Massive Data Visualization Techniques for use in Virtual Reality Devices

Jason A. Ortiz*
Argonne National Laboratory
University of Central Florida

Joseph A. Insley
Argonne National Laboratory
Northern Illinois University

Janet Knowles
Argonne National Laboratory

Victor A. Mateevitsi
Argonne National Laboratory

Michael E. Papka
Argonne National Laboratory
Northern Illinois University

Silvio Rizzi
Argonne National Laboratory

Abstract

Scientific simulations executed on supercomputers produce massive amounts of data. Visualizing this data is essential to discovery and dissemination, but methods for transforming and displaying such large data visualizations for use in Extended Reality (XR) devices are not commonly supported. We investigated the viability of existing XR applications (i.e., ParaView VR, SummitVR, and Omniverse XR) to display large data visualizations. Our investigations led us to create a proof-of-concept Virtual Reality (VR) application with Unity using Universal Scene Description (USD) files exported from Houdini to display and interact with large time-varying scientific data visualizations. We present our investigations as a basis for future work to display and interact with scientific data visualizations in XR.

1 Introduction

High Performance Computing (HPC) applications executing scientific simulations can produce massive amounts of data traditionally visualized with tools like ParaView and displayed on 2D monitors. However, common visualization tools cannot directly import or display such large data in its entirety, which leads to missed discoveries and opportunities. Exploring and manipulating multidimensional data with immersive Extended Reality (XR) technology, an umbrella term for Augmented Reality (AR), Virtual Reality (VR), and Mixed Reality (MR), can provide improved understanding and analysis of multidimensional data via natural interactions and gestures to move oneself and to rotate, translate, and scale virtual objects [1]. XR also affords an intuitive sense of copresence between multiple users sharing a virtual experience in-person or remotely, thus enabling improved collaboration scenarios [2].

We discuss existing XR applications that allow for scientific data analysis and visualization and their data size limitations, and we present the intermediate results of a proof-of-concept workflow and Unity VR application for large scientific data visualization and interaction.

2 Related Work

Since 2016, ParaView, a scientific data analysis and visualization tool, has provided support for data exploration using VR headsets [3]. The ParaView VR plugin allows users to view ParaView visualizations and edit the corresponding filters just as they would in the desktop GUI editor. SciVista SummitVR [4] has also introduced a ParaView module which allows for ParaView state file import, thus allowing multiple users to rotate, translate, and scale the same ParaView data together. Both applications have promising potential for scientific data discovery and dissemination, but we faced data size limitations when attempting to display our visualizations. Smaller datasets displayed adequately, but for large data, some meshes would not appear and animations would not play smoothly.

Similarly, Nvidia Omniverse, an "... easily extensible platform for 3D design collaboration and scalable multi-GPU, real-time, true-to-reality simulation", provides Apps and Connectors for common 3D tools such as Autodesk Maya, Epic Games Unreal Engine, and ParaView [5]. Omniverse allows for consistent interoperability between these applications because 3D scene and asset information is stored in Universal Scene Description (USD) files. These files can be imported, exported, and live-synced between applications via Omniverse Connectors. For example, Omniverse XR allows users to view and move around ParaView visualizations as USD scenes with AR or VR devices. Omniverse is not compatible with all types of computer networks and environments however, but it shows that the USD file format can provide a viable way to reliably store massive 3D data. As discussed in [6], "USD is an effective data format for encoding scientific data in preparation for cinematic visualization," which encouraged us to investigate viable workflows to transmit scientific data to XR devices.

3 Methodology

Figure 1 shows a proposed workflow for moving scientific data into the Unity game engine [7], which was chosen for its ease of use and existing packages allowing for XR interactions and USD file integration. After a simulation is executed and results are output,

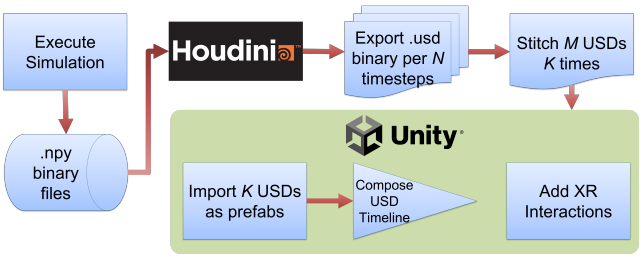

Figure 1: Workflow to create scientific data visualization XR app.

a user can import that data into SideFX Houdini [8] for example by using a custom Python script to read and transform the raw data as necessary. Our example transforms two 108 MB NumPy binary files to display the activated neurons of a neural network training on LIGO gravitational wave datasets [9]. For time-varying data, a user then exports N binary USD files for each animation frame via the USD Render Node. In our example, $N = 300$ and each file is approximately 276 MB. Next, a user executes the `usdstitch` [10] Python script provided by Pixar to combine M different USD files into one new binary USD file K times where $K = \frac{N}{M}$. In our example, $M = 75$, $K = 4$, and each of the four USD files are large at 18 GB. Each stitched USD file has $M = 75$ merged USD `timeSamples` representing 75 frames. Attempting to stitch more than 75 files failed due to exceeding memory limits.

*e-mail: jason.ortiz@knights.ucf.edu

3.1 Development in Unity

Using Unity version 2020.3.26f1 and Unity's USD preview package, version 3.0.0-exp.2, we import each stitched USD binary as a Unity prefab, which bundles all the USD assets and properties into one manageable Unity `GameObject`. Each prefab is an 89 MB metadata object that allows a Unity developer to inspect its USD file information within the Unity editor and work indirectly with the USD mesh data that is stored on local disk. Next, we create a `Timeline GameObject` and add two tracks for each USD prefab: one for playing the USD `timeSamples` and one for listing the USD prefab as active. This format allows us to show and play the animations for only one USD prefab at a time.

Next, using Unity's XR Interaction Toolkit package, version 2.0.2, we can enable XR device support for common XR devices such as the Meta Quest 2. We use the toolkit's scripts to make the USD prefabs in the scene interactable with XR rays which project from XR controllers. We also add a custom C# script with a callback function to play or pause the animation timeline within the `PlayableDirector GameObject` when the Right XR Controller emits a `Select` action on the interactable USD prefab, which is triggered on right-grip press.

We then play the application with a Meta Quest 2 tethered by a Quest Link USB-C cable to a desktop PC equipped with an Nvidia RTX 3080 GPU, 32 GB of RAM, and 10 GB of dedicated video memory.

4 Results

While playing the application, a user can press the right-grip button of the XR Controller while pointing at the prefab representing the data as seen in Figure 2. When the animation is paused, the measured

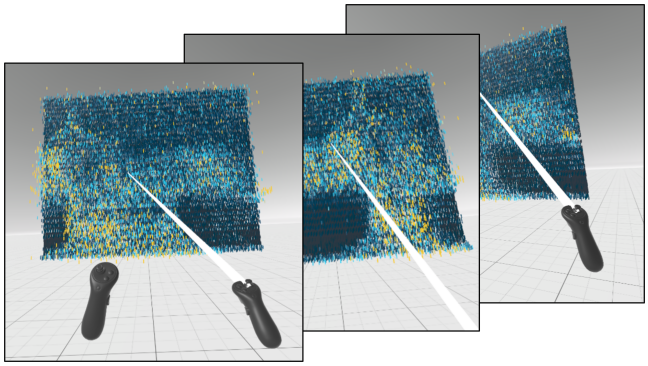

Figure 2: VR application running on Meta Quest 2 showing three frames of a user playing or pausing an animation of activated neurons in neural network layers.

frames-per-second (FPS) is approximately 70 on average, whereas while playing, FPS is approximately 7. Oculus PC VR experiences are recommended to consistently meet 90 FPS (i.e., ~11.1 ms per frame) for machines with recommended specs or 45 FPS (i.e., ~22.2 ms per frame) for machines with minimum specs [11]. The Unity Profiler shows that each frame takes approximately 180 ms to build the USD scene with its meshes. This large amount of time spent could be attributed to the extra CPU time necessary to retrieve the mesh data from the 18 GB USD binary for the next frame and rendering the millions of polygons comprising the mesh.

5 Future Work

Next, we plan to expand this app by: 1) Improving FPS through level-of-detail and/or mesh decimation before USD import to Unity; 2) Adding asset rotation, translations, scaling in Unity via XR interactions, and animation time reversal and scrubbing; 3) Investigating loading USD prefabs from remote sources using the Unity Addressables package, which may allow the USD prefab to be hosted closer to where the raw data is stored, thus enabling new opportunities for remote app playing; 4) Improving the graphics quality via the Unity Universal Render Pipeline and/or High Definition Render Pipeline; 5) Launching the app on MR and AR devices, such as the Microsoft HoloLens, and gathering performance metrics.

6 Conclusion

XR applications such as ParaView VR, SummitVR, and Omniverse XR allow for satisfactory analysis and exploration of some classes of scientific data. However, for massive scientific data produced by simulations executed in HPC applications, more work is necessary to adequately display and interact with such data. We propose a workflow that serves as a basis for future work to display and interact with scientific data in XR applications. The USD file format and related toolsets allow us to package scientific visualizations for consumption in game engines like Unity, and the XR Interaction Toolkit allows us to quickly enable data interaction scenarios for many XR devices.

Acknowledgments

Data courtesy of Eliu Huerta, Pranshu Chaturvedi, Asad Khan, Nikil Ravi, and Minyang Tian with Data Science and Learning at Argonne National Laboratory. This work was supported by and used resources of the Argonne Leadership Computing Facility, which is a U.S. Department of Energy Office of Science User Facility supported under Contract DE-AC02-06CH11357.

References

[1] I. Heldal et al. "Immersiveness and symmetry in copresent scenarios". In: *Pro. IEEE VR*. Mar. 2005, pp. 171–178. DOI: 10.1109/VR.2005.1492771.

[2] Huyen Nguyen et al. "Collaborative Data Analytics Using Virtual Reality". In: *IEEE Conf. on VR and 3D UIs (VR)*. Mar. 2019, pp. 1098–1099. DOI: 10.1109/VR.2019.8797845.

[3] Ken Martin et al. *Taking ParaView into Virtual Reality*. The Source. Sept. 22, 2016. URL: https://www.kitware.com/taking-paraview-into-virtual-reality/ (visited on 06/13/2022).

[4] *SummitVR*. Version 1.1.83. URL: https://www.summitvr.app/summitvr (visited on 06/20/2022).

[5] Nvidia. *Omniverse Platform for Virtual Collaboration*. URL: https://www.nvidia.com/en-us/omniverse/ (visited on 06/17/2022).

[6] Mark A. Bolstad. "Large-Scale Cinematic Visualization Using Universal Scene Description". In: *IEEE 9th Symp. on Large Data Analysis and Visualization (LDAV)*. Oct. 2019, pp. 1–2. DOI: 10.1109/LDAV48142.2019.8944362.

[7] *Unity*. Version 2020.3.26f1. URL: https://unity.com/.

[8] *Houdini*. URL: https://www.sidefx.com/products/houdini/ (visited on 06/27/2022).

[9] E. A. Huerta et al. "Accelerated, scalable and reproducible AI-driven gravitational wave detection". In: *Nature Astronomy* 5 (July 2021), pp. 1062–1068. DOI: 10.1038/s41550-021-01405-0. arXiv: 2012.08545 [gr-qc].

[10] *usdstitch*. Version 22.05b. 2016. URL: https://github.com/PixarAnimationStudios/USD/archive/refs/tags/v22.05b.zip.

[11] Meta. *Guidelines for VR Performance Optimization*. URL: https://developer.oculus.com/documentation/native/pc/dg-performance-guidelines/ (visited on 08/01/2022).