Head's Oscillations Simulation - Project Report

Etienne VATRY
HCI Project Report supervised by
Jean-Luc LUGRIN

ABSRACT Cybersickness is an obstacle to the development of virtual reality applications in any field. If the user experience is negative due to discomfort, it will hinder the adoption and advancement of virtual reality technologies. This can even have critical effects, for example in medical applications. Through this project, we aim to set up a system that will enable us to measure the impact, positive or otherwise, of simulating head oscillations during user movements. More specifically, we will be focusing on oscillations during walking or running. For this purpose, three virtual environments have been set up, with three different types of terrain. Tests can be carried out regularly to measure the user's discomfort in real time.

1. INTRODUCTION

The aim of this project was to create three virtual environments for virtual reality testing. The three environments are as follows:

- Flat terrain,
- Terrain with regular bumps,
- Terrain with noise-generated bumps.

At present, these lots are empty: they have no vegetation.

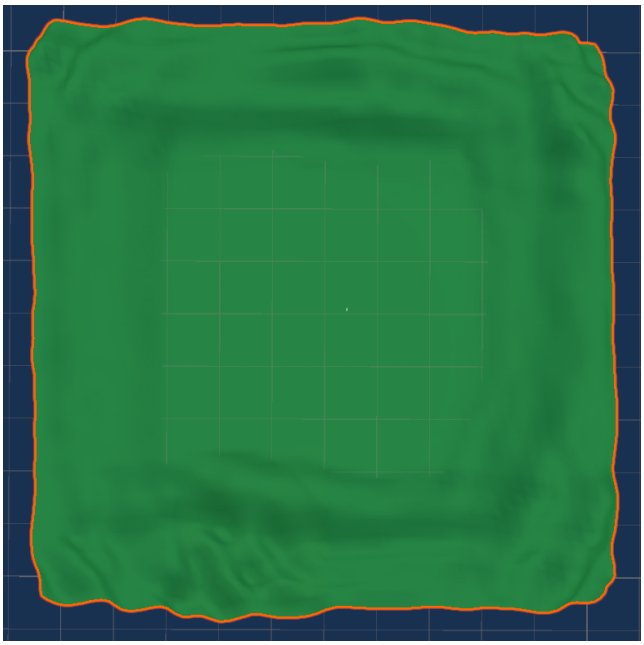

Fig. 1. Terrain from above.

etienne.vatry@stud-mail.uni-wuerzburg.de

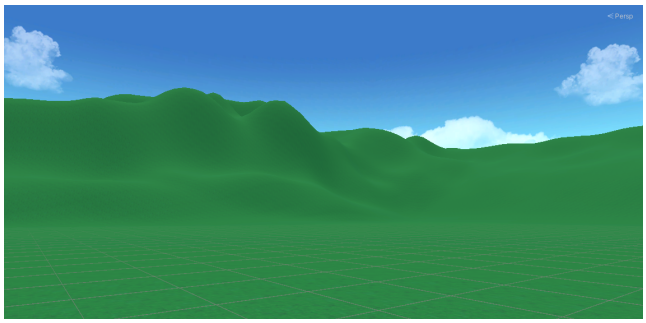

Fig. 2. Montains surrounding the terrain.

However, they have all been designed on the same basis - a large, flat surface surrounded by mountains. The choice to have three different terrains was motivated by a study showing that the more irregular the terrain, the greater the visual discomfort [QUARLES and ANG, 2022]. It may therefore be interesting to observe how this discomfort evolves with the addition of head oscillations. During a simulation, Fast Motion Sickness scales can be used at regular intervals to monitor the user's condition in real time. Step sounds have also been added. These sounds have also been shown to reduce cybersickness [KERN and ELLERMEIER, 2020,PENG et al., 2020]. Finally, the user's head oscillations during movement can be activated or deactivated, allowing the difference in user experience to be highlighted.

2. DESIGN AND DEVELOPMENT

2.1 Tools and Technologies

2.1.1 *Unity Engine.* To create these different virtual environments, we used Unity Engine. This choice was motivated by the flexibility of this tool, the presence of already complete templates, the wide variety of packages available, as well as its popularity. This popularity guarantees the availability of numerous resources and an active online community.

2.1.2 *Other Tools.* C# has been chosen as the programming language for the various scripts. This is the language used by default on Unity. Sounds are in wav files.

2.2 Methodology

2.2.1 *Research and Planning.* The first part of this project focused on two main points:

- Getting to grips with Unity Engine,
- Various research studies.

These steps gave me an initial idea of what needed to be done during the project. At the end of these two stages, a first provisional schedule was created (see Figure 3). Its aim was to be as realistic as possible to ensure steady progress in the project. In addition,

Handed in: July 2024.

bi-monthly meetings were organized to avoid the tunnel effect and understand how to improve the project.

Week	17/06 - 23/06	24/06 - 30/06	01/07 - 07/07	08/07 - 14/07	15/07 - 21/07
Objectives	- Reading/studying proposed courses - Study the use of curves in Unity Engine - Learn about virtual reality simulators	- Applying paper data to the virtual environment - Rendering tests with VR simulators - Research into adding sound	- Adding sound to simulation - Adding different types of environment - Final tests on simulators	- Final laboratory tests - Parameter adjustments based on test results	- End of project - Finalization of project report

Fig. 3. Initial project schedule.

2.2.2 *Development Process.* Once the schedule was established, the project could begin. The idea was as follows: first of all, we wanted to have a simulation that worked without virtual reality, so that we could fully understand how it worked. We therefore worked on the virtual environment with the flat terrain, the other two terrains being copies of the latter with added relief. The steps were as follows:

- Creating the environment,
- Add displacements,
- Adding oscillations,
- Add tests during simulation,
- Adding step noises.

Secondly, we had to adapt this to virtual reality. We therefore had to change the camera and movement controls. The rest remained more or less unchanged.

2.3 Environment Design

As described in the introduction (1), the terrain was created from a research paper [QUARLES and ANG, 2022]. The bumps of the second and third terrains are generated when the simulation is launched, enabling the parameters to be modified before the simulation begins. The three sections below describe these different environments in greater detail:

2.3.1 *Flat Terrain.* This terrain serves as the basis for the design of the other two. Apart from the mountains surrounding the surface where the user can evolve, there is no relief.

2.3.2 *Terrain with Regular Bumps.* For this terrain, regular linear bumps have been added at regular intervals. Their height and spacing can be modified to test different types of bump.


```
1 reference
void GenerateRegularBumps() {
    TerrainData terrainData = terrain.terrainData;
    int width = terrainData.heightmapResolution;
    int height = terrainData.heightmapResolution;

    float[,] heights = terrainData.GetHeights(0, 0, width, height);

    for (int x = 0; x < width; x++) {
        for (int y = 0; y < height; y += (int)bumpSpacing) {
            // Calculate height using a sinusoidal function to create a half-cylinder shape
            float heightValue = Mathf.Sin((float)x / width * Mathf.PI) * bumpHeight / terrainData.size.y;
            heights[x, y] += heightValue;
        }
    }

    // Smooth the heights
    SmoothHeights(heights, width, height);

    terrainData.SetHeights(0, 0, heights); // Apply the modified heights to the terrain
}

1 reference
void SmoothHeights(float[,] heights, int width, int height) {
    int smoothRadius = 1; // Smoothing radius

    for (int x = smoothRadius; x < width - smoothRadius; x++) {
        for (int y = smoothRadius; y < height - smoothRadius; y++) {
            float totalHeight = 0;
            int count = 0;

            for (int i = -smoothRadius; i <= smoothRadius; i++) {
                for (int j = -smoothRadius; j <= smoothRadius; j++) {
                    totalHeight += heights[x + i, y + j];
                    count++;
                }
            }

            heights[x, y] = totalHeight / count; // Average height within the smoothing radius
        }
    }
}
```

Fig. 4. Regular bumps code.

2.3.3 *Terrain with Noise-Generated Bumps.* These bumps are generated from Perlin noise. Their purpose is to reproduce ”natural” bumps. The height of the bumps and the intensity of the noise can also be modified.

```
1 reference
void GenerateIrregularBumps() {
    TerrainData terrainData = terrain.terrainData;
    int width = terrainData.heightmapResolution;
    int height = terrainData.heightmapResolution;

    float[,] heights = terrainData.GetHeights(0, 0, width, height);

    for (int x = 0; x < width; x++) {
        for (int y = 0; y < height; y++) {
            float xCoord = (float)x / width * noiseScale;
            float yCoord = (float)y / height * noiseScale;
            float noiseValue = Mathf.PerlinNoise(xCoord, yCoord) * noiseHeightMultiplier / terrainData.size.y;

            // Limit height variations
            heights[x, y] += Mathf.Clamp(noiseValue, 0, 0.05f); // Adjust clamping range as needed
        }
    }

    terrainData.SetHeights(0, 0, heights); // Apply the modified heights to the terrain
}
```

Fig. 5. Irregular bumps code.

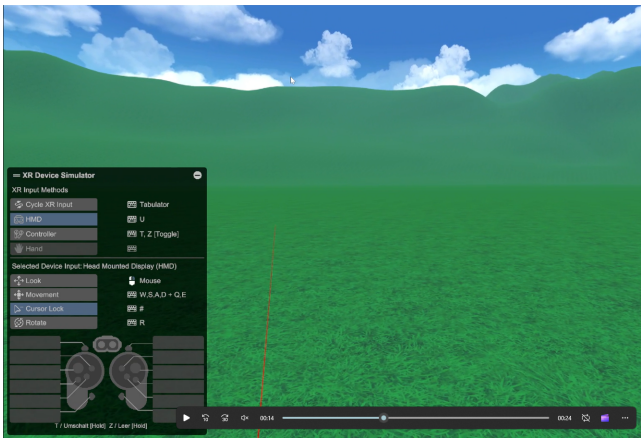

Fig. 6. Flat terrain.

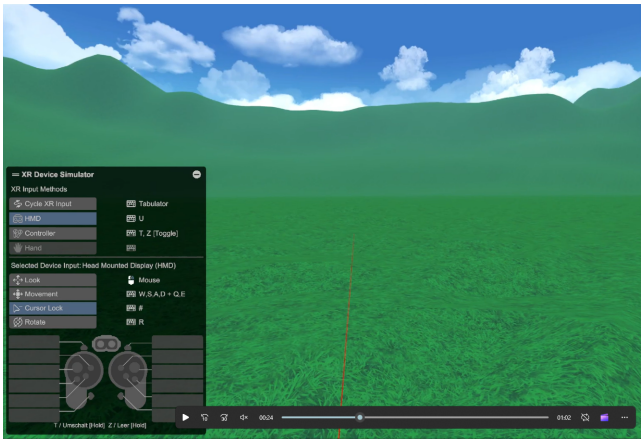

Fig. 7. Bumpy terrain.

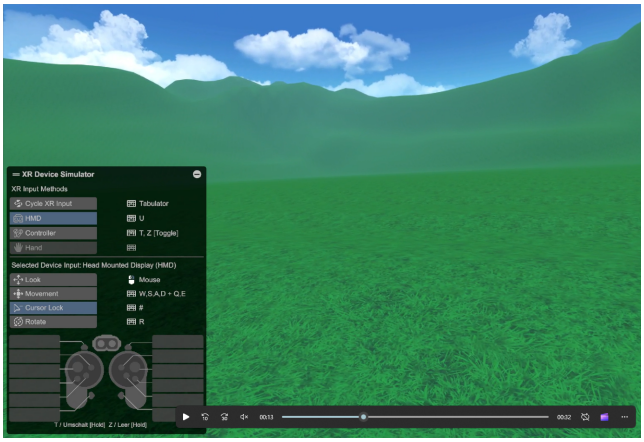

Fig. 8. Terrain with regular bumps.

2.4 Head Oscillation Simulation

2.4.1 *Research on Head Oscillations.* Initially, the data used for head movements were taken from the [WATERS, MORRIS, and PERRYS, 1973] research paper. These data were obtained during outdoor shots, not in a virtual environment. We quickly realized that the movements obtained did not render well in the virtual environment. We therefore chose another research paper, [SARUPURI et al., 2017], as the basis for our head oscillations. In this paper, data were taken while the subject was in the virtual environment.

2.4.2 *Implementation.* To implement the head movements, we used Unity Engine's "Animator" module. With the help of curves, we were able to fine-tune head oscillations during the simulation. Two types of animation were implemented: one for walking and one for running. Switching from one to the other is done using Boolean variables, managed in the code and updated as the user moves. Figures 9 and 10 show head movements in all three spatial directions. Figure 11 shows the relationships between states and the variables that allow you to move from one state to another.

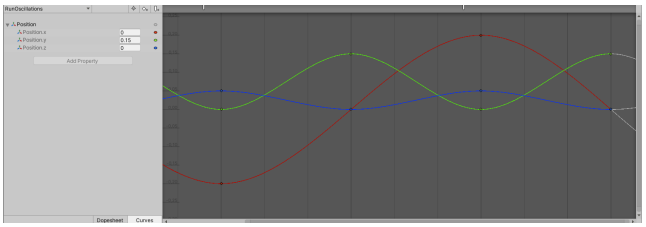

Fig. 9. Curves for the running animation.

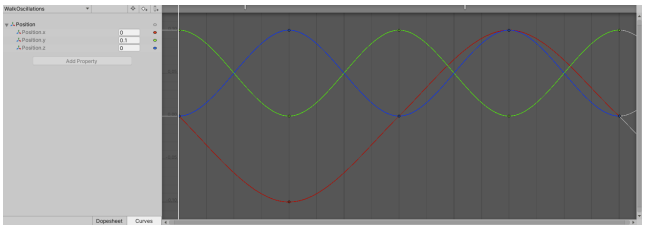

Fig. 10. Curves for the walking animation.

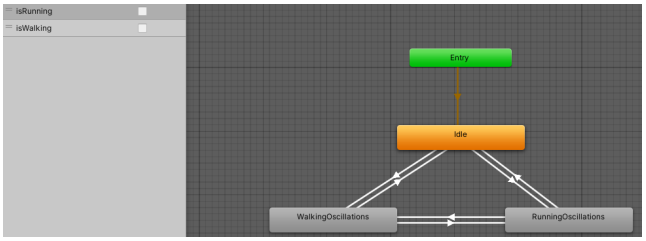

Fig. 11. Links between the different states.

Handed in: July 2024.

2.5 Additional Features

2.5.1 *Step Sounds.* In the [PENG et al., 2020] research paper, it was shown that the presence of footsteps when the user is moving contributes to the reduction of cybersickness. That is why we added this feature, with the option of activating it or not during the simulation. To do this, we searched for an extract of the sound of footsteps on grass in WAV format. We then modified it to the desired duration and applied it to certain events in our animation using a function in our script. Synchronization with head movements was easily achieved using the "Animator" tool.

2.5.2 *Real-Time Monitoring.* To monitor changes in the user's condition in real time, we chose to use Fast Motion Sickness scales [LUGRIN, 2024]. There are 7 comfort levels, ranging from 0 (very uncomfortable) to 6 (very comfortable). These appear at given intervals, adjustable from the script. To retrieve test data, a file bearing the scene name is created, and each test result is written to it. These are hidden at the start of the simulation and only appear during the tests. They are placed in a fixed position in the background and appear in front of the user while his or her movements are deactivated.

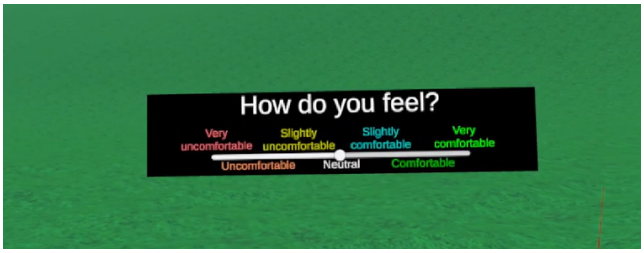

Fig. 12. FMS test.

```
1   (Time: 34,95173) FMS Test Number 1: 4
```

Fig. 13. One line of a text file containing test results.

3. FINAL PRODUCT DESCRIPTION

3.1 Application overview

The final simulation therefore has the following elements:

- Three different virtual environments,
- The ability to activate head oscillations,
- The ability to activate footsteps sounds,
- Visual comfort tests launched at regular intervals,
- The results of these tests are retrieved in a text file for analysis.

In addition, the duration of the simulation can also be set from within the script.

3.2 User controls

The controls are as follows:

- Walk: right OR left trigger button
- Run: right AND left trigger button
- Increase slider value during test: right trigger button
- Decrease slider value during test: left trigger button
- Validate test: A button
- Activate/deactivate oscillations: right grip button
- Activate/deactivate footstep sound during oscillations: left grip button (note: footstep sound can only be heard when oscillations are activated)

See figure 14 for controls.

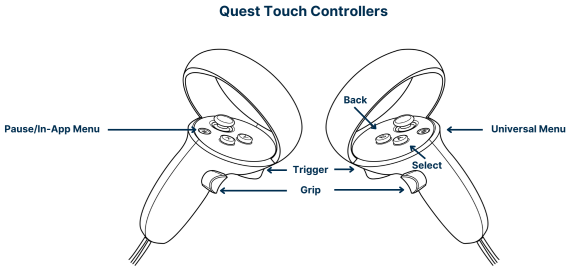

Fig. 14. Quest touch controllers.

4. CONCLUSION

The aim of this project was to create a versatile simulation to measure the impact of head oscillations on user comfort in virtual reality environments. By implementing three distinct virtual environments - a flat terrain, a terrain with regular bumps and a terrain with noise-generated bumps - we provided a comprehensive basis for future tests and studies.

Key features such as the ability to activate or deactivate head oscillations, the integration of footsteps and real-time monitoring of user discomfort using Fast Motion Sickness scales have been implemented. These features will provide a detailed understanding of how to enhance the virtual reality user experience.

The ability to capture and analyze user comfort data in real time provides valuable information that can be used to refine and improve virtual reality applications in a variety of fields, including medical, gaming and training simulations. The modular design of the simulation allows it to be adapted and extended for future research and development. I look forward to learning about these and following the various studies that may result.

In conclusion, this project has not only enabled me to develop a good tool for research in virtual reality, but it has also enabled me to learn many new concepts. Whether it is handling the Unity Engine or understanding the principles of virtual reality, this new knowledge will certainly benefit me.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Jean-Luc Lugrin, my supervisor, for his invaluable help, advice and support throughout this project. His expertise was crucial to the success of this work. Thank you for your continuous guidance and encouragement.

List of Figures

1 Terrain from above. . . . . .
2 Montains surrounding the terrain. . . . . .

Handed in: July 2024.

3 Initial project schedule. . . . . .
4 Regular bumps code. . . . . .
5 Irregular bumps code. . . . . .
6 Flat terrain. . . . . .
7 Bumpy terrain. . . . . .
8 Terrain with regular bumps. . . . . .
9 Curves for the running animation. . . . . .
10 Curves for the walking animation. . . . . .
11 Links between the different states. . . . . .
12 FMS test. . . . . .
13 One line of a text file containing test results. . . . .
14 Quest touch controllers. . . . . .

References

KERN, A. C., & ELLERMEIER, W. (2020). Audio in vr: Effects of a soundscape and movement-triggered step sounds on presence. *frontiers in Robotics and AI*, *7*(20). https://doi.org/10.3389/frobt.2020.00020 (cit. on p.).

LUGRIN, J.-L. (2024). Navigation [Course notes]. (Cit. on p.).

PENG, Y.-H., YU, C., LIU, S.-H., WANG, C.-W., TAELE, P., YU, N.-H., & CHEN, M. Y. (2020). Walkingvibe: Reducing virtual reality sickness and improving realism while walking in vr using unobtrusive head-mounted vibrotactile feedback. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3313831.3376847 (cit. on p.).

QUARLES, J., & ANG, S. (2022). You're in for a bumpy ride! uneven terrain increases cybersickness while navigating with head mounted displays. *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 428–435. https://doi.org/10.1109/VR51125.2022.00062 (cit. on p.).

SARUPURI, B., STEINICKE, F., HOERMANN, S., & LINDEMAN, R. W. (2017). Triggerwalking: A biomechanically-inspired locomotion user interface for efficient realistic virtual walking. *Proceedings of SUI*, *7*, 138–147. https://doi.org/10.1145/3131277.3132177 (cit. on p.).

WATERS, R. L., MORRIS, J., & PERRYS, J. (1973). Translational motion of the head and trunk during normal walking. *J. Biomechanics*, *6*, 167–172 (cit. on p.).