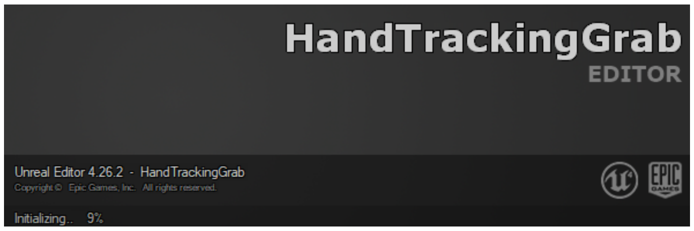
VR Hand Tracking

Razvan Ruxandari
Gheorghe Asachi Technical University of Iaşi
Faculty of Automatic Control and Computer Engineering

***Abstract*—This paper presents a virtual reality (VR) hand tracking work-in-progress game with a survival character, allowing users to interact with the virtual environment by using their hands to pick up objects. The game is developed using Unreal Engine 4.26.2, and the hand tracking technology is implemented through Oculus Plugin. The paper discusses the methodology, implementation details, and user feedback.**

***Index Terms*—Virtual Reality, Hand Tracking, Game, Survival, Interaction**


HandTrackingGrab
EDITOR
Unreal Editor 4.26.2 - HandTrackingGrab
Copyright © Epic Games, Inc. All rights reserved.
Initializing... 9%

I. Introduction

In the realm of virtual reality (VR) gaming, the predominant trend involves the utilization of controllers for user interaction. However, a distinct gap exists when it comes to games that leverage the innate dexterity of human hands. This paper presents a pioneering venture into the unexplored domain of hand tracking in VR gaming, specifically through the creation of a survival-themed game. Unlike the conventional controller-based experiences, this game harnesses the natural motion of users' hands for a more immersive and intuitive interaction.

The decision to delve into the realm of survival-themed gaming was prompted by the scarcity of such experiences in the VR industry. While a plethora of games cater to various genres, the fusion of hand tracking and survival gameplay remains largely uncharted. This endeavor seeks to break new ground by combining the captivating mechanics of a Minecraft-like survival game with the innovative technology of hand tracking.

Moreover, this project also serves a dual purpose by aligning with the objectives of an Image Processing course. The implementation of hand tracking in a VR environment involves intricate image processing techniques, making it an ideal subject for exploration within the context of this academic pursuit. This paper unfolds the journey of crafting a VR hand tracking game, shedding light on the development process, challenges encountered, and the unique aspects that make this survival-themed experience a noteworthy addition to the gaming landscape.

II. Related Work

The landscape of virtual reality (VR) gaming has witnessed significant developments in the past years, with a multitude of projects made both for the standard 2D Keyboard and mouse gameplay - but also with VR capabilities. Some examples are: FNAF VR 1 and 2 - games that require you to interact with certain objects/props (push buttons, pull levers, throw objects, or aim with laser guns) in order to achieve your goal: to survive the shift or Beat Saber - where your controllers behave as laser swords in rhythm-based gameplay where you're required to slash the upcoming "notes" so the player can progress through the song

My project aims to mash all of these aspects together, creating a wholly new experience with a VR-based survival game with hand-tracking capabilities in order to deliver more natural interactions

III. Methodology

The following key aspects delineate the methodology adopted for this project:

A. *Game Development Platform (Fig.1)*

The decision to employ Unreal Engine as the development platform was motivated by the user-friendly interface and the ease of game creation - no matter the type, theme, or complexity. Unreal Engine's visual scripting system (Blueprints) facilitated the implementation of hand-tracking features, enabling a simpler and more attractive development process and reducing the actual code writing by up to 100 percent. Unlike other platforms such as Unity, Unreal Engine's standalone game engine nature ensured a focused and dedicated approach to game development, designed to ensure accessibility to a wide spectrum of individuals, regardless of their current coding skills. Thus, it allows aspiring creators to craft memorable and impactful experiences, requiring only a few tutorials and adequate dedication to the development process.

B. *Hardware Setup*

The Meta Quest 2, formerly known as the Oculus Quest 2, served as the chosen VR headset for this project due to its availability and attractive price. Initially, the VR headset was connected only to my laptop via a USB to TypeC cable via Oculus Link. Later on, my PC was also included in the project development as a means of testing compatibility by offering it a broader range of devices. The hardware configurations used throughout the entire testing phase are CPU: Intel I7 9750h —— GPU: NVidia RTX 2060 —— 32GB RAM and CPU:

AMD Ryzen 7 5800x3D —— GPU: NVidia RTX 3060 —— 64GB RAM.

C. OpenXR Runtime

To harness the full potential of the Meta Quest 2 and ensure compatibility with the plugins that were included in the project, Oculus was selected as the default OpenXR Runtime. Oculus is the software layer that can facilitate the implementation of the OpenXR standard on Oculus devices, created to offer the possibility of building applications that will work across a wide variety of devices. Its compatibility in particular with Unreal Engine only encouraged the usage of this particular software.

D. Unreal Engine Implementation

Within Unreal Engine, the implementation of hand tracking involved a combination of Blueprint scripting and different modules to be integrated. The Blueprint visual scripting system allowed for the creation of interactive elements based on hand gestures and movements, as well as integrating custom actions and hand models.

The final implementation resulted in a WIP VR hand-tracking survival prototype, evolved enough to showcase the possibilities of such technology if given full investment.

IV. Implementation

Creating a survival game from scratch requires intense work to be put into it, and a VR type of survival is no different, if not even more demanding, both resource and implementation wise. Several crucial steps have been followed throughout the entire development process, from creating a new environment and acquiring the needed material in order to ensure functionality and compatibility, to artistic contribution for a truly immersive and engaging experience for users, while keeping it different from other typical survival games. In the following pages, the entire progress will be detailed for a better understanding of the workflow and the resources used to bring the project to a presentable state.

A. Plugins Used

Two primary plugins played a crucial role in the development of the project: the Oculus Plugin and the SteamVR plugin, both of which can be found within the Unreal Engine's plugins library. They provided essential functionalities, from which the Oculus Hand component stands out the most. This particular feature has been further extended throughout the development process to meet the desired hand-tracking options by adding a plethora of features, such as object pickup animation or new functions to aid to the skeleton of the given component.

B. Development

The entire project development is based around the same logic as any basic game creation- it all started with the Game Loop part of it. As a quick reminder, a game loop is the infrastructure that moves a player between gameplay and the "between session" elements that frame and propel

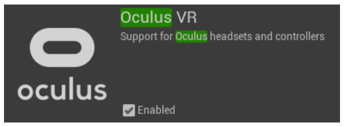

Oculus VR
Support for Oculus headsets and controllers
Enabled

Fig. 1. Oculus VR plugin

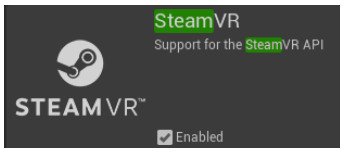

SteamVR
Support for the SteamVR API
Enabled

Fig. 2. SteamVR API Plugin

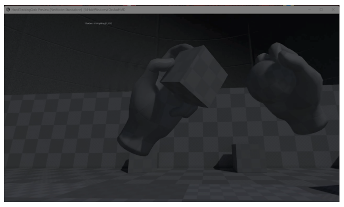
their experience. After constructing the basics, the problem of properly positioning the hands in order for them to be visible for the player arised, hence why the existence of an outside camera, also known as "HandPawn" within the actual project files.

Through an entire sequence of functions that determine your positioning, the whole HandPawn component manages to insert the hands in front of the camera with the according offset so that the player can freely see them without issue. Using the cameras the Quest 2 is provided with, said positioning is filtered through properly in order to achieve the best results.

Alongside the HandPawn component is another one simply called Hand, which provides a representation of our hands in the virtual environment. Alongside basic logic and further implemented functions deciding from the movement of the hand to the action of picking up a valid object, this component is vital in the project as a whole: without it, the idea in itself would be already scrapped.

C. Asset Integration

To enhance the visual and environmental aspects of the game, a combination of free and paid assets from the Epic Games Store was attached to the project. The diverse range of assets contribute to the creation of a captivating and dynamic game world. However, the high-quality nature of these assets needed a size reduction to optimize the project's size. Despite the reduction, the final cut totaled 20GB. In this particular demo, the main assets contained textures that contributed to the environment akin to a clearing within a steep forest, with an accessible water pond and lifelike trees surrounding the player.



D. Size Optimization

Given the goal of making the project accessible on most Windows devices with different specifications - one key strategy needed was the implementation of Level of Detail (LOD) techniques. LOD dynamically adjusts the level of detail and texture resolution for assets based on the distance between the user and the objects in the game. This not only ensured a more efficient use of resources, but also contributed to the seamless performance of the game on hardware with varying specifications.

In the long run, such aspects will always have to be taken into consideration, especially if the developer wishes to make it as accessible as possible. By applying advanced compression techniques, minimizing redundant code and strategically selecting the preferred asset resolutions, the goal of gaining a balance between visual quality and overall performance was reached in due time. This way, efficiency without much compromise of the visual aspects of the game has been reached.

V. Results

The current preview of the project stands as a prototype, showcasing a few working features. Despite the presence of some bugs, the basic elements of the game and working hand-tracking are in place, offering a glimpse into the potential of the project.

Looking ahead, there is a clear path for the game's future development. The ambition is to transform this prototype into a bug-free, finalized product that could potentially be shared with a wider audience. The decision to consider uploading the game to a platform (Steam, itch.io, etc.), whether as a free release or with a low price, is inevitable.

VI. Conclusions

This type of evolution in gameplay has not been fully reached yet due to the quite challenging problems that lay ahead of developers, from insufficient resources to lack of knowledge in said domain. VR Hand Tracking is still a maturing technology, and developers might be waiting for more upgraded and reliable solutions before adopting them to the projects.

In the end, the VR industry is dynamic, and advancements are being In due time, after possible influences of market trends and after this kind of technology reaches a stable point, VR Hand Tracking could become more prevalent in various gaming applications.

References

https://developer.oculus.com/documentation/unreal/unreal-hand-tracking/

https://docs.unrealengine.com/5.0/enUS/BlueprintAPI/OculusLibrary/HandTracking/

https://docs.unrealengine.com/4.26/enUS/BlueprintAPI/OculusLibrary/HandTracking/

https://developer.vive.com/resources/openxr/openxr-pcvr/tutorials/unreal-engine/integrate-hand-tracking-data-your-hand-model/