Live Data in VR

Roger S. Rivas
Clemson University
South Carolina , USA
rrivasg@clemson.edu

Abstract

The Live Data in Virtual Reality project developed on Unity is an exciting project created for the class 8810 Eye Tracking II: Gaze Sensing and Interaction in XR at Clemson University in spring-2023. The project aims to combine the latest virtual reality technology to create an immersive and interactive user experience in which users can navigate the environment, and every time the user's gaze focuses on a specific object, the program will fetch a number transmitted by a Web API representing the actual temperature in a water pump.

CCS Concepts: • **Human-Centered Computing** → *Virtual Reality* .

Keywords: Eye Tracking, Unity, Eye gaze input, VR, Live-data

ACM Reference Format:
Roger S. Rivas. 2023. Live Data in VR. In *Proceedings of Clemson University (Classs: 'CPSC 8810 Eye Tracking II: Gaze Sensing Interaction in XR').* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The use of virtual reality (VR) technology has exploded in recent years, with applications ranging from entertainment and gaming to education and healthcare. One up-and-coming area for using VR is the presentation and visualization of data in a virtual environment. It is possible to provide users with a more engaging and informative experience than traditional 2D charts and graphs by using VR for data visualization. This paper presents a VR project developed in Unity Engine that shows real-time information from a Web API and implements gaze tracking functions on the environment.

Virtual reality (VR) has been a subject of research since the mid-20th century[10], with early experiments conducted in psychology and engineering. However, it was not until the 1990s that VR technology became more widely available and affordable, leading to increased research in computer graphics, human-computer interaction, and gaming[10]. In the early days of VR research, much of the focus was on developing the technology, including creating more realistic and immersive graphics and improving tracking and input devices. As technology has advanced, research has shifted towards exploring the potential applications of VR in various fields such as medicine, education, and entertainment. Some areas of current VR research include exploring the use of VR for therapy and rehabilitation, developing more natural and intuitive user interfaces, and investigating the effects of VR on learning and memory. There is also ongoing research into the potential risks and benefits of VR, such as motion sickness and its impact on social behavior.

The benefits of creating a VR project with real-time data interaction and gaze tracking using Unity Engine are the following: First, real-time data interaction can allow users to manipulate and interact with data in a more intuitive and immersive way, making the experience more engaging and potentially leading to a better understanding of complex information. For example, real-time data interaction in a scientific visualization application could allow researchers to explore and manipulate complex datasets more quickly than traditional methods. Secondly, gaze tracking can enhance the user experience by allowing for more natural and intuitive interactions within the virtual environment. By tracking where the user is looking, the VR system can adjust the perspective, display the information most relevant to the user's current focus, and make the experience more immersive and intuitive. Finally, creating a VR project with real-time data interaction and gaze tracking can help to push the boundaries of what is currently possible in VR development. By exploring new techniques and technologies, developers can create more innovative and engaging VR experiences, potentially leading to breakthroughs in VR development.

This paper will provide detailed information on designing and implementing a live data visualization tool in VR using Unity and eye tracking. This paper aims to analyze related research techniques and provide feedback about the challenges faced, what was/was not completed, and the difficulty level of each implementation. Also, provide proven alternatives for implementing gaze tracking and real-time data in a VR project. There are five goals that the author focuses on in this research: create a VR environment in Unity Engine, use HTC Vive Pro Eye to read the user's gaze, dynamically load assets or data when the user watches

Classs: 'CPSC 8810 Eye Tracking II: Gaze Sensing Interaction in XR', Spring, 2023, Clemson, SC

ACM ISBN CPSC-8810 04/25/2023…$0.00
https://doi.org/XXXXXXX.XXXXXXX

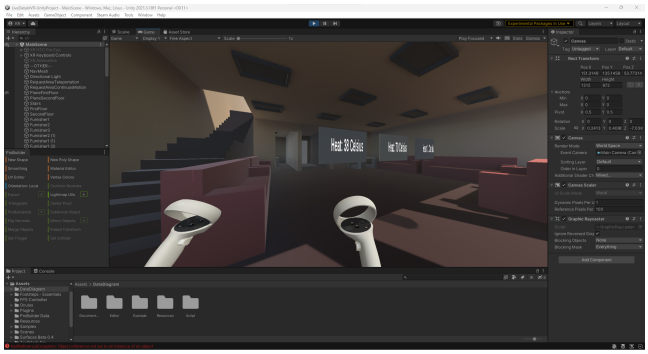

Figure 1. VR System running in Unity Engine.

a specific object, publish the project and research log on GitHub, and preferably provide a budget-friendly VR system with low cognitive load. The project developed in this paper is available on GitHub. To see the project, please visit: https://github.com/RogerSmithR/Live-Data-in-VR.

2 Related Work

2.1 User's gaze tracking

Several studies have explored the use of gaze tracking in various contexts. The following papers were used in this research as a guide to understanding different methods and concepts about gaze tracking before starting the development of the VR system. First, "What you look at is what you get: eye movement-based interaction techniques" by Jacob, Robert JK[4]. proposes a novel approach to eye gaze interaction that allows users to control a graphical user interface (GUI) by looking at specific regions of interest. Second, "Eye tracking based human computer interaction: Applications and their uses" by Chandra, Sushil, et al[1]. provide a comprehensive review of the applications of eye gaze tracking in human-computer interaction.

Also, this research used other papers to guide which tool should be implemented in the VR system. First, "Visualization of Eye Tracking Data in Unity3D" by Müller, Lisa-Maria, et al[9]. proposes methods for integrating eye-tracking data into a virtual reality (VR) environment using Unity3D. Second, "3D gaze in virtual reality: vergence, calibration, event detection" by Duchowski, Andrew T., et al[2]. presents a detailed analysis of the challenges involved in measuring eye gaze in virtual reality environments and proposes a method for detecting events, such as saccades and fixations, in 3D space.

2.2 Live data in VR

The following papers provide concepts and tools essential to understanding the principal challenges faced in this research regarding live data streaming. First, "3DRepo4Unity: Dynamic loading of version-controlled 3D assets into the Unity game engine" by Friston, Sebastian, et al[3]. proposes a method for dynamically loading 3D assets into the Unity game engine in real-time, which is explained in more detail in the section "Architecture Overview." Second, "Unity: Collaborative downloading content using co-located socially connected peers" by Jassal, Prateek, et al[5]. proposes a collaborative downloading of content in Unity using socially connected peers. The authors describe a peer-to-peer (P2P) architecture that enables users to share data and content in real-time, reducing the load on centralized servers and improving download speeds; this paper is used to understand other alternatives to obtain content from the internet. Third, the paper "GreedyDual Web caching algorithm: exploiting the two sources of temporal locality in Web request streams" by Jin, Shudong, and Azer Bestavros[6]. Proposes a web caching algorithm that exploits temporal locality in web request streams to improve performance. While not explicitly focused on VR, the paper's concepts can be applied to VR environments incorporating live data. Fourth, "Unity Networking Fundamentals" by Kelly, Sloan, and Khagendra Kumar[7]. The book provides a foundation for understanding the networking requirements and challenges of creating VR experiences incorporating live data. This book was used as a guide to understanding more about security challenges in networking in Unity.

2.3 Cognitive load

Cognitive load refers to an individual's mental effort to complete a task. "Measuring cognitive load using eye-tracking technology in visual computing" by Zagermann, Johannes, Ulrike Pfeil, and Harald Reiterer[12]. Provides valuable insights into the use of eye-tracking technology for measuring cognitive load in visual computing, which can inform the development and optimization of interactive systems to improve their performance, usability, and user experience. This paper is used as a reference to understand more about how cognitive load in VR systems affects user behavior, this research does not focus on implementing methods to reduce the cognitive load while using the system, but it does take into consideration the information provided by the paper.

3 Architecture Overview

The following section discusses the selection of Unity Engine and HTC Vive Pro Eye for the virtual reality project and how to obtain live data from a Web API (Application Programming Interface) by obtaining the user's gaze using eye-tracking technology. Unity was chosen because it is easy to use, has multi-platform support, a large community, built-in VR support, an extensive asset store, performance optimization tools, and continuous development. HTC Vive Pro Eye was chosen because it has eye-tracking technology, high resolution, developer support, Unity integration, and OpenVR support. The text also discusses obtaining the user's

gaze using Tobii SDK and executing HTTP (Hypertext Transfer Protocol) requests using UnityWebRequest[7]. Finally, the text describes how the idea of fetching live data was inspired by 3DRepo4Unity[3], and how the method was changed to show simple text instead of fetching the entire graph assets to reduce hardware and network costs.

3.1 Why Unity?

Unity Engine is a powerful game engine and development platform that creates games, simulations, and interactive experiences across multiple platforms. Unity is designed to be flexible and easy to use, focusing on making game development accessible to a wide range of users, from hobbyists to professional developers.

Unity Engine was chosen for this project compared to other software for the following reasons:

- `Previous knowledge`: The developer working on the project has more experience programming in C-Sharp language (since 2016) than other programming languages and has already had some experience working in Unity Engine (since 2022).
- `Easy to use`: Unity has a user-friendly interface and a simple learning curve, making it accessible to developers of all skill levels.
- `Multi-platform support`: Unity supports multiple platforms, including Oculus Rift, HTC Vive, PlayStation VR, and Google Cardboard, among others.
- `Community support`: Unity has a large and active community of developers who share their experiences and provide support through online resources.
- `Built-in VR support`: Unity has built-in support for VR, which means that developers can easily create VR applications and experiences without needing to write additional code.
- `Extensive asset store`: Unity's asset store contains a wide range of pre-built assets, including models, animations, and scripts, which can save developers time and effort. This variety of assets is essential to expand the project or simulate specific scenarios.
- `Performance optimization`: Unity offers built-in tools for optimizing VR performance, which is critical for delivering a smooth and immersive VR experience.
- `Continuous development`: Unity is constantly evolving, with regular updates and new features added, ensuring it remains a cutting-edge engine for VR development.

3.2 Why HTC Vive Pro Eye?

HTC Vive Pro Eye is a virtual reality (VR) headset that allows users to experience immersive virtual environments. The headset includes high-resolution displays, integrated eye-tracking technology, and a variety of sensors for tracking the user's movements and gestures. This device was chosen for this project compared to other devices for the following reasons:

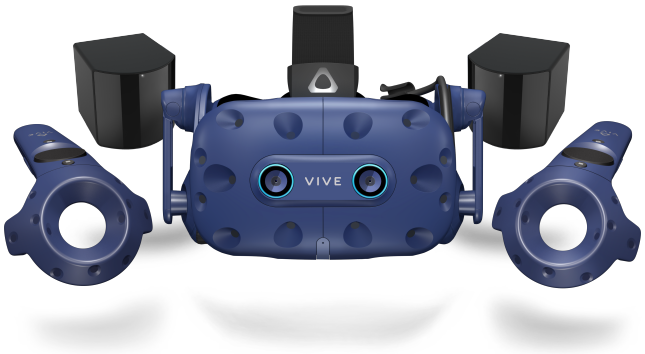

Figure 2. HTC Vive Pro Eye Device.

- `Device Availability`: The device is available for use free of charge in the Clemson University laboratory where the project was carried out.
- `Eye-tracking technology`: The Vive Pro Eye is the first VR headset to feature eye-tracking technology, which allows a more natural interaction, immersive experiences, and the potential for new forms of exchange, such as eye-based navigation and selection.
- `High-resolution`: The Vive Pro Eye has a high resolution of 1440 x 1600 pixels per eye, which provides a more detailed and precise VR experience (convenient when selling the software).
- `Developer support`: HTC is firmly committed to developer support, offering resources such as the Viveport SDK and the Vive Developer Community to help developers create VR experiences with the Vive Pro Eye.
- `Unity integration`: The Vive Pro Eye is fully integrated with Unity, so developers can easily create VR experiences for the Vive Pro Eye using the Unity engine.
- `OpenVR support`: The Vive Pro Eye is compatible with OpenVR, an open standard for VR development that supports multiple VR devices and engines. Developers can use the Vive Pro Eye with other VR devices and machines if desired.

3.3 Obtaining the user's gaze

The system identifies when the user watches an object using the "GazeFocusChanged," a class obtained from the Tobii SDK for Unity. Tobii SDK (Software Development Kit) is a suite of software tools and APIs (Application Programming Interfaces) provided by Tobii AB, which develops eye-tracking technology. The Tobii SDK allows developers to integrate eye-tracking functionality into their applications and games

and use eye-gaze data as an input method, among other features. This information was discussed in the paper "3D gaze in virtual reality: vergence, calibration, event detection" by Ph.D. Duchowski, Andrew T. et al[2]. in 2022. It discusses the challenges and opportunities of 3D gaze-tracking technology in virtual reality (VR) environments. It also describes some techniques for calibrating 3D gaze-tracking technology, like using natural features as reference points.

3.4 Obtaining live data

When the user's gaze is watching the object, the program executes an HTTP (Hypertext Transfer Protocol) request to a Web API which transmits a JSON response containing a number done using the class UnityWebRequest. UnityWebRequest is a class in the Unity game engine that allows developers to send HTTP requests and receive responses from web servers[7]. One crucial point is that the program executes the HTTP request in every frame, and the execution is controlled using Coroutines. Performance in every HTTP request is essential; for that reason, the request was made using the URL without SSL (Secure Sockets Layer). If confidentiality is the first concern, the data can be encrypted in the Web API and decrypted in the VR system to secure the transmission using asymmetric encryption to improve performance[11]. The idea of fetching live data was inspired

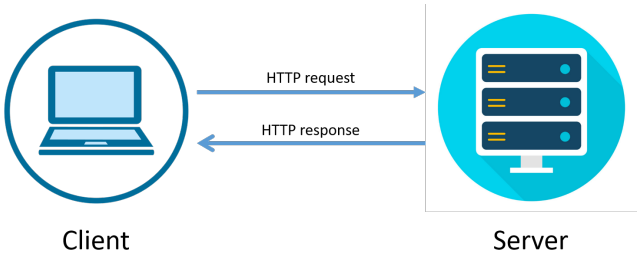

Figure 3. Representation of a Web HTTP Request.

by the paper "3DRepo4Unity: dynamic loading of version controlled 3D assets into the Unity game engine" by Friston, Sebastian, et al[3]., which describes a plugin for the Unity game engine that allows developers to access and use 3D models and data stored in the 3DRepo platform directly within Unity. This integration makes it easier for developers to access and use 3D data in their projects, streamlining the development process and enabling more complex and sophisticated 3D applications. However, the method was changed from fetching the entire graph assets to showing a simple text from an HTTP request because of the high hardware and network cost when using 3DRepo4Unity.[**?**]

The data is transmitted by a Web API created in ASP.NET Core Web API[8] by our team, and the server was provided for free by Sistemas RSA. To see the API URL, please visit http://livedataapi.sistemasrsa.com/api/heatvalue.

3.5 Development log

The section is designed to present the schedule followed and the project's progress.

January 2023: Defining project goals, schedule, and design.

February 2023: Creation of the virtual reality environment using Unity.

March 2023: Research related topics and show the project proposal in class, including a paper proposal. Also, it successfully tested and integrated dynamic loading of 3D assets and gaze interaction with objects in the code.

April 2023: SDK and plugins were implemented into the Unity project and tested with new code. Additionally, a web API project was created to improve data transmission performance, and a website was developed to showcase the research log content. The visual details in the Unity project were finalized, and the research log was presented in class.

May 2023: The standalone executable Windows and Mac OS versions are created. The final presentation is given in class.

4 Future Research

While the current implementation of the live data VR application allows users to view real-time data while using eye-tracking technology, there is room for further research in implementing real-time analytic. By integrating analytic into the application, users can better understand the data and draw insights more quickly.

First, the research could be focused on developing an analytic dashboard within the VR environment. This dashboard would allow users to view key metrics and visualizations, such as charts or graphs, in real-time. The dashboard could also provide interactive features like filtering data by different parameters or zooming in on specific data points.

Second, the research could focus on developing machine learning models to save data in the system and provide predictive analytic within the VR environment, which is essential, especially when there is no network connection. By analyzing historical data, these models could provide insights into future trends or anomalies in the data. These insights could help users make informed decisions and act quickly, even in lousy network connection scenarios.

Third, network performance was an important issue while developing the VR system. Further research should be done on improving the speed of importing assets in the VR project while having a bad network connection or slow connection speed. Using secure connections was also a main performance issue, and research could be done on improving data transfer using SSL (Secure Sockets Layer).

Overall, users can better understand the data and make more informed decisions by integrating faster connectivity and real-time analytics into the live data VR application. This area of research has the potential to revolutionize how users interact with and understand data in a virtual environment.

5 Conclusion

In conclusion, virtual reality (VR) technology has seen explosive growth in recent years, and one area of interest is using VR for data visualization. This paper presents a VR project developed in Unity Engine that utilizes real-time data interaction and gaze tracking to create a more engaging and informative user experience. The paper aims to provide detailed information on designing and implementing a live data visualization tool in VR using Unity and eye tracking, including challenges faced and alternative techniques. Also, the paper describes the selection process for the Unity Engine and HTC Vive Pro Eye for this project, along with eye-tracking technology to obtain live data from a Web API. Unity was chosen due to its user-friendliness, multi-platform support, large community, built-in VR support, extensive asset store, performance optimization tools, and continuous development. HTC Vive Pro Eye was selected for its high resolution, developer support, Unity integration, OpenVR support, and eye-tracking technology. Tobii SDK was used to obtain the user's gaze, while UnityWebRequest was used to execute HTTP requests. The idea of fetching live data was inspired by 3DRepo4Unity[3], and the method was modified to display simple text to reduce hardware and network costs. The project aims to create and publish a VR environment on GitHub for others to explore and build their own VR system.

Acknowledgments

The author thanks Ph.D. Andrew T. Duchowski from Clemson University for the helpful discussions about this work and for providing guidelines on the elaboration of this paper. The views and conclusions in this document are below the author's.

References

[1] Sushil Chandra, Greeshma Sharma, Saloni Malhotra, Devendra Jha, and Alok Prakash Mittal. 2015. Eye tracking based human computer interaction: Applications and their uses. In *2015 International Conference on Man and Machine Interfacing (MAMI)*. IEEE, 1–5.

[2] Andrew T Duchowski, Krzysztof Krejtz, Matias Volonte, Chris J Hughes, Marta Brescia-Zapata, and Pilar Orero. 2022. 3D gaze in virtual reality: vergence, calibration, event detection. *Procedia Computer Science* 207 (2022), 1641–1648.

[3] Sebastian Friston, Carmen Fan, Jozef Doboš, Timothy Scully, and Anthony Steed. 2017. 3DRepo4Unity: Dynamic loading of version controlled 3D assets into the Unity game engine. In *Proceedings of the 22nd International Conference on 3D Web Technology*. 1–9.

[4] Robert JK Jacob. 1990. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 11–18.

[5] Prateek Jassal, Kuldeep Yadav, Abhishek Kumar, Vinayak Naik, Vishesh Narwal, and Amarjeet Singh. 2013. Unity: Collaborative downloading content using co-located socially connected peers. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 66–71.

[6] Shudong Jin and Azer Bestavros. 2001. GreedyDual Web caching algorithm: exploiting the two sources of temporal locality in Web request streams. *Computer Communications* 24, 2 (2001), 174–183.

[7] Sloan Kelly and Khagendra Kumar. 2021. *Unity Networking Fundamentals*. Springer.

[8] Andrew Lock. 2021. *ASP. NET core in Action*. Simon and Schuster.

[9] Lisa-Maria Müller, Kilian Mandon, Pascal Gliesche, Sebastian Weiß, and Wilko Heuten. 2020. Visualization of Eye Tracking Data in Unity3D. In *Proceedings of the 19th International Conference on Mobile and Ubiquitous Multimedia*. 343–344.

[10] Janet H Murray. 2020. Virtual/reality: how to tell the difference. *Journal of visual culture* 19, 1 (2020), 11–27.

[11] Gustavus J Simmons. 1979. Symmetric and asymmetric encryption. *ACM Computing Surveys (CSUR)* 11, 4 (1979), 305–330.

[12] Johannes Zagermann, Ulrike Pfeil, and Harald Reiterer. 2016. Measuring cognitive load using eye tracking technology in visual computing. In *Proceedings of the sixth workshop on beyond time and errors on novel evaluation methods for visualization*. 78–85.

Received 25 April 2023; revised 26 April 2023; accepted 27 April 2023