Investigating the Apple Vision Pro Spatial Computing Platform for GPU-Based Volume Visualization

Camilla Hrycak* University of Duisburg-Essen,
David Lewakis† University of Duisburg-Essen
Jens Krüger‡ University of Duisburg-Essen

ABSTRACT

In this paper, we analyze the Apple Vision Pro hardware and the visionOS software platform, assessing their capabilities for volume rendering of structured grids—a prevalent technique across various applications. The Apple Vision Pro supports multiple display modes, from classical augmented reality (AR) using video see-through technology to immersive virtual reality (VR) environments that exclusively render virtual objects. These modes utilize different APIs and exhibit distinct capabilities. Our focus is on direct volume rendering, selected for its implementation challenges due to the native graphics APIs being predominantly oriented towards surface shading. Volume rendering is particularly vital in fields where AR and VR visualizations offer substantial benefits, such as in medicine and manufacturing. Despite its initial high cost, we anticipate that the Vision Pro will become more accessible and affordable over time, following Apple's track record of market expansion. As these devices become more prevalent, understanding how to effectively program and utilize them becomes increasingly important, offering significant opportunities for innovation and practical applications in various sectors.

Index Terms: Apple Vision Pro, Volume Rendering, Virtual Reality, Augmented Reality.

1 Introduction and Motivation

Virtual Reality (VR) and Augmented Reality (AR) trace back to Sutherland's seminal work in 1968 [29]. While significant progress was achieved in VR, it wasn't until the 1990s that the technology matured enough to be practical. In 1990, Caudell and Mizell [9] introduced the term 'Augmented Reality' to describe a heads-up display (HUD) system blending computer-generated imagery with reality. A significant milestone for widespread interest in immersive technology was the release of affordable VR devices in 2013, starting with the Oculus Rift [2], which sparked a surge of interest and research in VR and AR. Similarly, AR devices like Google Glass [1] have become more accessible to the public. While various VR and AR platforms have been developed, recent affordable devices predominantly employ inside-out tracking and operate on two primary principles for AR: optical see-through and video see-through, each offering distinct advantages and limitations.

Despite the success of professional applications and casual games like Beat Saber and Pokémon Go [7, 27], these platforms have not yet reached the widespread adoption of standard computers or mobile devices. This may be due to lingering hardware and software limitations that have prevented the emergence of a definitive "killer app," including challenges with display quality, tracking, and overall performance.

*e-mail: Camilla.Hrycak@uni-due.de
†e-mail: David.Lewakis@stud.uni-due.de
‡e-mail: Jens.Krueger@uni-due.de

In June 2023, Apple unveiled the Apple Vision Pro [3] system, positioned as a "spatial computing platform" supporting both AR and VR, thus establishing it as a comprehensive extended reality (XR) platform. The device became available in the US in February 2024 and is priced similarly to other professional AR devices like the Microsoft HoloLens 2 [26], in contrast to the more economical Meta Quest Pro [25]. What sets the Apple Vision Pro apart is not only its unique hardware features but also the reputation and market influence of Apple itself. Apple has a proven track record of transforming niche markets into profitable platforms, as seen with products like smartphones and smartwatches. This means that investing time in understanding Apple's hardware and software platforms can be valuable for researchers. While a doctor may not care about the brand of the headset used in a virtual reality-aided procedure, the longevity and widespread adoption of Apple's platforms can make learning to fully utilize the potential of these devices a worthwhile investment for data and visualization scientists.

This short paper provides early insights gained from implementing a testbed for VR and AR direct volume rendering on the Apple Vision Pro spatial computing platform. We chose direct volume rendering due to its inherent challenges on VR and AR devices, where modern graphics APIs typically prioritize surface rendering. From a hardware and performance perspective, the raster-bound nature of direct volume rendering presents intriguing complexities, particularly in environments with dual high-resolution screens under severe power and heat dissipation constraints.

2 Related Work

AR and VR are primarily utilized in gaming, collaborative environments, and scientific visualization, with medical research and clinical practice being the most common usage scenarios within the visualization domain. Although volumetric effects greatly enhance immersion in gaming and collaboration, performance constraints often lead to their approximation using techniques like imposters. Therefore, our review primarily focuses on medical applications.

Many areas of medicine today utilize data from imaging techniques, like CT or MRI, for diagnostics and for engaging patients in their own care processes. These volumetric data are best represented in a 3D environment, which not only conveys information more effectively but also preserves the context of the visual data. Additionally, 3D representation allows for intuitive interaction, unlike traditional 2D screens.

This data representation is useful not only for diagnosis but also in medical education and the planning of surgeries and therapies. For instance, Magdics et al. [23] developed a VR Nasal Cavity Education Tool using Volume Rendering to depict the human nasal cavity within the context of the entire head, employing DVR with spatially varying parameters to highlight the size and location of the nasal cavity relative to the skull.

Mayer et al. [24] introduced an educational VR application for teaching human anatomy to medical students, allowing them to virtually dissect the visible human. Meanwhile, Zhang et al. [34] presented a VR application aimed at training surgeons about potential natural physical variations in patients, which could enhance the

quality of surgeries. Additionally, Cao et al. [8] used DVR to model patient brains from MRI data for planning minimally invasive surgeries.

Furthermore, numerous studies, such as those by Taibo and Iglesias-Guitian [30], have focused on improving volume rendering for VR systems. Their work "Immersive 3D Medical Visualization in Virtual Reality using Stereoscopic Volumetric Path Tracing" employs aspects of a DVR-based approach (Monte Carlo volumetric path tracing) to improve cinematic rendering processes, reducing computational demands and enhancing effectiveness for interactive real-time applications.

Lastly, Valls-Esteve et al. [32] conducted a comparative study of advanced 3D imaging techniques for pediatric tumor surgeries, emphasizing the importance of volume rendering in transforming 2D radiological images into 3D visualizations that provide surgeons with a better understanding of complex anatomies and tumor locations.

Building on these advancements in medical VR applications, the release of the Apple Vision Pro in Febrary 2024 has introduced new prospects. Though initially available only in a limited release of estimated 100,000-200,000 devices (according to various internet sources) and only in the US, it has already expanded the possibilities for medical and educational VR applications. Its high-performance features support a range of specialized apps, tailored for professional training, surgical planning, and enhanced communication between healthcare professionals and patients [4].

Among these applications, the myMako app by Stryker [15] for surgical planning and the Cinematic Reality app [16] both leverage the immersive capabilities of the Vision Pro to provide detailed anatomical studies and interactive surgical simulations. Similarly, the Complex HeartX app by Elsevier [11] uses the platform to educate both medical professionals and laypersons about cardiovascular health and diseases, improving patient and family understanding of medical conditions through interactive experiences.

These applications demonstrate the potential of volume rendering technologies when combined with cutting-edge XR hardware like the Apple Vision Pro.

In summary, the integration of cutting-edge XR technologies like the Apple Vision Pro with sophisticated volume rendering applications marks a significant leap forward in the way medical professionals can plan surgeries, educate students, and communicate with patients, ultimately enhancing the quality and efficacy of medical care.

3 The Apple Vision Pro Platform

In this section, we explore the key components of the Apple Vision Pro, starting with its hardware specifications and subsequently examining the software ecosystem that supports effective programming and operation of the device. This analysis emphasizes the APIs that are crucial for direct volume rendering.

3.1 Hardware

The Apple Vision Pro's hardware can be divided into three primary categories: the computing platform, the displays, and the tracking hardware. On the logic board of the Apple Vision Pro, most of the space is dedicated to two main integrated circuits (ICs): the M2 SoC and the R1 co-processor.

The R1 co-processor is dedicated exclusively to managing the computational demands of tracking. This arrangement ensures that tracking operations are executed with high efficiency, akin to a standalone tracking system, while minimally impacting the main CPU's resources. As a result, we can assume that the bulk of the CPU and GPU capacities remain available for other tasks, comparable to the resource availability in a desktop workstation not engaged in tracking. Our profiling results underscore this assumption as we do not see any significant time spent by the system on tracking.

The M2 SoC in the Apple Vision Pro features a configuration similar to that of a lower-end MacBook Pro, with an 8-core M2 processor—comprising four performance and four efficiency cores—supplemented by 10 GPU cores and a 16-core neural engine. It is equipped with 16 GB of unified memory and offers up to 1 TB of SSD storage [3]. Preliminary performance comparisons using Geekbench 6 yield scores of 2451/8294/39763 (Single-/Multicore/GPU), which are comparable to those of a current Mac Mini M2, which scores 2683/10094/46522.

The Vision Pro utilizes two Micro-OLED screens, each providing a resolution of approximately 3660 × 3200 pixels. Although edge trimming slightly reduces the effective pixel count, the resolution remains comparable to dual 4K displays [18]. This high-resolution setup poses a significant challenge for direct volume rendering, as the rendering performance is proportional to the number of pixels covered by the volume on screen. To address this, the Vision Pro leverages built-in eye-tracking hardware to support foveated rendering, which dynamically adjusts the rendering resolution based on the viewer's gaze location, thus optimizing processing resources. The performance implications of using foveated rendering are further explored in Sections 4 and 5.

3.2 Software

The Apple Vision Pro operates on visionOS, an evolution of Apple's iOS that brings essential augmented reality (AR) capabilities, primarily through the RealityKit and ARKit APIs. RealityKit, acting as Apple's 3D rendering engine, leverages ARKit's tracking data to fuse virtual and real-world elements seamlessly. Notably, RealityKit uses the industry-standard MaterialX [28] for scene description, supporting imports from Unity [31], creations with the *Reality Composer Pr* tool, or programmatic generation.

However, any version up to the current visionOS release version 1.3 impose several limitations not present in iOS's AR capabilities. For instance, Custom Systems/Components and TextureResource.DrawableQueue are currently unsupported in the RealityRenderer [6]. The ramifications of these limitations on direct volume rendering are discussed in Section 4.

3.2.1 Spaces

The Apple Vision Pro supports diverse modes for application presentation, called *spaces*. Applications can operate within a shared space alongside other apps and the environment, utilizing designated areas as interactive *windows* similar to those on traditional 2D desktops, or as *volumes* within the communal space. This space integrates computer-generated scenes with live feeds from cameras and depth sensors, representing the AR mode of the Apple Vision Pro. Users can dynamically reposition elements within this shared space, enhancing interaction and engagement. Within the shared space an application can also create a mixed immersive space which allows the app to break out of predefined boundaries. For fully immersive experiences, applications can occupy the entire visual field, embodying a typical VR setup where the entire scene is computer-generated with no real-world objects visible. In shared spaces, applications must utilize the RealityKit Framework for presentation. In contrast, those in isolated environments can directly access the graphics hardware via the Metal API, managing output to dual framebuffers. Apple recommends starting applications in a shared space and transitioning to a fully immersive space as needed.

3.2.2 Programming Language

The primary programming language for Apple Vision Pro's high-level APIs is Swift, designed to streamline the development of new components. Apple also provides C++ wrappers to facilitate integration with existing codebases, which is particularly useful for data preprocessing and loading tasks. While it is possible to maintain significant portions of legacy C++ code, converting rendering

routines to Swift is recommended. This conversion simplifies code management by avoiding the complexities introduced by wrapper functions, which impose Swift-like paradigms on C++ code, particularly in memory management. Ultimately, the decision whether to convert rendering code to Swift largely depends on the personal preference of the programmers and managers. However, after implementing a version of the renderer using the C++ wrappers and native Swift code, we would recommend the latter. The aforementioned Swift/Objective-C paradigms in the wrapper classes cause the C++ code to resemble Swift code to a significant extent.

4 Volume Rendering

Volume rendering techniques are broadly classified into two categories: indirect volume rendering and direct volume rendering (DVR) [19]. Indirect volume rendering involves computing a polygonal surface representation of the volume, such as an iso-surface, which is then rendered as a triangulated mesh. With the Apple Vision Pro's native support for triangle mesh rendering through rasterization and raytracing, handling iso-surfaces becomes straightforward. If the iso-surface is static and can be precomputed, no additional coding is needed; the mesh can be generated by an external tool, such as a marching cubes algorithm [22], and directly imported into the RealityComposer Pro software using various industry-standard mesh formats. These formats may include detailed attributes such as surface properties, textures, and animation parameters, which simplify the rendering of iso-surfaces and other visualizations based on simple geometric primitives. Moreover, the RealityKit framework automatically calculates light interactions with the real-world environment, including shadows, thereby enhancing spatial understanding. While indirect volume rendering is straightforward and efficient, it may not be suitable for all applications because it reduces the volume to a single surface, limiting the ability to inspect the data comprehensively. Consequently, our discussion primarily focuses on direct volume rendering techniques.

4.1 Direct Volume Rendering

Over the last few decades, several DVR techniques have been developed. Among these, ray-casting stands out as the most conceptually straightforward implementation of the volume rendering equation [20]. Ray-casting has been implemented on both CPUs [33] and GPUs [21], with most recent scalable implementations based on the ray-guided rendering algorithm [17, 10, 12, 14]. Before the widespread adoption of ray-casting for DVR, slice-based volume rendering was the dominant method. In this approach, the volume is sliced along major axes or perpendicular to the viewing direction, and these planar slices are composited in a back-to-front or front-to-back order. The advantage of slice-based rendering is that it can directly map the volume rendering process to the rendering of textured translucent triangles without the need for sophisticated shader execution. In cases of axis-aligned slicing, not even 3D textures are required. However, the image quality of slice-based rendering is generally inferior to that of ray-casting, and it lacks the capability to incorporate sophisticated acceleration techniques. For more details and a thorough discussion of the advantages and disadvantages of these approaches, we refer the readers to the introductory book *Real-Time Volume Graphics* [13].

4.2 Fully Immersive Space DVR

In most scenarios, direct volume rendering on recent graphics hardware typically employs a GPU-based ray-casting approach. Known for its straightforward implementation, this method consistently yields high-quality results and becomes highly efficient when augmented with acceleration techniques. The GPU in the M2 Chip, which features 1280 ALUs and a maximum FP32 performance of 3.6 TFLOPs, supports such advanced capabilities. These features are accessible through Metal API Version 3, which aligns well with standards like DirectX 12 and Vulkan 1.3, thus facilitating the transition of ray-casting implementations from desktop systems to the Apple Vision Pro.

In our testbed, we developed a basic ray-caster that supports key functionalities essential for most applications, including volume transformations, a 1D smoothstep transfer function, real-time gradient computation for lighting effects, and axis-aligned clipping planes. Figure 1 displays the fragment shader code for the ray-caster, written in Metal Shading Language. Notably, the parameter `amplification_id` indicates whether parameters are sourced from the buffer for the left or the right eye, marking the primary adaptation from a standard desktop GPU ray-casting implementation. We deploy this implementation in the fully immersive space of the Apple Vision Pro's VR mode, where we have complete and unrestricted access to the GPU. Similar strategies are adopted in existing commercial applications such as Siemens' Cinematic Reality [16].

```
half4 fragment fragmentMain(v2f in [[stage_in]],
                            ushort amp_id [[amplification_id]],
                            texture3d< half, access::sample > volume [[texture(0)]],
                            device const ParamsArray& renderArray [[buffer(0)]])
{

    ShaderRenderParamaters renderParams = renderArray.params[amp_id];
    constexpr sampler s( address::clamp_to_border, filter::linear );
    float3 voxelCount = float3(volume.get_width(), volume.get_height(), volume.get_depth());

    float3 rayDirectionInTextureSpace =
        normalize(in.entryPoint-renderParams.cameraPosInTextureSpace);

    // compute delta
    float samples = dot(abs(rayDirectionInTextureSpace),voxelCount);
    float opacityCorrection = baseline/(samples*renderParams.oversampling);
    float3 delta = rayDirectionInTextureSpace/(samples*renderParams.oversampling);

    float3 currentPoint = in.entryPoint;
    float4 result = 0.0;
    do {
        float volumeValue = volume.sample( s, currentPoint ).r;
        currentPoint += delta;
        float4 current = transferFunction(volumeValue, renderParams);
        current.a = 1.0 - pow(1.0 - current.a, opacityCorrection);
        result = under(current, result);
        if (result.a > 0.95) break;
    } while (inBounds(currentPoint,renderParams));

    return half4( result );
}
```

Figure 1: The core fragment shader of our testbed GPU-based ray-caster.

4.3 Shared Space DVR

In the shared space, corresponding to the AR mode of the Apple Vision Pro, unique challenges arise. The RealityKit API, the only available API in this mode, permits access to programmable shaders solely through the CustomMaterial-node. However, this node is not yet available on the Apple Vision Pro, which restricts the use of programmable shaders and 3D textures. Consequently, developers must consider alternative rendering strategies for implementing volume rendering in an AR setting. The options include: (1) omitting this mode for volume rendering, (2) utilizing CPU-based volume rendering, (3) employing indirect volume rendering, or (4) reverting to slice-based rendering. Omitting this mode is often not feasible, and CPU-based rendering may be too slow, given the device's four efficiency cores and dual 4K resolution. Therefore, the remaining practical alternatives are indirect volume rendering, which, although straightforward, is limited to rendering a small number of isosurfaces and requires time-consuming depth sorting for transparent isosurfaces, or slice-based rendering. Given the lack of 3D texture support, developers might resort to texture atlases or implement axis-aligned slice-based rendering, which are significant restrictions of the current visionOS version and a critical takeaway.

Table 1: Performance of Different Volume Rendering Techniques Across Various Datasets. All values are the average milliseconds for end-to-end frame rendering time. For the Fully immersive ray-caster we show the effects of early ray termination (ERT), illumination and foveation.

			Ray-Caster							
			Foveation Enabled				Foveation Disabled			
Dataset Name	Size	Slice based	ERT On	ERT On Illumination	ERT Off	ERT Off Illumination	ERT On	ERT On, Illumination	ERT Off	ERT Off Illumination
C60	64^3	16.6	11.7	12.8	12.4	12.4	11.5	11.1	11.8	13.4
Bonsai	128^3	22.1	14.2	12.3	14.3	13.3	11.4	11.9	11.6	12.8
Foot	256^3	47.9	13.2	12.9	13.4	15.9	13.2	13.5	13.5	13.6
Visible Human Head	512^3	90.5	15.5	16.1	17.7	36.3	14.5	16.1	17.9	43.5

4.3.1 Shader Graph Limitations and Workarounds

The absence of programmable shaders, or more specifically the CustomMaterial node, on the Apple Vision Pro confines users to using Shader Graphs. However, this limitation does not entirely eliminate functionality. We successfully implemented several commonly used volume rendering features in Shader Graph notation, such as smoothstep transfer functions, clipping volumes, and opacity correction. Figure 2 bottom illustrates the Shader Graph version of the smoothstep transfer function, which is similar to the implementation used in the Metal Shader Language, as shown in Figure 2 top.

Implementing a full ray-caster, however, appears unfeasible due to the absence of 3D textures and the fact that Shader Graphs do not support loops, which are essential for a ray-caster. Consequently, we opted to implement direct volume rendering (DVR) using axis-aligned slicing.

```
float4 transferFunction(float v, ShaderRenderParamaters p) {
    float t  = clamp((v - p.smoothStepStart)/(p.smoothStepShift), 0.0, 1.0);
    return float4(float3(v), t * t * (3-2*t));
}
```

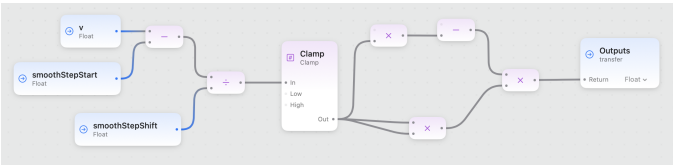

Figure 2: The smoothstep transfer function. The top image displays the implementation in Metal Shading Language as used by the ray-caster, while the bottom graph demonstrates the same functionality in Reality Composer Pro for the slice-based renderer.

5 Results

Outlined in Section 4, our application supports two distinct code paths tailored to specific operational contexts—either in a shared space or in a fully immersive environment.

In the shared space, we implemented an axis-aligned slice-based volume renderer utilizing a simple 1D sigmoid transfer function, commonly used in various imaging applications. In contrast, the fully immersive space (VR) employs a Metal-based GPU ray-caster. This ray-caster not only supports the same 1D transfer functions but also enhances performance and visual quality through early ray termination and local Phong illumination, with normals computed on-the-fly. Table 1 summarizes the performance metrics of our system, with timings averaged over a period of 10 seconds. The dataset was centrally positioned on the screen with the user's gaze fixed at the center of the dataset.

The table reveals that the performance of slice-based rendering halves each time the resolution doubles in each dimension. This is expected as performance is predominantly limited by the number of slices rendered, rather than the textures applied to each slice. Ray-casting timings indicate that early ray termination significantly boosts performance. Additionally, despite necessitating six additional texture fetches per sampling step, the on-the-fly computation of gradients introduces minimal overhead, indicating highly efficient GPU caching.

We evaluated our system using several well known datasets, ranging in size from 64^3 to 512^3. We selected a scenario from where the volume covers approximately 20% of the screen. Tests in the fully immersive mode were conducted with both lighting enabled and disabled, as well as with the automatic foveation feature toggled on and off, to assess its impact on performance and visual quality.

6 Conclusion

In this paper, we have demonstrated the Apple Vision Pro's capability for high-performance direct volume rendering in fully immersive spaces using a GPU-based ray-caster. Our results show that the M2 GPU efficiently executes complex ray-casting shaders and benefits significantly from acceleration methods. This capability allows for visualizations that rival those of traditional graphics workstations, although currently limited to fully immersive (VR) environments. The full potential of programmable shaders remains inaccessible in shared spaces (AR), where fallback options such as axis-aligned slicing lead to substantial performance and flexibility trade-offs. Given that visionOS shares the RealityKit API with iOS, apart from the absence of the CustomMaterial-Node in the ShaderGraph, we consider this limitation likely temporary and expect it to be resolved in future updates.

Developers currently interested in AR applications that require volume rendering should consider exploring indirect volume rendering techniques, such as triangulated iso-surfaces, or begin their development on iOS and transition to visionOS once support for the CustomMaterial-Node is established. The migration path from iOS to visionOS is straightforward, enabling a seamless transition once updates are implemented.

An additional point of interest is the impact of foveation. In our observations, significant performance gains were only noted when datasets were positioned in the periphery, making this feature potentially useful for applications that display multiple datasets simultaneously. However, for a single dataset, the volume will predominantly reside within the focus region, resulting in minimal to no performance gains.

Another promising aspect of Apple Silicon hardware is the neural engine. Although we did not explore it in detail in this short paper, the neural engine has the potential to significantly enhance performance in volume visualization. For instance, it could be utilized to implement super-resolution methods, thereby achieving higher performance and more detailed visualizations.

Moreover, it is important to note that the design of the Apple Vision Pro's casing currently limits its utility in clinical settings, as it cannot be disinfected [5]. This limitation is unexpected, especially considering Apple's marketing of the Vision Pro for medical AR applications. Nonetheless, we, along with Apple customer support, anticipate that third-party vendors will soon provide easily disinfected protective cases to address this issue.

REFERENCES

[1] Google Glass. https://www.google.com/glass/. Accessed: April 28, 2024. 1

[2] M. Abrash and B. Iribe. The oculus rift: A virtual reality headset for 3d gaming. *SIGGRAPH '12 Talks*, 2012. doi: 10.1145/2343483.2343491 1

[3] Apple Inc. Apple Vision Pro. https://www.apple.com/apple-vision-pro/, 2023. Accessed: April 28, 2024. 1, 2

[4] Apple Inc. Apple Vision Pro unlocks new opportunities for health app developers. https://tinyurl.com/yfz9am9z, 2024. Accessed: May 01, 2024. 2

[5] Apple Inc. How to clean apple vision pro and accessories. Online: https://support.apple.com/en-us/119574, 2024. Accessed: April 28, 2024. 4

[6] Apple Inc. Realitykit framework. Online: https://developer.apple.com/documentation/realitykit/, 2024. Accessed: April 28, 2024. 2

[7] Beat Games. Beat Saber. https://beatsaber.com/, 2018. Accessed: April 28, 2024. 1

[8] Y. Cao, W. Zhang, and J. Fu. Brain Modeling Guided by Direct Volume Rendering. In *Proceedings of the 2021 13th International Conference on Bioinformatics and Biomedical Technology*, ICBBT '21, pp. 15–21. Association for Computing Machinery, New York, NY, USA, 2021. event-place: Xi'an, China. doi: 10.1145/3473258.3473261 2

[9] T. Caudell and D. Mizell. Augmented reality: an application of heads-up display technology to manual manufacturing processes. In *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, vol. ii, pp. 659–669 vol.2, Jan 1992. doi: 10.1109/HICSS.1992.183317 1

[10] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann. Gigavoxels: ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09, pp. 15–22. Association for Computing Machinery, New York, NY, USA, 2009. doi: 10.1145/1507149.1507152 3

[11] Elsevier. Complete heartx - an immersive journey through the heart. https://www.elsevier.com/products/complete-heartx, 2024. Accessed: 2024-05-01. 2

[12] K. Engel. Cera-tvr: A framework for interactive high-quality teravoxel volume visualization on standard pcs. In *2011 IEEE Symposium on Large Data Analysis and Visualization*, pp. 123–124. IEEE, 2011. 3

[13] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-Time Volume Graphics*. Ak Peters Series. A K Peters/CRC Press, 2006. 3

[14] T. Fogal, A. Schiewe, and J. Krüger. An analysis of scalable gpu-based ray-guided volume rendering. In *2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)*, pp. 43–51, Oct 2013. doi: 10.1109/LDAV.2013.6675157 3

[15] S. J. R. Gauss Surgical, Inc. mymako - the mako digital experience. https://apps.apple.com/us/app/mymako/id6450731605, 2024. Accessed: 2024-05-01. 2

[16] S. H. GmbH. Cinematic reality. https://apps.apple.com/us/app/cinematic-reality/id6474022077, 2023. Accessed: 2023-04-30. 2, 3

[17] M. Hadwiger, J. Beyer, W.-K. Jeong, and H. Pfister. Interactive volume exploration of petascale microscopy data streams using a visualization-driven virtual memory approach. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2285–2294, Dec 2012. doi: 10.1109/TVCG.2012.240 3

[18] iFixit. Vision pro teardown part 2: What's the display resolution? https://tinyurl.com/ifixitVisionPro, 2023. Accessed: April 28, 2024. 2

[19] C. Johnson and C. Hansen. *Visualization Handbook*. Academic Press, Inc., USA, 2004. 3

[20] J. T. Kajiya and B. P. Von Herzen. Ray tracing volume densities. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pp. 165–174. Association for Computing Machinery, New York, NY, USA, 1984. doi: 10.1145/800031.808594 3

[21] J. Krüger and R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings IEEE Visualization 2003*, 2003. 3

[22] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, aug 1987. doi: 10.1145/37402.37422 3

[23] M. Magdics, D. White, and S. Marks. Extending a virtual reality nasal cavity education tool with volume rendering. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pp. 811–814, Dec 2018. doi: 10.1109/TALE.2018.8615248 1

[24] L. Mayer, M. Sünchsen, S. Reinhold, and M. Teistler. vimilab – teaching medical imaging and radiological anatomy in a virtual reality environment. In *2023 IEEE 11th International Conference on Serious Games and Applications for Health (SeGAH)*, pp. 1–5, Aug 2023. doi: 10.1109/SeGAH57547.2023.10253787 1

[25] META. Meta Quest Pro. https://www.meta.com/de/quest/quest-pro/, 2024. Accessed: April 28, 2024. 1

[26] Microsoft. Microsoft HoloLens 2. https://www.microsoft.com/en-us/hololens, 2019. Accessed: April 28, 2024. 1

[27] Niantic. Pokémon GO. https://pokemongolive.com/en/, 2016. Accessed: April 28, 2024. 1

[28] D. Smythe, J. Stone, L. Gritz, G. Quaroni, and N. Harrysson. Materialx: An open standard for network- based cg object looks. ACM SIGGRAPH 2017 BOF, 2017. 2

[29] I. E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), pp. 757–764. Association for Computing Machinery, New York, NY, USA, 1968. doi: 10.1145/1476589.1476686 1

[30] J. Taibo and J. A. Iglesias-Guitian. Immersive 3d medical visualization in virtual reality using stereoscopic volumetric path tracing. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pp. 1044–1053, March 2024. doi: 10.1109/VR58804.2024.00123 2

[31] Unity Technologies. Unity. https://unity.com/, 2023. Game development platform. 2

[32] A. Valls-Esteve, N. Adell-Gómez, A. Pasten, I. Barber, J. Munuera, and L. Krauel. Exploring the potential of three-dimensional imaging, printing, and modeling in pediatric surgical oncology: A new era of precision surgery. *Children (Basel)*, 10(5), May 2023. doi: 10.3390/children10050832 2

[33] I. Wald, G. Johnson, J. Amstutz, C. Brownlee, A. Knoll, J. Jeffers, J. Günther, and P. Navratil. Ospray - a cpu ray tracing framework for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):931–940, Jan 2017. doi: 10.1109/TVCG.2016.2599041 3

[34] N. Zhang, H. Wang, T. Huang, X. Zhang, and H. Liao. A vr environment for human anatomical variation education: Modeling, visualization and interaction. *IEEE Transactions on Learning Technologies*, 17:391–403, 2024. doi: 10.1109/TLT.2022.3227100 1