Designing, Testing and Evaluating 3D User Interfaces for 3D Navigation

Francisco Cerqueira
up201905337@edu.fe.up.pt

Luís Guimarães
up202204188@edu.fe.up.pt

Fabio Huang
up201806829@edu.fe.up.pt

Vasco Alves
up201808031@edu.fe.up.pt

***Abstract*—This paper presents a comprehensive exploration of the development and evaluation of two distinctive 3D navigation systems, WeLeap and WiiFly, addressing the growing demand for intuitive and efficient navigation mechanisms within 3D environments. WeLeap utilizes Leap Motion technology to capture hand and finger motions, enabling 5 degrees of freedom (5-DOF) movement through hand gestures. WiiFly extends the functionalities of the Wii remote for intuitive 3D navigation, offering 5-DOF control. Additionally, we also develop an evaluation method to gather feedback on each approach, which includes executing tasks of different natures and filling out a System Usability Scale (SUS) questionnaire. Finally, we compare them based on four criteria: usability, precision, reliability, and speed.**

***Index Terms*—5-DOF, data analysis, graphical user interfaces, human-computer interaction, input devices, performance evaluation, user experience**

I. Introduction

As three-dimensional (3D) environments expand their presence in diverse fields like gaming, virtual reality, architectural visualization, and simulation, the need for intuitive and efficient navigation mechanisms becomes increasingly vital. The concept of degrees of freedom (DOF) holds significant importance in these spaces, defining the spatial movement capabilities by representing the independent parameters that determine the configuration or state of a rigid body [1]. As a result, tailoring navigation tools to accommodate the unique demands and degrees of freedom within these 3D environments becomes imperative.

We were tasked with the challenge of developing two approaches for 3D navigation intended for execution on a computer using Unity 2022.3.11f1. These approaches were designed in accordance with a specific set of requirements:

- Enabling **5 degrees of freedom** (5-DOF) movement, as the Roll angle was discarded.
- Allowing users to switch between two distinct camera modes: **Navigation/Fly mode**, granting unrestricted navigation in all directions and the ability to rotate independently, and **Orbit mode**, anchoring the user at an arbitrary center while enabling translation and rotation around it.
- Adhering to, at most, a **single trigger action**, considered to be any action with a binary state, such as keys or buttons.
- Providing users with the flexibility to dynamically **adjust the movement speed**.

In this work, we developed two distinct approaches to tackle the challenge and created an evaluation method to compare the two approaches based on user experience. Finally, we processed the collected data to obtain comparative results of the developed approaches.

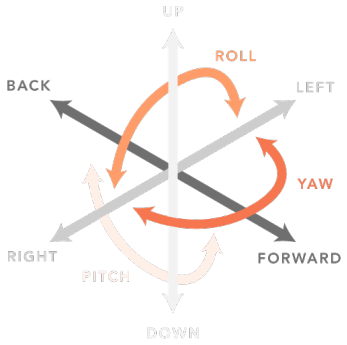


Fig. 1: Illustration depicting the range of motion for a rigid body in a 3D space, showcasing its ability to alter orientation (yaw, pitch, and roll) and position along the X, Y, and Z axes. Source: [2]

This paper is structured as follows: Sections II and III present the proposed systems to address the given challenge, Section IV outlines the user evaluation process to compare both approaches and Section V delves into the analysis and discussion of the results to determine the superior system. Lastly, Section VI discusses the drawn conclusions and outlines future avenues of work.

II. WeLeap

Leap Motion technology, now known as *Ultraleap*, is a hand-tracking system that captures hand and finger motions without requiring contact. The Leap Motion system can be used for various applications, including gaming [3], music performance [4], virtual and augmented reality, etc.

This approach explores the utilization of Leap Motion technology to design an experimental navigation system intended for virtual 3D environments. We integrate 5-DOF movement using hand gestures and devise a more intricate solution that demands greater levels of coordination. Yet, its effectiveness hinges on the user's ability to surpass the technical hurdles. The project repository was made publically available.[1]

[1]https://github.com/FabioMiguel2000/Leap-Motion-3D-Navigation

Fig. 2: WeLeap hardware component: Leap Motion Controller. Source: [5]

A. Technical Specifications

The hardware of the WeLeap system comprises the **Leap Motion Controller**, an optical hand-tracking sensor device equipped with a camera that captures hand movements, communicating with the computer through a USB-A connection.

Regarding software, the WeLeap system relies on the following elements:

- **Ultraleap Hand Tracking Service (v5)**: This service interprets the intricate motions executed by users through the Leap Motion Controller, transforming them into a comprehensive 3D input array.
- **Ultraleap Plugin for Unity**: This plugin is a Unity package [6] that allows communication between Unity and the Ultraleap Hand Tracking Service.

Figure 3 shows a diagram of the WeLeap architecture. The Leap Motion Controller connects to the computer via USB-A and provides information to the Ultraleap Hand Tracking Service. This service communicates with Unity via the Ultraleap Plugin for Unity, which is responsible for all the input processing and rendering.

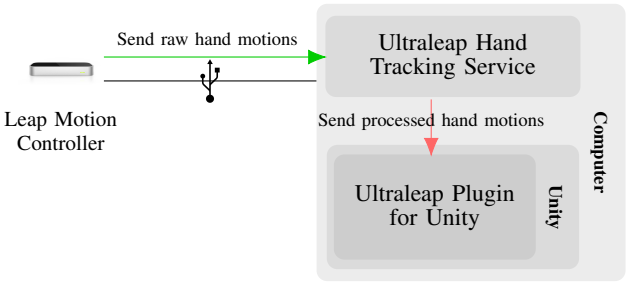


Fig. 3: WeLeap architecture diagram. The tracking service processes the data from the Leap Motion Controller and sends it to the Ultraleap plugin.

B. Navigation Control

1) Navigation Mode: In Navigation mode, the **left hand** manages the left/right and forward/backward translation through tilting movements – left/right for lateral motion and forward/backward for longitudinal movement. Additionally, it facilitates vertical translation by raising or lowering the hand. On the other hand, the **right hand** oversees camera rotation in accordance with its vector orientation. For instance, when the palm's normal aligns perpendicularly to the screen, indicating an upward hand vector, the camera rotates in an upward direction.

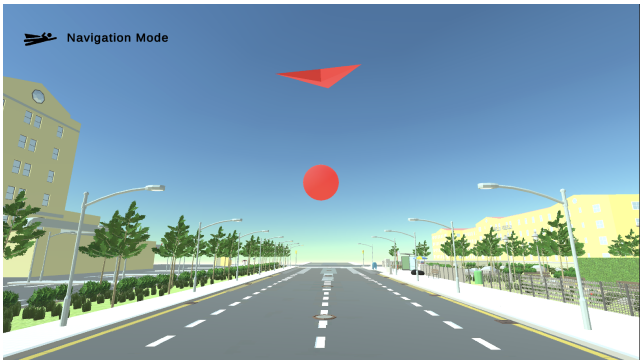


Fig. 4: WeLeap's User Interface. It comprises an orientation arrow, an orbiting center sphere, and a mode widget.

The translation and rotation speeds can be dynamically adjusted based on real-time changes and are directly proportional to the distance from the neutral position. For example, as the hand tilts forward, the forward translation accelerates, and as the x-coordinate of the hand vector increases, the camera's upward rotation speed intensifies.

2) Orbit Mode: In Orbit mode, the functionality is limited to rotation and forward/backward translation. Consequently, the **left hand** exclusively manages forward and backward translation, while the **right hand** operates similarly to the navigation mode, but it orchestrates rotations around the orbit center instead of around the user.

3) Mode Switching: Toggling between navigation and orbit modes occurs through a **pointing gesture**.

In navigation mode, the gesture triggers the creation of an indicator that continuously moves outward in the direction the camera faces until the gesture ends. Upon the gesture's cessation, the mode switches to orbit, with the indicator serving as the new center for orbiting.

When in orbit mode, using the pointing gesture followed by its reversal ("unpointing") removes the indicator and reverts the mode back to navigation.

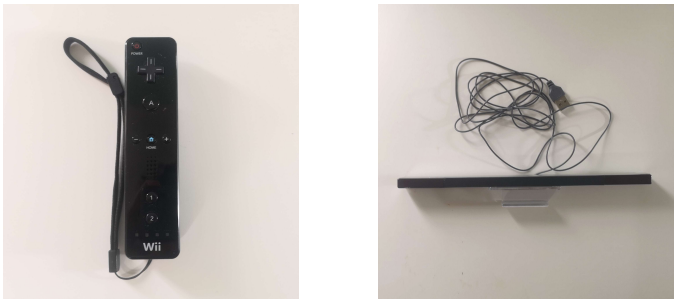
C. User Interface

The User Interface of this approach, depicted in Figure 4, consists of the following elements:

- An **orientation arrow** positioned at the top-center of the screen. This arrow indicates the direction of player translation, aligning with the movement direction. It appears only when the translation speed exceeds zero.
- An **orbiting center sphere** located at the screen's center. This sphere is a visual marker for the orbit center, specifically during orbit mode.
- A **mode widget** positioned at the top-left corner of the screen. This widget provides a clear indication of the current navigation mode in use.

D. Technical Details

This method incorporates "deadzones" for each degree of freedom, enabling easier attainment of a neutral state where



(a) Wii remote. (b) Sensor bar.

Fig. 5: WiiFly hardware components.

neither translation nor rotation occurs. This neutral stance is intentionally set slightly away from value 0 to accommodate the natural tilt of relaxed hands. Additionally, we have established boundaries for the controllable degrees of freedom, which notably enhances the system's controllability.

During the implementation phase, we observed that long sleeves frequently disrupt hand recognition and that the pointing gesture detection is inconsistent. Furthermore, the system's optimal functionality depends on appropriate environmental lighting conditions.

III. WiiFly

Previous research [7, 8, 9] has already focused on using the Wii remote to interact with 3D environments. Building upon this foundation, our work involves an innovative extension of its functionalities, transforming it into an intuitive interface for users to navigate in sophisticated 3D navigational systems, encompassing a 5-DOF control mechanism. The project repository was made publically available.[2]

A. Technical Specifications

The hardware of the WiiFly system comprises the following components:

- **Wii remote (Wiimote):** This serves as the input device for controlling the WiiFly system. Equipped with motion sensors (gyroscope and accelerometer), an infrared camera, and various buttons, the Wii remote accurately tracks the movements and gestures of the user, communicating with the computer through a Bluetooth connection.
- **Sensor bar:** The Sensor Bar, specifically the *Xahpower USB Sensor Bar*, ensures precise tracking of the Wii remote's orientation. With a total of six LEDs – three on each side – it emits infrared (IR) light that allows the Wii remote's IR camera to determine the cursor's position precisely. As it conveniently plugs into any USB-A port, it does not necessarily require a connection to the hosting computer.

Regarding software, the WiiFly system relies on the following elements:

- **Wiimote API:** The *Unity Wii Remote API* [10] is a crucial component enabling the integration of Wii remote capabilities within the Unity environment. This API establishes a channel for communication between the Wii remote and the Unity application, processing the remote inputs to high-level data structures and configuring the appropriate device settings.
- **WiiPair:** The software application *WiiPair* [11] facilitates Bluetooth connection setup between the Wii remote and a computer, ensuring ongoing communication stability between the two devices.

The Bluetooth link between the Wii remote is set up via the WiiPair software, sending an authentication token to maintain the connection. In Unity, the Wiimote API interacts with the remote, configuring it and awaiting to receive input events. The remote's motion is tracked using the IR light from the sensor bar connected to a USB-A port. A visual representation of this architecture is illustrated in Figure 6.

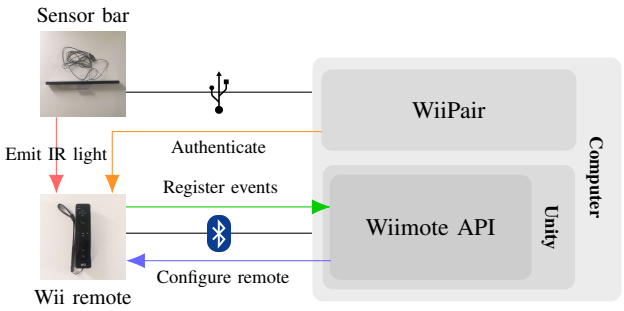


Fig. 6: WiiFly architecture diagram. The Wii remote uses the IR light from the sensor bar to track its motion and sends that information to the Wiimote API.

B. Navigation Control

The position and orientation of the camera are controlled by the Wii remote, which uses the sensor bar to keep track of its motion.

1) Camera Orientation:

a) Fly Mode: In Fly mode, the remote's orientation directly influences the camera's orientation. A visible cursor on the screen signifies the direction towards which the remote is pointed. The position P of the cursor directly impacts the camera's angular speed ω, as outlined by Equations 1 and 2. Here, λ_x and λ_y represent the maximum horizontal and vertical angular speeds, respectively. Both P_x and P_y are confined within the range $[-1, 1]$, where negative values indicate the left or top part of the screen and positive values indicate the right or bottom part of the screen.

$$\omega_x = \lambda_x \times P_x \tag{1}$$

$$\omega_y = \lambda_y \times P_y \tag{2}$$

[2]https://github.com/xico2001pt/wiifly-3d-navigation

b) Orbit Mode: Similar to Fly mode, the cursor position dictates the direction in which the camera rotates. However, in this mode, the camera does not revolve around its own axis but rather around a designated center point. For example, if the cursor is on the left side of the screen, the camera will orbit around the center point at full speed to the left.

2) Camera Translation: The camera's translational movement is exclusively relative to its orientation, whereas forward and backward motion are the only achievable directions. The linear velocity of this movement is directly governed by the distance between the remote and the sensor bar.

At a neutral distance, the linear speed of the camera registers as zero. As the distance deviates from this neutral point, the camera's motion changes accordingly: moving farther away from the sensor bar induces a backward camera motion, whereas approaching the sensor bar initiates forward movement. Although the Wiimote API lacks documentation regarding the units of measurement for this distance, empirical observations have indicated its linear behavior. As a result, Equation 3 can be employed to depict the correlation between the distance d and the linear speed v of the camera. Here, k denotes a constant determining the speed factor, while N_d signifies the distance at which the velocity should reach zero.

$$v = k \times (N_d - d) \tag{3}$$

3) Mode Switching: While in Fly mode, activating the Orbit mode is achieved by pressing the "A" button on the Wii remote while pointing toward the desired surface to orbit around. If an intersection point is found, the mode transitions to Orbit mode. Pressing the button again will revert to the Fly mode.

C. User Interface

The WiiFly system's User Interface, depicted in Figure 7, comprises the following elements:

- A **cursor** indicating the virtual position where the Wii remote points concerning the sensor bar.
- An **auxiliary grid** featuring a central circumference aiding users in associating screen regions with directional movements (left, right, top, bottom). The circumference defines the "dead zone" where the camera's orientation remains static.
- A **velocity bar** to visually represent the camera's linear speed.
- A **mode widget** to display the current movement mode.

Moreover, a red cube appears upon activating the orbit mode, symbolizing the selected orbit center.

D. Implementation Details

1) Cursor Position:

a) Smoothing: Raw input data retrieved from hardware devices, like the Wii remote, often suffer from inherent noise. This noise can stem from various sources, such as sensor inaccuracies, environmental interferences, or rapid movements. However, implementing the Exponential Moving Average (EMA) technique effectively addresses this issue. EMA operates by computing a weighted average of the incoming data over time, assigning higher significance to recent readings and diminishing the impact of older ones, resulting in a more stable and accurate representation of the cursor's position. This smoothing process significantly enhances the **reliability** of the data obtained from the Wii remote, enabling a more precise tracking experience for users.

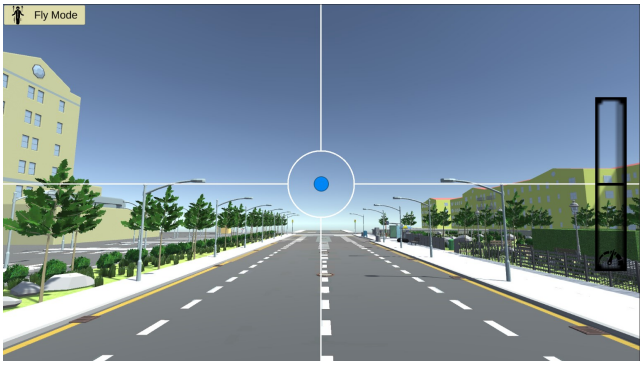


Fig. 7: User Interface of the WiiFly system. It comprises a cursor, an auxiliary grid, a velocity bar, and a mode widget.

b) Virtual Zoom: The Wii sensor bar's signals weaken at the edges because the remote captures fewer infrared rays in those zones. This reduction in captured rays leads to decreased tracking accuracy. Implementing a virtual zoom technique adjusts the remote's sensitivity, compensating for the reduced rays at the edges. This technique decreases the rotation range that the remote needs to reach to be at the edge of the screen. Nonetheless, this adjustment sacrifices some freedom of movement, necessitating precise calibration for optimal performance.

c) Default Behaviour: If the signal is lost and the cursor remains in the same position, the user may lose control of the movement, leading to unexpected and uncontrolled motion. To address this, the default behavior for the cursor position, when losing the signal, involves a controlled repositioning towards the center of the screen.

2) Remote Distance:

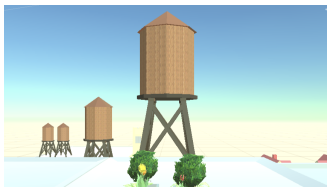
a) Signal Intensity: The Wii remote determines the distance to the sensor bar by analyzing the intensity of the emitted IR light. Since the remote captures up to four rays, it computes the intensity by averaging the values obtained from each individual ray.

b) Smoothing: As explained in Section III-D1, the input data acquired from the Wii remote is affected by inherent noise, including the recorded IR light. Employing EMA helped to address this issue, leading to a more consistent depiction of the distance of the remote.

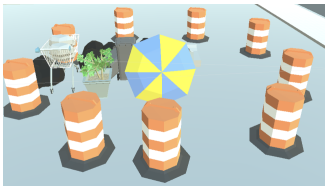
c) Default Behaviour: If the signal between the Wii remote and the sensor bar becomes weak or is lost, it is essential to prevent sudden jumps or inaccuracies in tracking. To address erratic behavior, the standard procedure includes readjusting the distance value to a mid-range position where the camera's velocity remains zero.

3) Movement Stability: Integrating "deadzones" becomes crucial in ensuring movement stability, as they establish a

(a) Gas station.



(b) Water tanks.



(c) Circular barrels.

(d) Garden's main entrance.

Fig. 8: Simple tasks locations. Scene created by *255 Pixel Studios* [12].

designated area centered on the input range, permitting minor movements or shaking without immediate system response. By implementing this mechanism, users can navigate with a degree of tolerance, allowing for subtle hand tremors or slight shifts without triggering significant actions. This technique was used for camera rotation and speed control. Additionally, we established boundaries for the controllable degrees of freedom, which notably enhances the system's controllability.

4) Mode Switching: Identifying the orbiting focal point operates by casting a ray through the screen's center and identifying the first object it collides with. For this process to work, the camera and scene entities must have an associated Unity "Collider" component. If no collision point is detected, the mode remains unchanged.

E. Additional Note

Initially, the WiiFly approach intended to leverage the gyroscope to capture motion from the Wii remote. However, this approach swiftly revealed its instability, as the gyroscope method required frequent recalibration due to its erratic behavior, proving unreliable for consistent motion tracking.

IV. User Evaluation

The developed user evaluation method aims to gather user feedback on various aspects of the experience:

- **Usability:** assessing the ease and intuitiveness of the solution.
- **Precision:** evaluating whether the solution enables accurate motion.
- **Reliability:** examining the consistency between real-world input and virtual output.
- **Speed:** determining if the solution facilitates swift motion and effectively controls speed.

The experimental protocol devised involves the participants engaging in both approaches (intra-group testing). Half of the participants would test one system first, while the other half would start with the other. At least one supervisor for each system must be present throughout the testing.

(a) Start region.

(b) Finish region.

Fig. 9: Map containing the circular circuit checkpoints (green) and finish line (red). Scene created by *Synty Studios* [13].

A. Experiment Protocol

The experimental procedure comprises three phases: gathering personal information, assisting users in system utilization, and evaluating their performance upon completing predetermined tasks.

The first phase aims to collect information from the user that can be relevant to identify correlations in the task results. This stage is conducted at the beginning of the testing session and only occurs once, assigning the following tasks to the user:

1) Complete an **informed consent form** wherein the user agrees to share information relevant solely to the experiment and assures anonymization of their identity.
2) Complete a **personal information form**, requesting details such as age, gender, and experience with video games.

The subsequent phases are conducted individually for each system. One of the systems is performed first, followed by the subsequent system (only after the participant completes all phases in the first system). The second phase aims to educate the participant on using the approach and is outlined as follows:

1) View a **video tutorial** illustrating the system's operation and the utilization of devices.
2) Engage in **unrestricted exploration** for approximately 3 minutes (or longer upon the participant's preference) to facilitate familiarity with the controls and overall experience. Participants are encouraged to seek clarification by asking questions.
3) Undertake **guided navigation**, encompassing the execution of all functionalities within the method, such as movement and rotation along various axes or orbiting around a focal point. In cases where the participant encounters difficulties, the supervisor is available to offer instructions.

The third and last phase focuses on collecting the test data by evaluating the user performance upon completing predesigned tasks and gathering feedback from a post-test questionnaire. This phase is organized as follows:

1) Execute a series of **"simple tasks"** specific to the map depicted in Figure 8. These tasks involve the user to:

TABLE I: Participants demography of our user evaluation.

ID	Age	Gender	Have you ever played videogames?	What kinds of games?
0	22	Male	Yes	FPS
1	22	Male	Yes	RPG, FPS, Strategy, Action
2	24	Female	Yes	MMO-RPG, Strategy, Card Game
3	22	Female	Yes	MMO-RPG, Platforming, Simulation
4	23	Male	Yes	RPG, FPS, Platforming, Strategy, Sandbox, Action, Simulation, Card Game
5	47	Male	Yes	RPG, FPS, Strategy, Sandbox
6	24	Female	Yes	-
7	22	Male	Yes	RPG, FPS, MOBA, Sandbox, Action, Simulation
8	24	Male	Yes	RPG, FPS, Platforming, Strategy, Sandbox
9	22	Male	Yes	RPG, Platforming, Action, Simulation

   a) Halt inside the **gas station**.
   b) Rotate around a **water tank**, completing a full rotation.
   c) Navigate through the **circular barrels**, necessitating delicate and precise movements.
   d) Reach the park garden through its primary entrance without colliding with the surrounding bushes.
2) Complete the **circuit challenge** outlined in the map illustrated in Figure 9. Participants must pass through every checkpoint.
3) Fill a **post-test questionnaire** to assess user experience and encountered difficulties, providing a section for gathering subjective feedback. This form adopted the System Usability Scale (SUS) template.

B. Experiment Requirements

The technical specifications for running WeLeap and WiiFly are detailed in sections II-A and III-A respectively.

Furthermore, in preparation, two Unity scenes need to be set up. The first scene is essential for the second phase and for executing the "simple tasks". The second map is specifically designated for the circuit task.

The initial system state necessitates additional prerequisites depending on the approach being tested. For WeLeap, participants should position their hands in a manner that minimizes movement and camera rotation. For WiiFly, participants must commence with the Wii remote positioned 50 centimeters perpendicularly from the sensor bar, corresponding to the point where the camera velocity is zero.

C. Participants

We performed **summative testing** with 10 participants, five of whom first tested the WiiFly approach and the other five the WeLeap approach. Details regarding the participant demography are included in Table I.

V. Results and Discussion

A. Task Performance

1) Simple Tasks: During this stage, we observed several aspects regarding the different approaches. Regarding WeLeap:

- Overall, the approach was technically demanding. While most participants couldn't fully master it during testing, a small subset of 2 participants could.
- Consequently, tasks requiring higher precision posed challenges for most participants and were impossible for a small subset but were manageable for those who mastered the technique.
- Achieving a neutral translation and rotation (stopping) was difficult for everyone.
- The recognition of pointing gestures was notably inconsistent.

For WiiFly:

- Participants easily grasped the concept.
- Individuals tended to over-rotate the controller. Increasing the distance could mitigate this issue.
- The maximum speed could be slightly higher.
- There was greater difficulty in mapping speed. Increasing the motion distance range could mitigate this. However, the test scenario was not optimal due to limited space.
- During the tests, the sensor bar was positioned on the right side of the computer. However, positioning it below the screen would have made it more intuitive.

Overall, the WiiFly approach can provide more **precision** to both technical and non-technical users.

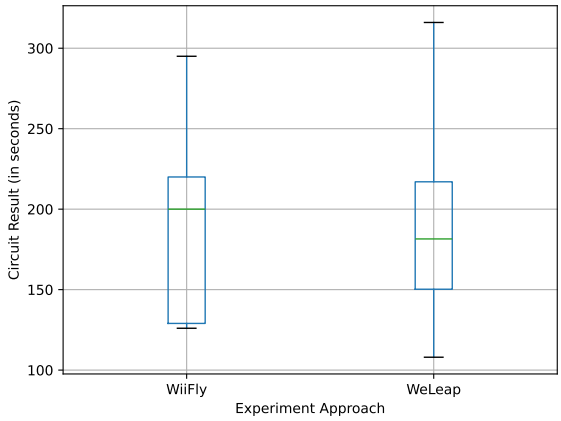


Fig. 10: Circuit completion time box plot representation.

TABLE II: Circuit task statistical analysis.

	WeLeap	WiiFly
Completion Rate (%)	80	90
Checkpoint Miss (%)	5.5	0.5
Mean (s)	189.750	187.222
Std. Deviation (s)	67.273	56.907
Median (s)	181.500	200.000

2) Circuit: The data extracted from Figure 10 and Table II demonstrates a higher level of **reliability** in the WiiFly approach. This is evident from the higher success rate, lower standard deviation, and narrower range between the maximum and minimum values compared to the WeLeap approach. While the WeLeap approach achieved the quickest run, indicated by the lowest minimum time, the average run time of the WiiFly approach is marginally faster due to a slightly lower mean time.

We performed a statistical test to accurately compare the **speed** between the two approaches. Specifically, this test focused on the data related to circuit completion time.

TABLE III: The p-values for the Shapiro-Wilk test and Paired T-test.

Test/Approach	WeLeap	WiiFly
Shapiro-Wilk (p-values)	0.7460	0.2710
Paired T-test (p-value)	0.2377	

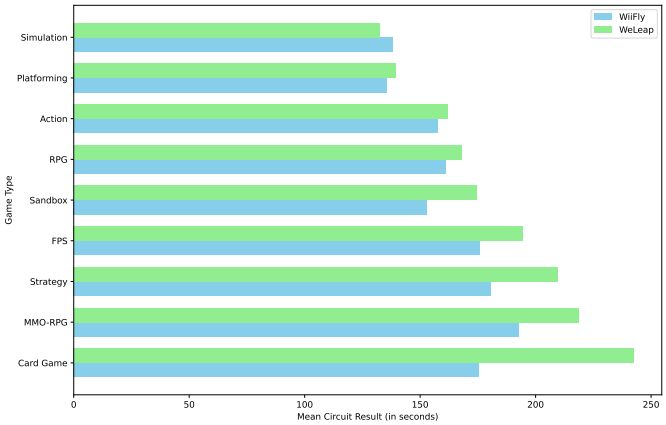


Fig. 11: Mean circuit time by game genre.

Prior to conducting the statistical test, the *Shapiro-Wilk Test* was employed to assess whether the results from both approaches exhibited a Normal distribution pattern. The p-values obtained for WeLeap and WiiFly were 0.746 and 0.2710 respectively. Since these values surpass the conventional significance level of 0.05, according to this test, there is insufficient evidence to reject the null hypothesis that the circuit completion time data conforms to a normal distribution.

Assuming a normal distribution of the data and the comparison between the two systems, the appropriate statistical test is the *Paired T-Test*. The p-value for this test was 0.2377, exceeding the commonly used significance level of 0.05. Consequently, it indicates weak evidence against the null hypothesis, suggesting that the difference in group means is not statistically significant.

In summary, the statistical test does not yield compelling evidence to support a significant difference between the means of the two groups under examination.

During this evaluation phase, a noteworthy observation was made regarding the WeLeap approach: participants tended to misinterpret the orientation arrow as the player entity within the virtual environment. Consequently, this confusion led to missed checkpoints, as participants inadvertently maneuvered the player entity in a way that caused it to bypass certain checkpoints.

3) Participant Characteristics: Figure 11 shows that participants playing platforming and simulation games have the best mean performance. The second best mean performance is by participants who play sandbox, RPG, and action. Interestingly, this pattern repeats in both approaches.

Also, Figure 12 shows that the more types of games participants play, the better their performance.

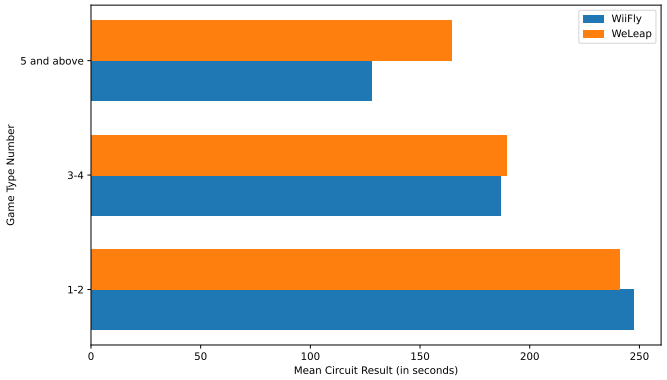


Fig. 12: Mean circuit time by the cardinality of game types.

B. User Preferences

Table IV displays the median scores for each question of the post-test questionnaire.

To derive the **SUS score** for a single participant, their scores for each question were transformed from the original range $[1, 5]$ to $[0, 4]$ and added together. Scores for questions where the lowest value indicates the optimal response (IDs 2, 4, 6, 8, and 10) were adjusted by assigning them their complementary values; in other words, a score s was reassigned to $4-s$. Lastly, each sum was multiplied by a factor of 2.5 to normalize the total score within the $[0, 100]$ range.

At the bottom of Table IV, the mean and median of the SUS score of all participants were calculated accordingly to Equations 4 and 5, where P denotes the number of participants, Q the number of questions, and $s_{i,j}$ represent the normalized score given by the participant i in the question j. For the median, we assume that the values were previously sorted.

$$\text{SUS Score Median} = \begin{cases} \sum_{j=1}^{Q} s_{\frac{P+1}{2},j} & \text{if } P \text{ is odd} \\ \frac{1}{2}\left(\sum_{j=1}^{Q} s_{\frac{P}{2},j} + \sum_{j=1}^{Q} s_{\frac{P}{2}+1,j}\right) & \text{otherwise} \end{cases} \quad (4)$$

$$\text{SUS Score Mean} = \frac{\sum_{i=1}^{P}\sum_{j=1}^{Q} s_{i,j}}{P} \quad (5)$$

For the WeLeap approach, the SUS score mean was 57.0, positioning the system usability below average, as a SUS score of 68 is considered the average [14]. Conversely, the SUS score mean for the WiiFly approach was 83.5, indicating an above-average performance by a considerable margin.

On average, in terms of **usability**, users find the WiiFly approach more intuitive, consistent and less cumbersome. For the WeLeap approach, some participants reported that the fact that the supported hand motions differed for both hands resulted in unintuitiveness.

VI. Conclusion

Developing a 3D user interface involves extensive effort, with testing and system evaluation being crucial components. This iterative process may necessitate adjustments to the

TABLE IV: User preferences table with the median score for each question. The values range from 1 (strongly disagree) to 5 (strongly agree), and for each question, the best score, which in some cases might be the lowest value, is highlighted in bold.

ID	Question	WiiLeap	WiiFly
1	I think that I would like to use this system	4.0	**5.0**
2	I found the system unnecessarily complex	2.0	**1.0**
3	I thought the system was easy to use	3.0	**4.0**
4	I think that I would need the support of a technical person to be able to use this system	3.5	**1.0**
5	I found the various functions in the system were well integrated	**4.0**	**4.0**
6	I thought there was too much inconsistency in this system	2.0	**1.0**
7	I would imagine that most people would learn to use this system very quickly	3.0	**5.0**
8	I found the system very cumbersome to use	2.5	**2.0**
9	I felt very confident using the system	3.0	**4.0**
10	I needed to learn a lot of things before I could get going with this system	3.0	**1.0**
	SUS Score Median	56.25	**86.25**
	SUS Score Mean	57.0	**83.5**

original design, leading to further rounds of testing. Two approaches may be designed and ideal for different purposes.

For example, one approach might prioritize precision but demand more user training, while another may be simpler to learn but less optimal for high-performance tasks. It is essential to consider the characteristics of the target users and tailor the system accordingly. In this research, WiiFly emerged as the most usable, precise, and reliable approach. However, when it comes to the speed factor, the statistical test yielded inconclusive results, making it challenging to determine the superior system.

In continuation of this project, for the WeLeap approach, we plan to conduct additional experiments and tests involving various hand motions. This aims to enhance usability for less technical users and improve precision for those with more technical expertise. As for the WiiFly approach, we are exploring the possibility of expanding it to a 6-degrees-of-freedom (6-DOF) system. This expansion could leverage the roll angle of the Wii remote, utilizing either the accelerometer or gyroscope. Adjustments to the system's rotation sensitivity, which requires further refinement, as well as the maximum speed and range, are also under consideration. Additionally, we intend to implement a calibration feature that allows users to set the neutral distance and speed range before initiating system use.

REFERENCES

[1] Nov. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics) (visited on 12/22/2023).

[2] D. by Onit, Feb. 2022. [Online]. Available: https://dragonflycv.com/what-does-6-degrees-of-freedom-or-6-dof-mean/ (visited on 12/21/2023).

[3] [Online]. Available: https://docs.ultraleap.com/ultralab/vr-gaming-controllers-and-hands.html.

[4] Apr. 2023. [Online]. Available: https://midipaw.com/.

[5] jefBinomed, *Devfest nantes 2014 - demos part 2 - leap motion*, Oct. 2018. [Online]. Available: https://jef.binomed.fr/2015/02/20/2015-02-20--DevFest-Nantes-2014-Demos-Part-2-Leap-Motion/.

[6] [Online]. Available: https://developer.leapmotion.com/unity.

[7] S. Sreedharan, E. S. Zurita, and B. Plimmer, "3d input for 3d worlds," in *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, 2007, pp. 227–230.

[8] L. Gallo and M. Ciampi, "Wii remote-enhanced hand-computer interaction for 3d medical image analysis," in *2009 International Conference on the Current Trends in Information Technology (CTIT)*, IEEE, 2009, pp. 1–6.

[9] M. Hoffman, P. Varcholik, and J. J. LaViola, "Breaking the status quo: Improving 3d gesture recognition with spatially convenient input devices," in *2010 IEEE Virtual Reality Conference (VR)*, IEEE, 2010, pp. 59–66.

[10] Flafla2, *Unity Wii Remote API*, 2022. [Online]. Available: https://github.com/Flafla2/Unity-Wiimote.

[11] J. B. Tucker, *WiiPair*, 2019. [Online]. Available: https://github.com/jordanbtucker/WiiPair.

[12] [Online]. Available: https://assetstore.unity.com/packages/3d/environments/urban/city-package-107224 (visited on 01/11/2024).

[13] [Online]. Available: https://assetstore.unity.com/packages/3d/environments/urban/polygon-city-low-poly-3d-art-by-synty-95214 (visited on 10/26/2023).

[14] A. S. f. P. Affairs, *System usability scale (sus)*, Sep. 2013. [Online]. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (visited on 01/10/2024).