.

5.2. User Study: Raycasting Process

We first evaluated VVRT as discussed in Sec. 3 (i.e, without the live raycasting view). We had two students from Scientific Visualization (who are already familiar with raycasting), one student from a computing science-related field, and four others.

5.2.1. Educational Questions

The overall feedback on the educational questions, see Table 1, is clearly positive. All previous students of the Scientific Visualization course think that the raycaster will be able to help future students. Additionally, all users found that they learned something new about raycasting from the application, except for students who were already experienced in raycasting. However, they believed that the application might be helpful to other people learning about the technique. This is in line with expectations since VVRT currently only covers the basics of volume raycasting.

The visualization of the raycasting process and the ability to control and experiment in the application were most appreciated. Less than half of the users found the tutorial and the ray calculation window useful. This suggests that there is potential to further improve on these aspects in future work.

Table 1: Raycasting Process*: educational questions and answers.*

If you followed the Scientific Visualization course, do you think the Raycaster application would be helpful to future students of the course?	
Yes	2
No	0
Did the application help you understand raycasting better?	
Yes	4
No, but I already understood raycasting well beforehand	3
No	0
Do you think the application can help other people understand raycasting better?	
Yes	7
No	0
Which of the following things do you think were successful in helping you understand raycasting?	
The text tutorial that you can follow at the bottom left	2
The visualization of the raycasting progress	6
Being able to experiment and try things out in the application	5
The ray calculation window with the breakdown of a single ray calculation	1

How would you rate these aspects of the application?

	Very good	Good	Neutral	Bad	Very Bad
Visuals	1	5	1	0	0
User Interface	3	4	0	0	0
Tutorial	3	3	1	0	0
Ray Calculation	3	5	1	0	0

5.2.2. Technical Questions

From the technical questions, see Table 2, we find a general consensus that VVRT is easy to use. There was no particular part of the application that stood out in this regard. Most participants agreed that the application had a good level of complexity, with some users finding it too simple. No one found the application too complex.

5.2.3. Suggestions, comments and feedback

Feedback was generally favorable; participants found the application to be "quite interesting" and a "great addition". We received numerous constructive suggestions regarding how VVRT can be improved. One participant suggested to "start by explaining the purpose and what it aims to achieve" to gain a better understanding of "what [they] are we working toward, and what is the end goal?". In addition, also the role of the render preview should be clarified early ("Render preview pixels only became clear to us in the final step"). Another participant remarked that the "5 parameters below the plot of the transfer function don't have labels. It's better to indicate what they're for exactly." and remarked that the Bucky being "such an abstract dataset [...] may not be a good initial choice". They further suggested that "you could make it even better by having a couple of "scenarios", where you tell the user how to set up the ray caster. E.g. only visualize the surface, or only visualize the most dense parts of a volume." Additional suggestions are clearer explanations of the goal of ray casting, as well as creating multiple levels where the user will aim to visualize different aspects of a dataset. Overall the comments were positive with ideas for future improvements and features (see Sec. 6 for further discussion). Different comments in this study also remarked on the voxel grid visualization ("visualize the voxel grid in some way, so you can see where the rays sample the volume."; "It would be nice to see the voxel grid (bucky, bunny, etc.) in some way inside of the unit cube"), a feature which we implemented with the with the live raycasting feature.

From the user study, we can conclude that the application successfully helped in increasing the understanding of raycasting. The visualization and ability to control and experiment in the application were most appreciated, while the ray calculation window and tutorial left the most room for improvement.

Table 2: Raycasting Process*: technical questions and answers.*

How would you rate these aspects of the application?

	Very good	Good	Neutral	Bad	Very Bad
Ease of Use	4	2	1	0	0

What did you think of the complexity of the application?	
Too simple, more settings and controls would be an improvement	1
Good	6
Too complex, there are too many unnecessary settings and controls	0

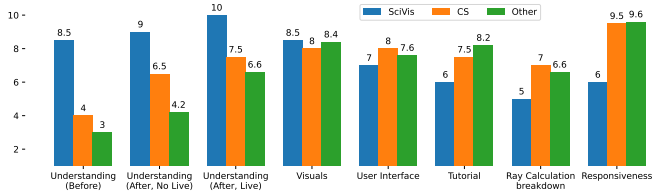


Figure 13: Live raycasting: *user ratings on their own understanding of raycasting and VVRT features on a scale from 1 (bad) to 10 (excellent).*

5.3. User Study: Live Raycasting

We conducted a second user study to evaluate the live raycasting feature (Sec. 4). The study had two students from the Scientific Visualization course, two participants from a computing science related field, and five who have no background in computing science.

Effectiveness of the Raycaster Application. Users were asked to assess their understanding of raycasting on a scale of 1 ('Never heard of it') to 10 ('Could pass an exam'). As shown in Fig. 13, understanding improved after using VVRT, with live preview leading to significantly better outcomes (mean values across all user groups: before: 4.9; after without live preview: 6.5; after with live preview: 7.8). Individually, as expected, the level of understanding increases with more technical background. Visuals were assessed similarly across all groups, whereas other aspects achieved a similarly high rating across CS and non-CS students, while the SciVis students—who have the most knowledge—rated them lower.

Impact of the Visual and Interactive Features. Participants were asked which parts of VVRT helped them understand raycasting. They could select any number of elements. The majority of respondents noted that the *visualization of the virtual camera* and the ability to *experiment within the application* were helpful in understanding raycasting. Specifically, these two elements were highlighted by almost all participants. The *live preview* was also mentioned frequently as a significant aid in comprehension, particularly by those with a lower initial understanding of raycasting.

User Satisfaction with Specific Features. Individual aspects of the application were rated from 1 ('unusable') to 10 ('excellent'). Ratings for features such as visuals, user interface, tutorial, and breakdowns are generally high, with most features rated between 6 and 10 by the users. The breakdown of ray calculations received slightly lower ratings compared to visuals and responsiveness, indicating potential areas for improvement in clarity or user interaction. The *text tutorial* was frequently mentioned, and the feedback suggests it was a valuable part of the learning experience.

Hardware and Responsiveness. Several users mentioned the performance of the application on their hardware, supplementing our measurements as reported in Sec. 5.1. While we achieved high performance in our measurements, users with old systems noted some slowdowns when using live preview (e.g., a user reported laggy performance with a GeForce 9300M, a mobile GPU released in 2008).

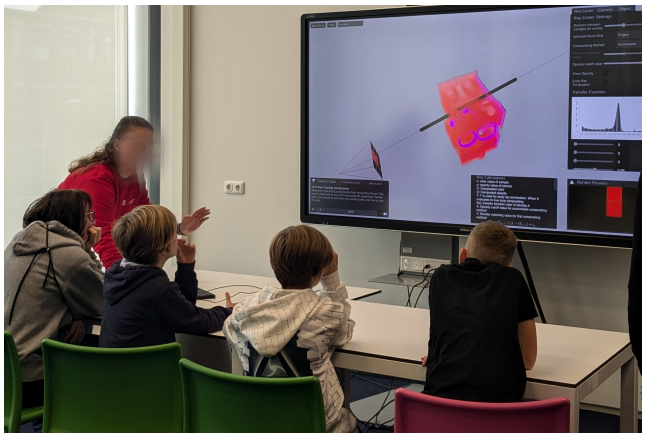


Figure 14: *VVRT at an annual free science festival for children of primary school age (`https://zpannendzernike.nl/`).*

General Feedback and Suggestions. Positive remarks included the utility of the live preview feature and the professional appearance of the application. Regarding potential improvements, it was mentioned that opacity computations could be explained more clearly and a Linux build would be appreciated (we only provided a Windows executable for the study).

6. Discussion

VVRT successfully achieves its main goals, but also exhibits various limitations in the current state, opening up several directions for improvement, as also indicated by the user studies. In Sec. 6.1 we cover pedagogical use cases and embedding in education, while in Sec. 6.2 we discuss VVRT's set of features and requirements.

6.1. Pedagogical Use Cases

Aims and Target Group. The goal of VVRT is to help users to better understand the steps of volume raycasting and their characteristics. Users can be members of the general public (e.g., in the context of exhibitions or science fairs) with the objectives to raise interest for visualization and to convey some basic principles. For example, we have presented VVRT in an annual free science festival for primary school pupils (Fig. 14). Users can also be students of a scientific visualization course. We plan to embed VVRT in the upcoming iteration of our course as discussed below in more detail.

Embedding in (Scientific) Visualization Course. The scientific visualization course at our university has the following learning outcomes and classifications according to Bloom's taxonomy [Blo56, Sha25] (*in italics*): (1) Understand the basic principles and fundamental methods of scientific visualization and the connection to related fields (*Understand*); (2) Implement scientific visualization techniques in a small team (*Apply and Create*); (3) Report about the implementations, in terms of algorithms developed, experimental results, and critical analysis (*Analyze and Evaluate*). (1) is covered by lectures, (2) and (3) with students developing their own visualization tool—integrating and combining techniques and

concepts—along the covered content in the lectures and an accompanying report, respectively.

VVRT directly supports (1), i.e., understanding by demonstrating aspects via interactive demonstration in the lectures, and providing VVRT to the students of the course to explore for themselves. The planned embedding to the course is to use VVRT interactively for illustration purposes during the lectures when discussing the fundamentals of volume raycasting. Then, to let students to explore VVRT themselves to deepen their understanding.

Tutorial for Visualization Students. For independent student exploration, we plan to design a tutorial that is closely aligned with the lecture, and potentially to integrate it further in the lab assignments. For this, setting up more levels going over the individual parts of raycasting with more guidance and examples would be required, with a dedicated level for every major step: volumetric data, taking samples, transfer functions, and compositing.

User Study Evaluation. The small sample size of our user study limits the extent to which our findings can be generalized. Additionally, we did not record detailed demographic information, limiting our ability to account for user-specific factors or biases in the interpretation of results. Moreover, the absence of a baseline for comparison makes it difficult to accurately measure the relative effectiveness or improvement offered by our application. Future research should expand on this, incorporating a larger participant group to enhance the robustness of the results. A more comprehensive evaluation is planned for the upcoming visualization course iteration, when VVRT will be fully integrated into the curriculum, enabling a deeper analysis of its impact on learning.

Extensibility. VRT, the basis for VVRT, is designed to not only serve visual computing-related courses at our University, but it is planned that is has a much broader impact. With VRT, a perspective goal is to contribute to teaching of visual computing topics on a national and international level. The educational tool is built with the cross-platform engine Unity, and we make it fully available to be extended and/or adjusted to fit the requirements of courses at other institutions or educational tutorials. VRT is joint work involving lecturers in the Scientific Visualization and Computer Graphics group (SVCG), and is also a good example of involving students work done in their BSc and MSc thesis projects.

6.2. Features and Requirements

Trilinear Interpolation Visualization. Trilinear interpolation is commonly performed to get the density of a sample from the voxel grid, but this is currently only explained in the tutorial text. A visualization of this step would improve user understanding of this concept. One could for example show the trilinear interpolation of a sample when it is selected. An alternative would be to employ a low-resolution voxel grid such that individual voxels are big enough to visualize this step in this context.

Acceleration Structures. Another valuable extension would be the visualization of acceleration structures, such as octrees, and how they are employed for empty space skipping for example. Presenting these structures (some of which are already present in VRT 2.0) within the raycaster would offer deeper insights into how spatial partitioning divides the volume into hierarchical regions. The existing work in VVRT on GPU-based raycasting and 3D texture sampling directly supports the creation of these interactive visualizations. With the current framework extended, users could see not only the rendered volumetric data but also how an octree recursively subdivides the voxel space into smaller sub-volumes. This feature should enable users to intuitively understand the role of acceleration structures in managing volumetric data, providing a visual and interactive way to explore how these structures affect ray traversal. Such an extension would make VVRT an even more complete educational tool for understanding both volume rendering and data structures that can be used accelerate it.

Performance. While our tests showed high performance with systems commonly used today, one user with an older systems experienced slowdowns when using the live preview (Sec. 5). Further optimizations could help with faster rendering of the live preview feature, in particular on older hardware. For example, pre-computed surface normals could be used for isosurface lighting, or empty space skipping could be employed (and its underlying procedure visualized to the user as discussed above).

Platforms VRT has been ported to virtual reality in its latest version [vWVdlHFK23], and VVRT could potentially benefit from virtual reality too. Currently, live raycasting is only supported on Windows, limiting accessibility for users on other platforms. Future work will focus on extending support to other operating systems to ensure broader usability. A web-based implementation could further enhance accessibility by enabling users to interact with VVRT directly in a browser without requiring installation. Additionally, mobile support for Android and iOS would allow for more flexible and portable use, making VVRT available in a wider range of learning environments.

7. Conclusion

We have introduced *Virtual Volume Raycaster* (VVRT), an interactive application that enables learners to view and explore the volume raycasting process in real-time. It builds on our previous work on Virtual Ray Tracer (VRT), an interactive environment for teaching ray tracing concepts that uses the cross-platform engine Unity. VVRT aims to help students of scientific visualization as well as the general public to gain a (better) understanding of the steps involved in volume raycasting and their characteristics. Learners are encouraged to play with and explore the impact of different settings, e.g., concerning the transfer function or ray sampling step size. We make VVRT fully available to be extended and/or adjusted to fit the requirements of courses at other institutions, educational tutorials, or of members of the general public. Two user studies showcase the utility and potential of VVRT in supporting the understanding and teaching of volume raycasting.

Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project Number 327154368 - SFB 1313.

References

[Ble24] BLESINGER P.: *Real-time Visualisation of Volume Raycasting in Virtual Ray Tracer*. BSc thesis, 2024. URL: https://fse.studenttheses.ub.rug.nl/34382/.

[Blo56] BLOOM B. S.: *Taxonomy of Educational Objectives: The Classification of Educational Goals*, 1st ed. Longman Group, 1956.

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. *SIGGRAPH Comput. Graph. 18*, 3 (jan 1984), 137–145.

[FTI86] FUJIMOTO A., TANAKA T., IWATA K.: Arts: Accelerated ray-tracing system. *IEEE Computer Graphics and Applications 6*, 4 (1986), 16–26.

[GFE*12] GRIBBLE C., FISHER J., EBY D., QUIGLEY E., LUDWIG G.: Ray tracing visualization toolkit. In *Proceedings of ACM SIGGRAPH* (2012), ACM, pp. 71–78.

[HAM19] HAINES E., AKENINE-MÖLLER T. (Eds.): *Ray Tracing Gems*. Apress, 2019. http://raytracinggems.com.

[MSW21] MARRS A., SHIRLEY P., WALD I. (Eds.): *Ray Tracing Gems II*. Apress, 2021. http://raytracinggems.com/rtg2.

[Nic12] NICHOLSON S.: Strategies for meaningful gamification: Concepts behind transformative play and participatory museums. *Meaningful Play 1999* (2012), 1–16.

[Rus99] RUSSELL J.: An interactive web-based ray tracing visualization tool. *Undergraduate Honors Program Senior Thesis. Department of Computer Science, University of Washington* (1999).

[RW15] REINERS T., WOOD L. C.: *Gamification in education and business*. Springer, Cham, Switzerland, 2015.

[SAH*16] SIMONS G., AMENT M., HERHOLZ S., DACHSBACHER C., EISEMANN M., EISEMANN E.: An Interactive Information Visualization Approach to Physically-Based Rendering. In *Vision, Modeling & Visualization* (2016), The Eurographics Association.

[Sha25] SHABATURA J.: Using bloom's taxonomy, 2025. Accessed: 2025-03-13. URL: https://tips.uark.edu/using-bloom s-taxonomy.

[SHP*19] SIMONS G., HERHOLZ S., PETITJEAN V., RAPP T., AMENT M., LENSCH H., DACHSBACHER C., EISEMANN M., EISEMANN E.: Applying visual analytics to physically based rendering. *Computer Graphics Forum 38*, 1 (2019), 197–208.

[SJL15] SPENCER B., JONES M., LIM I.: A visualization tool used to develop new photon mapping techniques. *Computer Graphics Forum 34*, 1 (2015), 127–140.

[SM09] SHIRLEY P., MARSCHNER S.: *Fundamentals of Computer Graphics*, 3rd ed. A. K. Peters, Ltd., USA, 2009.

[SSM02] SMYK M., SZABER M., MANTIUK R.: *JaTrac — an exercise in designing educational raytracer*. Springer, 2002, pp. 303–311.

[Tel14] TELEA A. C.: *Data Visualization: Principles and Practice, Second Edition*. A K Peters/CRC Press, Sept. 2014.

[VdlHWFK22] VERSCHOORE DE LA HOUSSAIJE W. A., WEZEL C. S. V., FREY S., KOSINKA J.: Virtual Ray Tracer. In *Eurographics 2022 - Education Papers* (2022), Bourdin J.-J., Paquette E., (Eds.), The Eurographics Association, pp. 45–52.

[VGV*20] VITSAS N., GKARAVELIS A., VASILAKIS A., VARDIS K., PAPAIOANNOU G.: Rayground: An Online Educational Tool for Ray Tracing. In *Eurographics 2020 - Education Papers* (2020), Romero M., Sousa Santos B., (Eds.), The Eurographics Association.

[vWVdlHFK23] VAN WEZEL C. S., VERSCHOORE DE LA HOUSSAIJE W. A., FREY S., KOSINKA J.: Virtual Ray Tracer 2.0. *Computers & Graphics 111* (Apr. 2023), 89–102. doi:10.1016/j.cag.2023.01.005.

[Wal23] WAL L. V. D.: *Virtual Ray Tracer: Ray Casting Support*. BSc thesis, 2023. URL: https://fse.studenttheses.ub.rug.nl/31739/.

[ZAD15] ZIRR T., AMENT M., DACHSBACHER C.: Visualization of coherent structures of light transport. *Computer Graphics Forum 34*, 3 (2015), 491–500.