KEYWORDS Virtual Reality; Full Body; Full Room; GUR; Analytics; Visualization; Tool

FRVRIT - A Tool for Full Body Virtual Reality Game Evaluation

Daniel MacCormick, Alain Sangalang, Jackson Rushing, Ravnik Singh Jagpal, Pejman Mirza-Babaei & Loutfouz Zaman
University of Ontario Institute of Technology
Oshawa, Ontario, Canada
daniel.maccormick@uoit.net,alain.sangalang@uoit.net,jackson.rushing@uoit.net,ravnik.jagpal@uoit.net,pejman@uoit.ca,loutfouz.zaman@uoit.ca

ABSTRACT

Virtual reality (VR) games continue to grow in popularity with the advancement of commercial VR capabilities such as the inclusion of full body tracking. This means game developers can design for novel interactions involving a player's full body rather than solely relying on controller input. However, existing research on evaluating player interaction in VR games primarily focuses on game content and inputs from game controllers or player hands. Current approaches for evaluating player full body interactions are limited to simple qualitative observation which makes evaluation difficult and time-consuming. We present a Full Room Virtual Reality Investigation Tool (FRVRIT) which combines data recording and visualization to provide a quantitative solution for evaluating player movement and interaction in VR games. The tool facilitates objective data observation through multiple visualization methods that can be manipulated, allowing developers to better observe and record player movements in the VR space to improve and iterate on the desired interactions in their games.

CHI'19 Extended Abstracts, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5971-9/19/05.
https://doi.org/10.1145/3290607.3312825

ACM Reference Format:
Daniel MacCormick, Alain Sangalang, Jackson Rushing, Ravnik Singh Jagpal, Pejman Mirza-Babaei & Loutfouz Zaman. 2019. FRVRIT - A Tool for Full Body Virtual Reality Game Evaluation. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI'19 Extended Abstracts), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3290607.3312825

1 INTRODUCTION

Commercial virtual reality (VR) has risen in popularity [10] since it was recently revitalized by the *Oculus Rift* [4]. Since then, numerous VR systems have been created, each with varying unique features. One such device is the *HTC Vive* [6]. The *Vive's* key selling feature is that of the "room-scale" experience, which highlights the use of full physical movement around a 3D space tracked by the device. Taking advantage of this, developers have created games such as *Beat Saber* [9] and *Job Simulator* [3] that encourage players to move in specific ways based on interactions within the game world. These types of games enable players to interact with their full body, such as when reaching for objects or walking to a goal. These forms of movement for interaction were simply not practical before the current state of commercial VR hardware. As a result, existing evaluation approaches will need to be adapted to consider the player's body. With this in mind, we created FRVRIT to aid VR developers with analyzing and evaluating player movements and full-body interactions within their games.

2 BACKGROUND

Analytics and data visualization have been used in interactive application development for a number of years. Petros Daras et al. [12] utilized a *Microsoft Kinect* [2] to evaluate a dancer's performance. They applied the skeleton tracking capabilities of the *Kinect* along with data acquisition software developed in *MatLab* [11] to record and automatically evaluate how a dancer performed. They then proceeded to integrate the visualization into the *Unity* [1] game engine, mapping the movements onto 3D avatars.

Vixen [13] is a tool for analytics visualization. It uses a tracking system within *Unity* and represents the data as an overlay within the game scene. In addition, it represents custom events as 2D sprites above the timeline. FRVRIT uses a similar data collection system that is also integrated into *Unity*. In addition, we make use of similar visualization methods, as Vixen also used line renders as timelines of player movement.

Norman et al. [15] developed a prototype client that explores and retrieves from motion capture archives based on an inputted sketch. Part of their process involved data harvesting using a *Vicon* optical capture system during motion-capture sessions. These motion capture traces were then visualized in a 3D space similar to FRVRIT's visualization workspace.

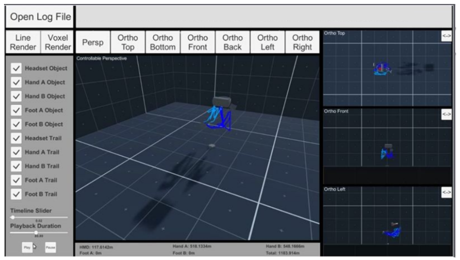

Figure 1. The full UI for FRVRIT. Currently showing the timeline view, with the tracked objects rendered within the replay. The multiple views and the buttons to control which camera perspective is used for each view give users flexibility.

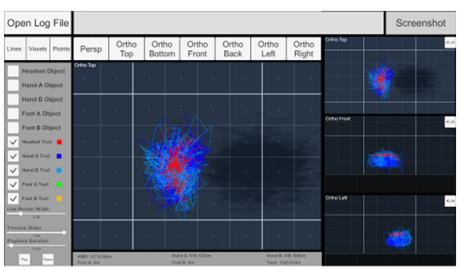

Figure 2. Timeline View in FRVRIT. The path between the data points is represented by a coloured line, with each colour representing a separate object.

Kriglstein et al. [14] performed a user study with several different gameplay visualizations. They compared heatmaps, circular cluster representations, and enclosure representations. They found that heatmaps were effective for identifying hotspots for a single variable, which is how it is used in FRVRIT. They also determined that colour is useful when representing several variables at once. The use of colour was considered when developing FRVRIT as each of the individually tracked components are represented by a unique colour.

We identified two common movement styles within commercial VR games, low- and high- intensity. Low-intensity games such as *Job Simulator* and *PaintLab* [5] rely on the player moving around the space, interacting with objects placed all around the scene. High-intensity movement style games such as *Beat Saber* have less movement around the scene overall but involve more strenuous actions. We designed FRVRIT with the goal of working well with evaluating both movement styles, across any VR game. Building on the existing research noted above, FRVRIT focuses on creating a visualization to aid game developers with quantitatively evaluating player movement and interaction to better match the intended design of their VR games.

3 FULL ROOM VIRTUAL REALITY INVESTIGATION TOOL - FRVRIT

FRVRIT was developed within *Unity*. From our testing, it can accurately portray player movements in any third-party VR game. FRVRIT is a visual evaluation tool, that is paired with a data recording program. These two components are integrated so that the data outputted from the recording is formatted for the input required by the visualization. FRVRIT currently only supports the *Vive*. It collects full body movement data from three main sources: the headset (player's head); two handheld controllers (player's hands); and up to two additional trackers which can be placed on the player's feet to track foot positions while walking.

3.1 Recording the Data

The recording program stores data gathered from each of the *Vive's* tracked objects with real-time positional data as three-dimensional vectors and rotational data as quaternions. The *Vive's* positional data is in meters, with the middle of the play-space's floor designated the origin. The data is taken from the gyroscopes and accelerometers contained within the hardware. This program was based on an online tutorial [7] on converting a sample *Vive* application into a background recording application. We made some adjustments to the original program, specifically increasing the amount of tracked objects, adding a controllable window, and reformatting the output. It was important to track data from all five objects as the head provides information on where a player is looking, the hands are the main interaction points for VR games, and the feet indicate where a player is walking relative to the full room scale environment.

FRVRIT's recording software works as a C++ executable that can be run in the background alongside a third-party VR game, collecting all of the tracked information without affecting gameplay. Once the data recording is stopped, the resulting data is output into a text file in CSV (comma-separated-value) format to be loaded into the visualization tool. The file browser used in the visualization tool is a modified version of *Simple File Browser* [8], a free asset from the *Unity Asset Store*. Once the data is loaded in, the visualization process can begin.

3.2 Visualizations

While the data collection occurs in VR, the visualization is presented to the developer in a separate non-VR environment afterwards. The data is represented in 3D, with individual points being placed at the real-world positions of the tracked objects in a 1:1 manner, since *Unity* also uses meters as its units. The main viewport in the center of the screen has seven possible camera angles, with the primary angle being a controllable perspective view (see Figure 1). The user is able to switch between these cameras at any time, providing various options to view the data representation. There are also three additional views on the side of the screen that are locked into orthographic projections, providing several angles simultaneously. In addition, there is a raw representation of the displacement values for each tracked object, below the main viewport. There are three primary visualization methods: the timeline view (see Figure 2), the voxel heatmap view (see Figure 3), and the point cloud view (see Figure 4).

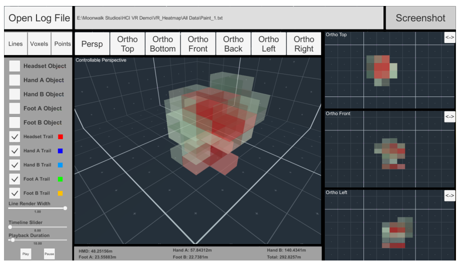

Figure 3. Voxel Heatmap View in FRVRIT. The voxel sizes can be controlled, allowing for both broad and precise analysis. The more data represented within the voxel, the more red it becomes.

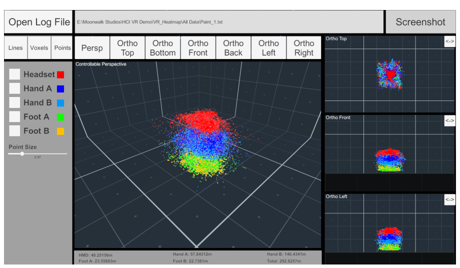

Figure 4. Point Cloud View in FRVRIT. The point cloud represents all of the data points at the same time, without live playback.

3.2.1 Timeline View.

The first view is the timeline. This method replicates the player's movements in the order they occurred and supports live playback, allowing it to be used as a generic replay system for any VR game. The objects themselves can also be rendered into the scene if desired. As the timeline progresses, the paths are drawn out alongside the objects. The developer can selectively view whichever part of the data they find to be the most relevant, or display all of it at the same time. This provides them with a high amount of control when viewing the data.

3.2.2 Voxel Heatmap View.

The second visualization method is the voxel heatmap. This visualization method is best for identifying "hot-spots" within the data, or areas where the player spent a significant amount of time. This could potentially be used for determining if players are moving as intended, or if they are too static when playing.

3.2.3 Point Cloud View.

The final view is the point cloud. Similar to the timeline view, it represents all of the objects at once, utilizing the same colour-based system. The intent of this view is to view all of the data at a glance. The point cloud sizes can be controlled and toggled individually.

4 EVALUATION

We performed a simple evaluation on the prototype after its completion. We chose to use a cognitive walkthrough method so we could test the functionality within our development team. We determined that we should evaluate the tool's functionality with a low-intensity game, a high-intensity game, and a full-body tracked game. *PaintLab* was selected as the low-intensity game, since it is almost entirely player-driven, with very little pressure to engage in movement. *Beat Saber* was used as the high-intensity game due its to high-pressure nature and popularity. Lastly, we created a simple first-party game to test the tool with a game that involved feet tracking in its design (see Figure 5). We anticipated that the key difference between low- and high- intensity movement would be the density of the tracked data points. We expected the high-intensity visualization to show concentrated areas of data points whereas the low-intensity data points to be spread evenly throughout the 3D space. We played each of the three games as a normal player would. However, we wore the two additional trackers on our feet for all three games to confirm the data would be recorded correctly across all tests. Once we had all of the data sets recorded, we visualized and evaluated them using FRVRIT.

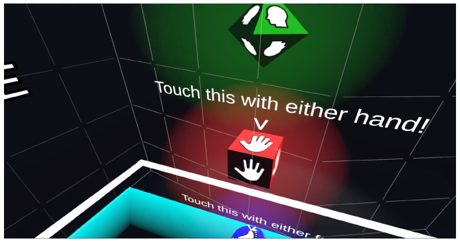

Figure 5. First-party full-body tracked game. In the game, targets appear around the user, throughout the entire play space. These targets have to be hit by a specific body part - the head, the hands, or the feet. The game is set up so that the players' feet were tracked by the additional trackers and rendered in game. This facilitated kicking of the objects, which meant that we could engage the player's entire body within the game.

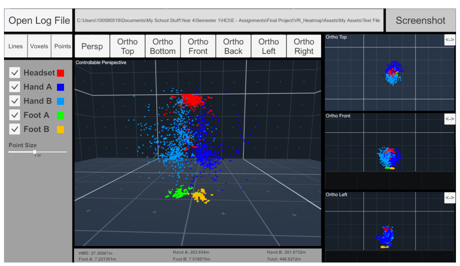

Figure 6. The data from the playing of *Beat Saber*

4.1 Results and Discussion

Two major design issues were noted as a result of testing. We found that the inability to view multiple data sets at the same time is problematic, as it results in a need for external analysis. The second major design issue is that of heatmap performance. The voxel heatmap is the most intensive visualization method of the three. Our initial test data sets were small and worked well, however the larger "real" data sets have highlighted a lack of optimization within this view.

As expected, there was a noticeable difference in the data from the low-intensity games and the high-intensity game. The low-intensity games had a much more spread out movement pattern, compared to the very focused pattern of the high-intensity game (see Figures 6-8). In addition, since the data was recorded externally, the foot visualization worked as well in the third-party games as it did in the first-party game. With these findings matching our expectations, FRVRIT has the potential to be an effective tool for VR game developers. In a similar manner, developers can compare their expected player movement and interactions in their games, to the results viewed within FRVRIT.

5 CONCLUSION

We developed FRVRIT as a way to aid VR game developers with evaluating player movement and interaction quantitatively, specifically in comparison to their own expectations. This is achieved through visualization of recorded data in multiple representation formats. Our own internal testing has indicated that FRVRIT is effective at accomplishing this goal, but can be improved upon going forward.

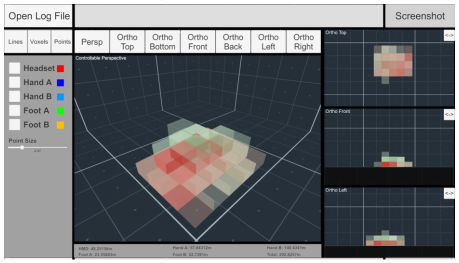

Figure 7. The data from the playing of *PaintLab*

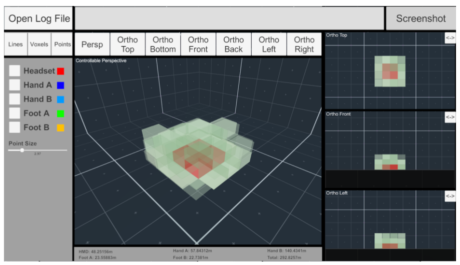

Figure 8. The data from the playing of our first-party full body game

6 FUTURE WORK

Most of FRVRIT's future work is focused on improving the workflow of the developer using the tool, as well as enabling more detailed evaluation of player movement and interaction within their games. We plan to improve performance in general, but most specifically with the voxel heatmap. Next, we will add support for tracking what the player sees throughout the play session as well. Given the dynamic camera capabilities provided by VR, this is especially critical. Afterwards, we plan to implement the ability to overlay multiple data sets at the same time, allowing for easier comparison between individual players. We will implement additional controls and parameters that the developer using the tool can adjust. For example, the ability to change the colour of the render would be beneficial, especially for accessibility purposes. Finally, we will run a user study with professional game developers to evaluate how effective FRVRIT is as an evaluation tool in their game production.

REFERENCES

[1] 2005. Learning C# and coding in Unity for beginners. https://unity3d.com/learning-c-sharp-in-unity-for-beginners
[2] 2014. Kinect - Windows app development. https://developer.microsoft.com/en-us/windows/kinect
[3] 2016. Job Simulator: the 2050 Archives | Owlchemy Labs. https://jobsimulatorgame.com/
[4] 2016. Oculus Rift: VR Headset for VR Ready PCs | Oculus. https://www.oculus.com/rift/#oui-csl-rift-games=robo-recall
[5] 2016. PaintLab on Steam. https://store.steampowered.com/app/455160/PaintLab/
[6] 2016. VIVEâĎć Canada | Discover Virtual Reality Beyond Imagination. https://www.vive.com/ca/
[7] 2017. How to Get Raw (Positional) Data from HTC Vive? - CodeProject. https://www.codeproject.com/Articles/1171122/How-to-Get-Raw-Positional-Data-from-HTC-Vive
[8] 2017. Simple File Browser - Asset Store. https://assetstore.unity.com/packages/tools/input-management/simple-file-browser-98451
[9] 2018. Beat Saber. http://beatsaber.com/
[10] 2018. Global augmented/virtual reality market size 2016-2022 | Statistic. https://www.statista.com/statistics/591181/global-augmented-virtual-reality-market-size/
[11] 2018. MATLAB - MathWorks. https://www.mathworks.com/products/matlab.html
[12] Dimitrios S. Alexiadis, Philip Kelly, Petros Daras, Noel E. O'Connor, Tamy Boubekeur, and Maher Ben Moussa. [n. d.]. Evaluating a dancer's performance using kinect-based skeleton tracking. In *Proceedings of the 19th ACM international conference on Multimedia - MM '11* (2011). ACM Press, 659. https://doi.org/10.1145/2072298.2072412
[13] Brandon Drenikow and Pejman Mirza-Babaei. [n. d.]. Vixen: interactive visualization of gameplay experiences. In *Proceedings of the International Conference on the Foundations of Digital Games - FDG '17* (2017). ACM Press, 1–10. https://doi.org/10.1145/3102071.3102089
[14] Simone Kriglstein, GÃijnter Wallner, and Margit Pohl. 2014. A user study of different gameplay visualizations. ACM, 361–370. https://doi.org/10.1145/2556288.2557317
[15] Norman Sally Jane, Lawson Sian E.M., Olivier Patrick, Watson Paul, Chan Anita M.-A., Dade-Robertson Martyn, Dunphy Paul, Green Dave, Hiden Hugo, Hook Jonathan, and Jackson Daniel G. 2009. AMUC: Associated Motion capture User Categories. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367, 1898 (July 2009), 2771–2780. https://doi.org/10.1098/rsta.2009.0030