VIRTUAL REALITY GAME DEVELOPMENT AND OPTIMISAION WITH EXTERNAL HAND MOTION SENSOR

Stud. ERIK DOBRA; Consultant: Dr. ÉVA NAGYNÉ HAJNAL
Óbuda University Alba Regia Technical Faculty, Hungary

SUMMARY: *Virtual reality is an environment simulated by a computer that you can interact with. VR replaces reality with an imaginary world that can be seen through a special head-mounted device (HMD). In this environment, hand tracking is an important task. Many devices do not have this functionality or it is not sufficiently accurate. It is common that hand motion tracking is designed to follow a special hand controller, whereas many applications would require the use of someone's own hand.*

In this paper I would like to demonstrate the addition of a hand recognition device to VR glasses. In the assignment I added a leap motion sensor to a pair of Windows Mixed Reality VR glasses, which allows hand tracking without a hand controller. The implemented solution was tested in a sample application. The theme of the game is horror, which encourages the player to react quickly. I used an alternative implementation of moving in virtual space in the application. The development of the control unit redesigned for virtual space was done in a Unity based project. The system allows players to perform specific gestures using the thumb and index finger, specifically to press buttons. I evaluated the software using the built-in profiling system, which shows what future optimizations are possible within the project. In summary, we have succeeded in creating a hand recognition system that tracks accurate hand movements in real time, which is a significant improvement in hand recognition implementation compared to the driver produced for the system.

Keywords: *VR; Virtual Reality; Hand Gesture sensor; Leap Motion;*

1. Introduction, Objective

My goal was to create a VR game that was both fun to use and has the potential to open up new ways for users to use handheld interactions and in-game movement. I chose virtual reality as a topic because this sector is currently at a turning point. I looked at the problem from a developer's point of view: how can I create software that meets the needs using specific hardware, how difficult is it, how can it be done. Bringing our limbs into the virtual space is a stepping stone to this, but not the last.

I must add, of course, that the ever-changing implementation of virtual reality has its place in many areas, not only in the entertainment industry, but can also benefit many other industries in the field of simulation: we know of applications in the hazardous or heavy industry from our studies today. There is a possibility that my system could be used for other industries, such as in heavy-duty work simulations[1], or with patients with recovering sensing abilities[2].

I also add that there is an increased tendency for people to stay more and more indoors (specifically because of the current health situation[3]) and consequently people are looking for ways to break away from everyday reality, so there is an increased willingness to connect to virtual reality[4]. That's how I got my first VR devices, which then pushed me forward.

Within the game, subtle hand movement tasks have been implemented, such as

controlling a console, which presents the player with tasks. It was also created three different ways to control the game. The comparison of these is shown on a learning curve. To test the stability I used the profiler provided by the development environment.

2. Game concept

Knowing what tools I would be able to work with, I tried to create a game plan. My first idea was a virtual library, but it turns out that similar things had already been created[5], and the scope for its development is quite given (and narrow). I also thought about holding social events in virtual reality, but events that require complex senses, such as wine tastings or festivals where you can't virtualise the touch of others or the taste of wine - I felt we were not yet at the point where this could be created virtually and seamlessly.

My final idea was a kind of adventure course and a random maze became my final decision.

The theme was intended to have some horror content to encourage the player to complete the game. I have devised a way to make the game generate a random path, so that every time you try, you will find yourself facing a new challenge. The rules of the game are the next.

- Player runs away from a scary character, if he gets caught, it's game over,
- on the other hand, the game should challenge the player in some way, and the game control and interactions should implement some kind of rethought technique. I thought that a scary character with a phobia of my own and many other people was a good idea, so I chose the arachnophobic character, the spider.
- I also decided in time that the player should have three lives, because I want the player to be able to focus on the tasks, balancing the fear factor and the task solving within the game.
- To make sure that not only one person can play the game, a stopwatch was implemented, which is timed to end when the player completes the game, so that players can compete against each other.

Of course, as it's a random track generation game, the fastest players will be those who know the game and can play with a sense of purpose, although luck also plays a small part, as it's faster to complete a track that's straighter and doesn't require a lot of searching for elements of the task.

3. Determination of alternative movement

Meta's Oculus Quest was the only VR device that included rudimentary hand recognition, but it was not feasible to develop with it, and even its accuracy is not up to the level of other devices, more specifically Leap Motion guarantees millimetre accuracy[6] while the Quest and Quest 2 only come close. The only drawback of the Leap Motion is its field of view compared to the Quest, but that's plenty to provide a decent VR experience[7].

After considering the Leap Motion control unit, I thought about implementing it first and foremost, but that required physically connecting the Leap Motion to the head unit. Based on the manufacturer's recommendation, I mounted the Leap Motion in the middle of the front of the HMD and designed an adapter to mount it, according to the manufacturer's website[8] and then modelled it in Fusion 360. I drew a frame around the model and then cut out the port location (fig. 1). Along the frame, I made two bumps on each side, and then created a support element to fit the curve of the VR Headset, which fits at the points of the bumps to ensure it is held in place.

All this was done in Ultimaker Cura, which implements the printing mesh for the model. The elements were then printed (fig. 2).

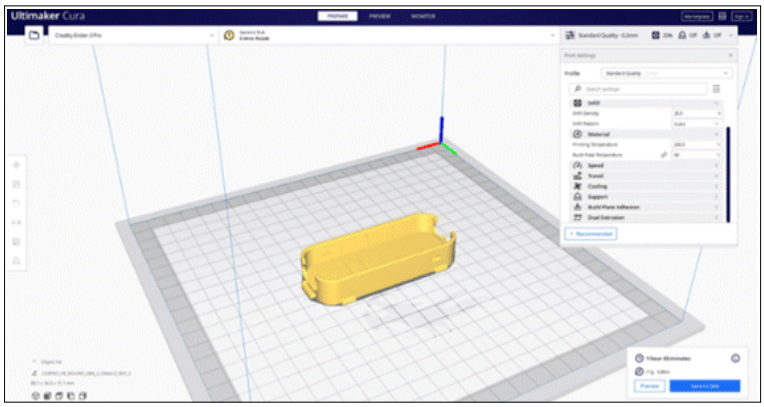

Fig. 1. Modelling in Fusion 360 software

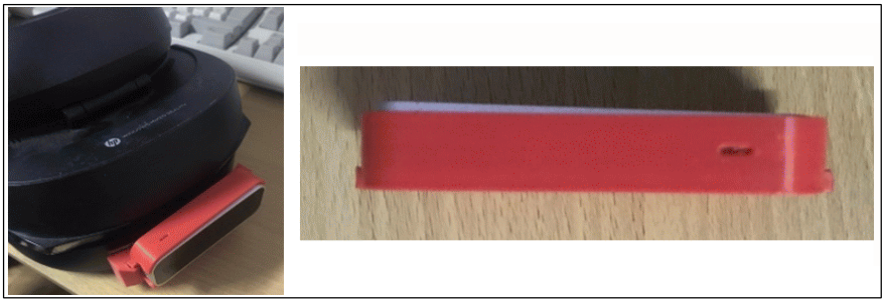

Fig. 2. Printed Leap Motion holder from different angles

After that I had to plan what kind of cases, VR features could be implemented if you use your own hands in the virtual space (fig. 3).

ATM-like models were created with buttons on them. The question is whether fingers can be recognised and interacted with. Next to the buttons, I have also placed sliding cubes that need to be dragged to continue the game.

During testing, I found that if the height was set correctly when the VR HMD was first used, it was quite pleasant and user-friendly to use these elements. If the height was not set correctly, it was not possible to reach the console. If it wasn't set correctly (for example, the ATM was too big compared to the user), then you had to reach up above the user's head, then the Leap Motion hand recognition sensor started to get unstable and didn't register button presses properly.

There is also a life level indicator system (fig. 4) that monitors the rotation of the carpal bones on the left arm and appears when you turn your palm towards you. The player has three lives, and this is indicated to the player here.

4. Alternative solutions for virtual movement guided by hand gestures

a. Control by wrist movement. In order to capture a movement (motion capture), we need to have sensors to determine the direction and state of the body.

The primary problem was the global/local north point assignment. Which sensor will tell me which way I am looking, "which way is north"? Unfortunately Leap Motion doesn't have any sensors that could help me, I might have been able to with optical motion detection, but that would have been a very resource intensive task.

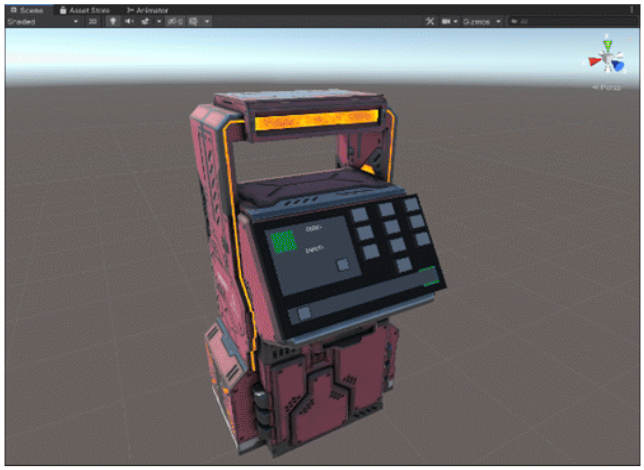

Fig. 3. Interactive buttons within the game

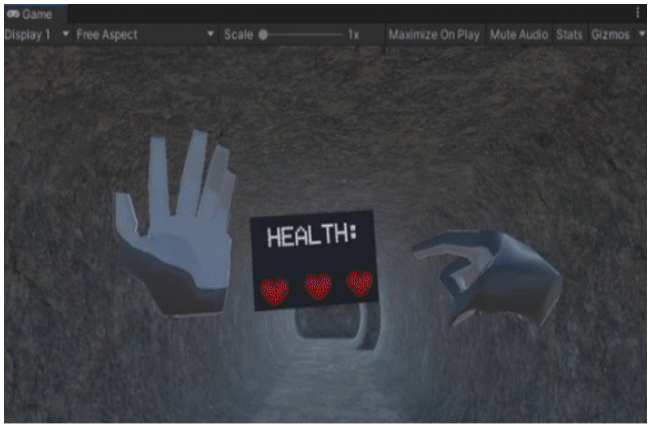

Fig. 4.: Life indicator fitted to wrist movement

The basic idea of where to place trackers (sensors that transmit motion and direction) is to look for four positions[9]:

- Head-tracking-Head movement detection
- Hand-tracking-Hand movement detection
- Hip-tracking - hip movement detection
- Leg-tracking - foot detection.

The HMD Headset is supposed to be a good solution, but there's a problem with that too. Unfortunately, Unity does not support the integration of dynamic north point selection when using these two devices at the same time, and no matter how I tried to get to the bottom of the problem, I always ended up with the same issue.

On the other hand, I also thought about trying the one developed by HTC Vive Tracker, which can transmit data when attached to the player. Unfortunately, I quickly abandoned this idea when I found out how much it would cost to build, as these devices only work with Base Station. The Base Station actually works by constantly flooding the room with invisible light. The receptors of the devices to be monitored pick up the light emitted and from that you can tell where they are in relation to the Base Stations[10].

At first, I thought that the relative movement would be the difficulty in the

game due to the lack of a global north point, but I soon found that it was very difficult to get used to this movement pattern. Later a combination of the two control principles (joystick and hand recognition) were used.

b. Magicsee R1 controller. The first, in which I used a tiny Magicsee R1 joystick ring. This ring has an analogue lever that can be hidden on the inside or outside of the hand, which doesn't interfere with hand recognition, so it retains the advantage of both worlds, which is its own speciality. Fine hand movement can still be used to solve in-game tasks, but movement is handled by the joystick and headset.
This is a safe approach to the problem that will ensure proper, guaranteed operation.

c. Virtual VR keystrokes with hand recognition. Another, more daring approach, which is not provided by the vendors, is to push the boundaries of the framework with our own implementation. As some elements are not implemented in the Unity development environment, situations similar to improvisation have arisen in one or two places.

Here's how it works:

If the player extends the thumb and index finger (forming a pistol) with both hands, a script is activated, which results in a virtual keystroke (in my case 'W') (fig. 5).

Fig. 5. Activating hand gesture illustration

A keystroke will force the virtual keyboard to move forward, and then you can use the north point of the headset to move forward. If one of your hands is no longer showing the gesture, the forward progress will stop.
It is important to note, once again, that this solution is not native, it is not feasible by default in the current development environment, but it is representative of what this build would be capable of. For me, this was important to point out because it was conceptually suitable, but a solution that could affect the stability of the computer would not be considered as final (in my case, virtual keystrokes meant that even if I minimized the program, it still monitored the hand movements and guaranteed keystrokes outside the program).

5. Comparative analysis of manual motion control methods

I have compared the learning curves of the above-mentioned solutions in a non-representative compilation that I have experienced. The three different locomotion technologies I implemented were compared using the timer used in the game. Each successful run was recorded, measured and recorded in 'table 1', rounded to minutes for simplicity. The 25 runs of each type were normalised to show the performance that can be achieved with these three types of movement. The following is the formula for the calculation:

$$z_i = -\frac{x_i - x(max_n)}{x(max_n) - x(min_n)}$$

where:

x_i= the actual measured value
n = the set with the smallest interval and the smallest variance

In each case, I played the game 25 times to illustrate the potential for improvement. As the figure above shows, the most difficult movement technique to learn was the wrist movement based one, which made it even

more difficult to learn due to the lack of dynamic north point selection mentioned above. The joystick controller solution is the most effective, but it still does not give up the idea of having empty hands, so the golden mean is the 'fingergun' method, which is more difficult to learn than joystick ring, but guarantees similar results.

Occlusion culling is the first problem that I had to solve (fig. 7).

Green: rendering, Blue: scripts, Orange: physics simulation, Turquoise: animation.

As the graph shows, in the first versions of the game, the experience was poorly optimised and the game often stalled, as shown by the jagged edges on the graph. The

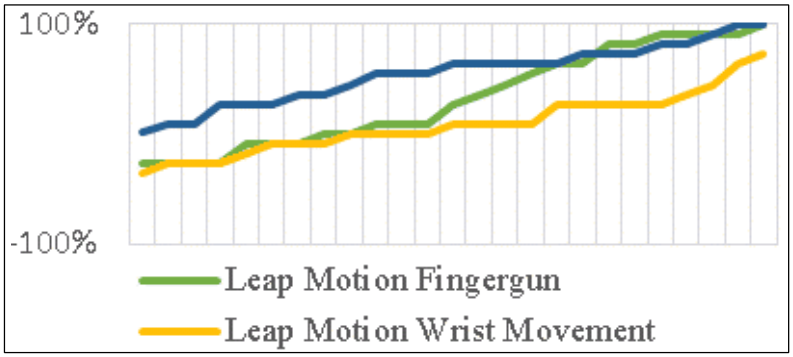

Fig. 6. Learning curve I have experienced.

6. Game optimisation, results, opportunities

It was used different techniques to make the VR gaming experience as optimal as possible for everyone. To evaluate software problems, I used a profiler built into the Unity development environment[11]. This mainly shows, in frames, what activity/ processes the machine is currently processing. Here you can analyse outliers to make the game run as smoothly as possible, without slowdown-causing processes.

value in green is the rendering, which means that most of the resources are needed to render the image. The edges may indicate that looking in a direction where there are several game elements, processing them could put a sudden load on the machine, so there were elements to process on the track that the program was taking into account even when you were not near (or not looking at) them.

The solution to this is tricky within Unity because I knew that I had to implement some kind of occlusion culling. Unfolding this,

Fig. 7: Unity profiler graph for the 0-8 ms interval

what it does with our program is that things/textures that the player can't see are ignored and rendered inactive. It is important to note that if there is dynamic content on the map (such as an enemy that is moving), it must be marked within the program not to be affected by this procedure, otherwise it can freeze that entity until the player has a view of it.

The problem with this is that Unity likes to 'bake' this data into the program, so it knows what it does and does not apply to before the program is run. The objects in my program are generated dynamically at runtime, so Unity has no idea what kind of elements can be cleaned from the player.

I have created a simple algorithm for this:

Take the player, place an invisible, translucent brick body (collider) on top of it, and scale it up to a larger size. This will be our 'line of sight'. Anything that physically collides with this brick will be displayed, anything else that is not marked as dynamic will not be displayed.

This solution works, but I should add that if you are in an open space, you may see gaps in the texture on the horizon. To remedy this, I blurred the space further away so that it's not so distracting.

Leap Motion integration:

The next optimization I did was with the Leap Motion integration itself. When one starts developing for a VR device in Unity, basically the OpenVR base should be marked as an option, this has the advantage of compatibility, but it severely limits the hardware in both functionality and responsiveness.

Contrary to everything, I turned off this compatibility layer and manually integrated the SDKs' content and found that the response time of the hardware was much shorter than if I had left it to the development environment.

I did a measurement using both internal and external footage of how instantly each of the two methods responded to the original camera-adjusted image. I set the camera and internal footage to 240 FPS and compared the values to the native visualizer and to each other.

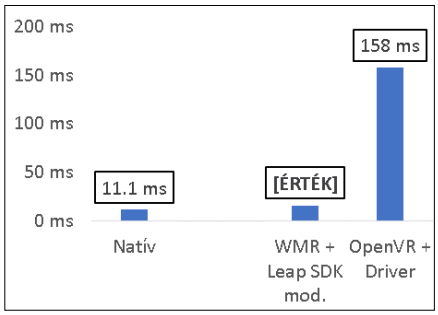

Fig. 8: Leap Motion response time with different implementations.

As the figure above shows, we have managed to create a near-native implementation compared to what OpenVR guaranteed. The combination of these two solutions has resulted in a well-functioning, stable and optimised game. The profiler also now presents a much smoother and higher frame rate game.

Possibly, new challenges could be associated with the new characters, so even refining recognition, perhaps recognizing new hand gestures, could be a possible development point.

7. Conclusion

My goal was to solve the problem of using hands in the VR environment, replacing handheld controllers. The choice of topic was motivated by the fact that there are many applications where this might be necessary. This was supported by the literature research, but it is not solved in many HMD devices. Where it is solved, the accuracy of hand tracking and gesture recognition typically leaves something to be desired. As part of this experiment, I tested several devices and selected a hand recognition device, the Leap Motion sensor, to pair with my existing VR glasses. To fit it together, A 3D printed mounting element

secured the device. I reviewed the literature to see what options were available if we could incorporate the hand recognition device for in-game movement and interaction. I then implemented a virtual console that could be controlled by buttons and a special display that worked based on gesture evaluation. I implemented 2+1 separate types of solutions for controlling spatial movement and compared their usability. I created a game that randomly generated a maze and included all the features I developed. I tested the application to see how stable it is using a profiler and then optimized it. I investigated which SDK would give the best results and developed a simple occlusion culling algorithm that could be used in VR. Based on my tests, I was able to implement a low-latency hand recognition algorithm, which is a beneficial improvement over the built-in compatibility layer. As an extension, I proposed additional new program features that could have specific behaviours.

Notes

1. F. Nainggolan, B. Siregar, and F. Fahmi, "User Experience in Excavator Simulator Using Leap Motion Controller in Virtual Reality Environment," in Journal of Physics: Conference Series, vol. 1566 (Institute of Physics Publishing, 2020), https://doi.org/10.1088/1742-6596/1566/1/012093.
2. D. E. Holmes et al., "Using Fitt's Law to Model Arm Motion Tracked in 3D by a Leap Motion Controller for Virtual Reality Upper Arm Stroke Rehabilitation," in Proceedings - IEEE Symposium on Computer-Based Medical Systems, vol. 2016-August (Institute of Electrical and Electronics Engineers Inc., 2016), 335–36, https://doi.org/10.1109/-CBMS.2016.41.
3. Ravi Pratap Singh et al., "Significant Applications of Virtual Reality for COVID-19 Pandemic," Diabetes and Metabolic Syndrome: Clinical Research and Reviews 14, no. 4 (July 1, 2020): 661–64, https://doi.org/10.1016/J.DSX.2020.05.011.
4. "Coronavirus Sent Us Home. Will VR Bring Us Back Together? - Protocol — The People, Power and Politics of Tech," accessed November 10, 2021, https://www.protocol.com/-vr-headset-future-of-work.
5. Thumm' 'Jean-Luc and McAvoy' 'Scott, "GeiselVR," April 12, 2018.
6. Chaowanan Khundam et al., "A Comparative Study of Interaction Time and Usability of Using Controllers and Hand Tracking in Virtual Reality Training," Informatics 8, no. 3 (September 1, 2021), https://doi.org/10.3390/informatics8030060.
7. "Leap Motion Hand Tracking vs Oculus Quest One vs Vive One | PaleBlue," accessed November 10, 2021, https://pale.blue/2020/07/09/leap-motion-hand-tracking-vs-oculus-quest-one-vs-vive-one/.
8. "Tracking | Leap Motion Controller | Ultraleap," accessed November 10, 2021, https://www.ultraleap.com/product/leap-motion-controller/.
9. "Welcome to Steamworks," accessed May 11, 2022, https://partner.steamgames.com/vrlicensing.
10. Base Stations & VR Arcades. The Basics, The Do's, and The Don'ts | by SpringboardVR |Medium," accessed December 2, 2021, https://medium.com/@springboardVR/base-stations-vr-arcades-1dffac2abf05.
11. "Unity - Manual: Profiler Overview," accessed November 10, 2021, https://docs.unity3d.com/Manual/Profiler.html.

References

1. 'Jean-Luc, Thumm', and McAvoy' 'Scott. "GeiselVR," April 12, 2018.
2. "Base Stations & VR Arcades. The Basics, The Do's, and The Don'ts | by SpringboardVR | Medium." Accessed December 2, 2021. https://medium.com/@springboardVR/base-stations-vr-arcades-1dffac2abf05.
3. "Coronavirus Sent Us Home. Will VR Bring Us Back Together? - Protocol — The People, Power and Politics of Tech." Accessed November 10, 2021. https://www.protocol.com/vr-headset-future-of-work.
4. Holmes, D. E., D. K. Charles, P. J. Morrow, S. McClean, and S. M. McDonough. "Using Fitt's Law to Model Arm Motion Tracked in 3D by a Leap Motion Controller for Virtual Reality Upper Arm Stroke Rehabilitation." In Proceedings - IEEE Symposium on Computer-Based Medical Systems, 2016-August:335–36. Institute of Electrical and Electronics Engineers Inc., 2016. https://doi.org/10.1109/CBMS.2016.41.
5. 5Khundam, Chaowanan, Varunyu Vorachart, Patibut Preeyawongsakul, Witthaya Hosap, and Frédéric Noël. "A Comparative Study of Interaction Time and Usability of Using Controllers and Hand Tracking in Virtual Reality Training." Informatics 8, no. 3 (September 1, 2021). https://doi.org/10.3390/informatics8030060.
6. "Leap Motion Hand Tracking vs Oculus Quest One vs Vive One | PaleBlue." Accessed November 10, 2021. https://pale.blue/2020/07/09/leap-motion-hand-tracking-vs-oculus-quest-one-vs-vive-one/.
7. Nainggolan, F., B. Siregar, and F. Fahmi. "User Experience in Excavator Simulator Using Leap Motion Controller in Virtual Reality Environment." In Journal of Physics: Conference Series, Vol. 1566. Institute of Physics Publishing, 2020. https://doi.org/10.1088/1742-6596/1566/1/012093.
8. Singh, Ravi Pratap, Mohd Javaid, Ravinder Kataria, Mohit Tyagi, Abid Haleem, and Rajiv Suman. "Significant Applications of Virtual Reality for COVID-19 Pandemic." Diabetes and Metabolic Syndrome: Clinical Research and Reviews 14, no. 4 (July 1, 2020): 661–64. https://doi.org/10.1016/J.DSX.2020.05.011.
10. "Tracking | Leap Motion Controller | Ultraleap." Accessed November 10, 2021. https://www.ultraleap.com/product/leap-motion-controller/.
11. "Unity - Manual: Profiler Overview." Accessed November 10, 2021. https://docs.unity3d.com/Manual/Profiler.html.
12. "Welcome to Steamworks." Accessed May 11, 2022. https://partner.steamgames.com/vrlicensing.

Acknowledgement
The research was supported by the NTP-HHTDK-21-0024 grant.