XREcho: A Unity plug-in to record and visualize user behavior during XR sessions

Sophie Villenave
s.villenave@vassileo.fr
Vassiléo
Montpellier, France

Jonathan Cabezas
jonathan.cabezas@etu.univ-lyon1.fr
Univ Lyon, INSA Lyon, CNRS, UCBL, LIRIS, UMR5205
F-69621 Villeurbanne, France

Patrick Baert
patrick.baert@enise.fr
Univ Lyon, Centrale Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205
F-69130 Ecully, France

Florent Dupont
florent.dupont@univ-lyon1.fr
Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205
F-69622 Villeurbanne, France

Guillaume Lavoué
guillaume.lavoue@enise.fr
Univ Lyon, Centrale Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205
F-69130 Ecully, France

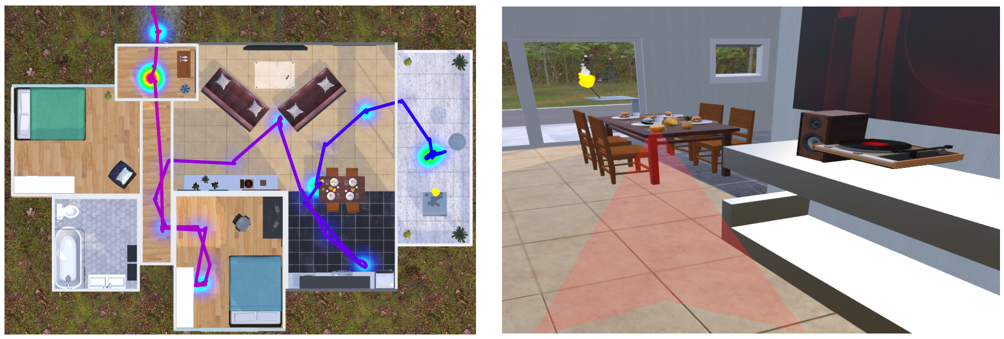


Figure 1: Overview of XREcho : a Unity plug-in for recording, replaying and visualizing user behavior and interactions in xr sessions. Left : Trajectory curve and position heatmap of a participant to an XR experimentation visualized with XREcho. Right : First-person view of a participant to an XR experimentation with the location of the eye gaze replayed by XREcho.

ABSTRACT

With the ever-growing use of extended reality (XR) technologies, researchers seek to understand the factors leading to a higher quality of experience. Measurements of the quality of experience (e.g., presence, immersion, flow) are usually assessed through questionnaires completed after the experience. To cope with shortcomings and limitations of this kind of assessment, some recent studies tend to use physiological measures and interaction traces generated in the virtual world in replacement or in addition to questionnaires. Those physiological and behavioral measurements are still complex to implement, and existing studies difficult to replicate, because of a lack of easy and efficient tools to collect and visualize such data produced during XR experiments. In this paper, we present XREcho, a Unity package that allows for the recording, replaying and visualization of user behavior and interactions during XR sessions; recorded data allows to replay the whole experience, as it includes movements of the XR device, controllers and interacted objects, as well as eye tracking data (e.g., gaze position, pupils diameter). The capabilities of this tool are illustrated in a user study, where 12 participants' data have been collected and visualized with XREcho. Source code for XREcho is publicly available on GitHub.[1]

CCS CONCEPTS

• **Human-centered computing** → **Visualization toolkits**; • **Computing methodologies** → **Virtual reality**.

KEYWORDS

Virtual Reality, Visualization Toolkit, Unity, Behaviour Analysis

ACM Reference Format:
Sophie Villenave, Jonathan Cabezas, Patrick Baert, Florent Dupont, and Guillaume Lavoué. 2022. XREcho: A Unity plug-in to record and visualize user behavior during XR sessions. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3524273.3532909

[1]GitHub link for XREcho : https://github.com/Plateforme-VR-ENISE/XREcho

1 INTRODUCTION

Virtual, augmented and mixed reality (further mentioned as "extended reality" or "XR") technologies are nowadays used in a plethora of domains, from entertainment to pedagogy passing by marketing. While the public interest is high, there are a lot of open questions for researchers in the field to answer, whether it be on the enablers of immersion or the potential drawbacks of these technologies.

As of today there are multiple studies aiming at understanding human behavior within virtual worlds. Those studies try to assess certain cognitive responses, such as presence, flow, stress or enjoyment. Assessment of participants cognitive response in XR is mostly made via questionnaires. However questionnaires may not be reliable enough and may be sometimes subject to bias. This is particularly true in the field of XR, where participants may experiment very unequal quality of experience, because of sometimes marked differences in terms of XR practice, vision, sensitivity to cyber-sickness or even video-game abilities. Moreover, those questionnaires can take different forms and protocols to answer them may vary. As an example, Alexandrovsky et al. [2] investigated the reliability of questionnaires by comparing the results between administration inside and outside the XR and during or after the experiment.

To get more reliable results, some recent studies tend to use physiological responses and interaction traces generated in the virtual world in addition to questionnaires. While those studies are promising and allow for a better understanding of human behavior within the XR world, they rarely share the tools used to collect data. To the best of our knowledge, there is no open-source XR data collection framework that could help the community advance research in this field.

In this context, our objective is to allow broad access to objective and standardised behavioral data. We thus propose a Unity plug-in named XREcho. Our plug-in allows to (1) record XR sessions (user movements and fields of view, object interactions and full eye tracking data), (2) replay them in real time and (3) provide meaningful visualizations. Recorded data is saved in CSV files so it can be imported in most data analysis softwares.

In the current state, our plug-in is compliant with OpenXR applications using the Unity XR Interaction Toolkit. Because those tools tend to evolve quickly, we tried to have the most generic implementation. Some changes in the scripts might be needed to provide full support for OpenVR or Oculus. However we recommend to the scientific community to adopt OpenXR for their future developments, as it is becoming a well accepted industry standard and saves the hassle of having to switch XR backends depending on the device.

In the rest of this paper, we review the work related to our research axis, then present an overview of the functionalities of XREcho as well as its architecture and implementation and finally illustrate the use of our tool in a case study.

2 RELATED WORK

An aspect of XR research focuses on the evaluation of cognitive responses of XR user depending on varying parameters of the environment. While most of these studies use questionnaires to answer research questions, some recent studies incorporate the usage of physiological and movement data. However, as there are still few XR studies that rely on such objective data to answer their research questions, there is a lack of models and methods to deeply understand behavioral data. Robitaille et al. [5] conducted a study and created a model that was able to classify different types of user behaviors using only position and direction of viewing.

Determining the presence (or not) of differences between the real world and the XR one is an important step towards being able to use XR as a tool for experimentation. Recent studies going in this direction ([1], [3]) tend to show there are few differences between the two worlds in matter of human locomotion and eye-gaze in short scenarios and ideal XR conditions. However further research is needed to confirm those results as they partly contradict older similar studies.

If we focus specifically on visual attention data (from eye tracking), several works have been done to analyse and visualize them in the context of 360° video [7]. Several efficient attention predictors have also been developped [4]. However, Rossi et al. [6] showed these models proposed for 360° video do not apply to XR experiments allowing 6 Degrees of Freedom (DoF) (compared to 3-DoF with 360° videos) and proposed new metrics to analyse human behavior in XR contexts. Zerman et al. [8] studied position and viewing angle in relation to volumetric videos of humans in the context of AR shown on a mobile phone. The studies mentioned above are very recent and remain preliminary, more studies and datasets are needed to fully understand visual attention in 6-DoF XR contexts.

To the best of our knowledge, the XR community suffers from a lack of efficient and easy-to-use tools to collect, analyze and visualize behavioral data (e.g., user movement, gaze position) and interaction traces in XR environments. We believe that it makes it harder to reproduce studies and create new ones in different contexts. In light of these observations, we developed a Unity plug-in allowing the recording, replaying and visualization of XR sessions. We named our plug-in *XREcho*, to express the fact that we can now see the echo of our XR experiments. Since we wanted to create a powerful and complete tool accessible to the majority of researchers, XREcho will be available on GitHub.

3 FUNCTIONALITIES

This plugin we propose offers three main features: (1) record XR sessions (XR device, controller and interacted objects movements; eye-tracking data) within Unity; (2) replay recorded XR sessions (either in first-person or in top view); (3) visualize user position (trajectory curves, individual or multiple user position heatmaps).

Those functionalities are detailed below. For each of them, we present how to use XREcho during development via the Unity Editor and its use during the execution of the application via a GUI displayed on the top-left of the application, only visible on the computer side. For a rapid overview of the in-application use of XREcho, figure 2 presents main functionalities as a flowchart.

3.1 Recording

Unity Editor

Figure 3 illustrates the interface of the Recording Manager within the inspector of the Unity Editor. From this interface we can choose

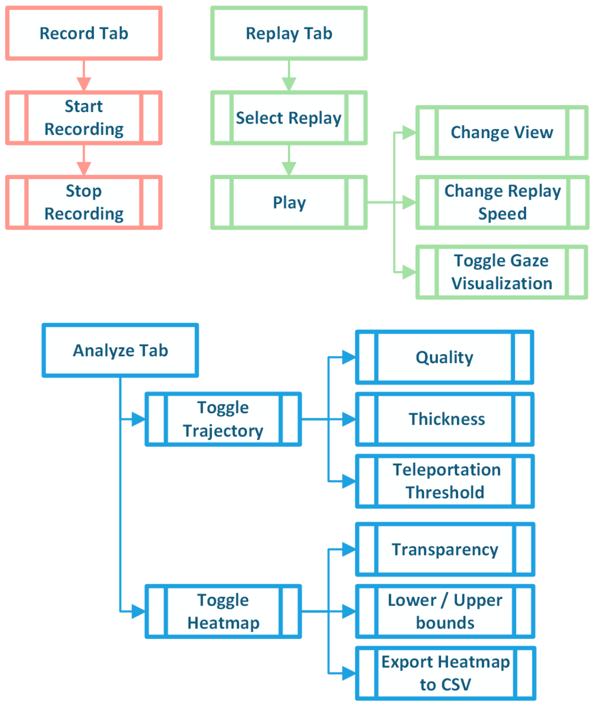
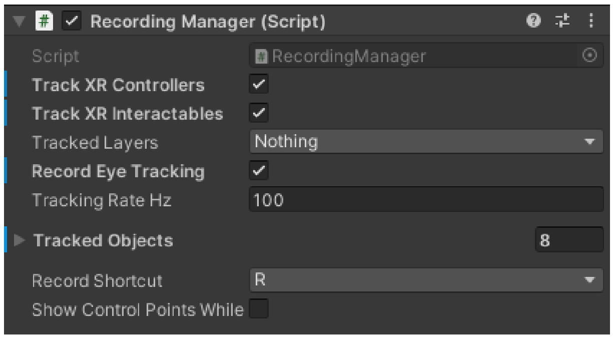


Figure 2: XREcho use in-app

to track the position and rotation of the XR Controllers and / or every GameObject that can be interacted with. When *GameObjects* are selected to be tracked, they are added to the *Tracked Objects* list whose content is displayed in this interface. Any type of object can be manually added to the *Tracked Objects* list. Unity proposes to arrange *GameObjects* in layers to facilitate bulk application of properties or manage possible interactions between *GameObjects*. Layers can be selected to be added to the list of *TrackedObjects*. Checking *Record Eye-Tracking* on the interface adds a prefab that includes the necessary scripts for the recording of the eye tracking data (validity of data, openness of the eyes, pupils diameter, pupils position, gaze origin, distance of the eyes from the central point) from the HTC Vive Pro Eye. Eye-tracking related scripts should be modified to ensure compatibility with other eye-tracking able devices such as the Varjo XR3 or the Hololens 2. Tracking rate of the *Tracked Objects* can be modified, its default value being 100Hz. Using a higher tracking rate may induce performance issues. A shortcut can be set to use of the recorder within the application without a mouse.

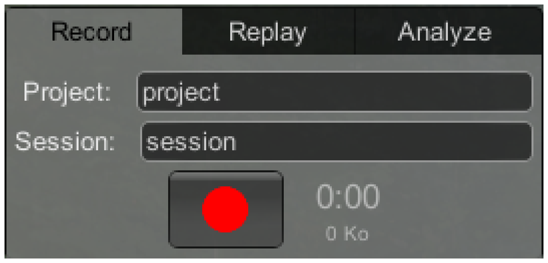
In-Application GUI

Figure 4 presents the in-application GUI shown on the computer side for the record tab. From here the current XR project and session names can be changed right before recording. Those parameters are automatically loaded from the last recording. Clicking on the record button (or pressing the selected shortcut) starts the recording of the selected objects; ongoing recording information (elapsed time and total data size) are displayed.


Recording Manager (Script)

Script	RecordingManager
Track XR Controllers	✓
Track XR Interactables	✓
Tracked Layers	Nothing
Record Eye Tracking	✓
Tracking Rate Hz	100
Tracked Objects	8
Record Shortcut	R
Show Control Points While	

Figure 3: *Recording Manager* within Unity Editor



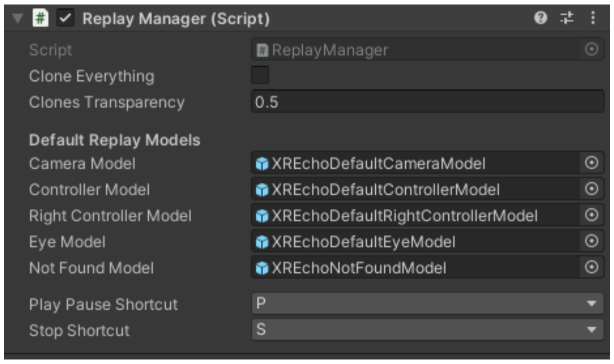
Project: project
Session: session
0:00
0 Ko

Figure 4: Recording tab in the application

3.2 Replaying

Unity Editor

Figure 5 illustrates the interface of the Replay Manager within the inspector of the Unity Editor. From this interface we can choose to use clones of the *TrackedObjects* when replaying so that the original *GameObjects* remain untouched. For each category of *TrackedObjects* (Headset, Controllers, Interactables, Eyes) and for *TrackedObjects* not found in the current scene, we can assign default models to be used when replaying which substitute the original *TrackedObjects*. Shortcuts can be set to use the XR player within the application without a mouse.

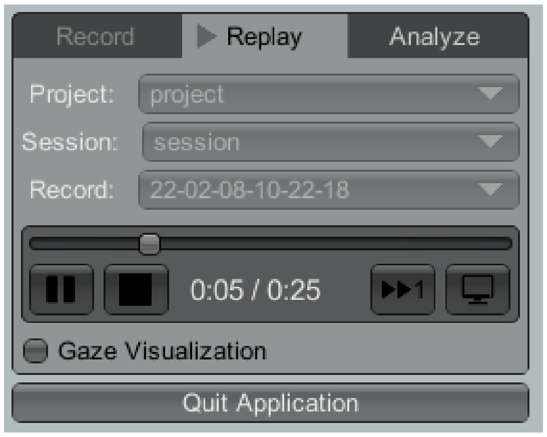

Replay Manager (Script)

Script	ReplayManager
Clone Everything	
Clones Transparency	0.5
Default Replay Models	
Camera Model	XREchoDefaultCameraModel
Controller Model	XREchoDefaultControllerModel
Right Controller Model	XREchoDefaultRightControllerModel
Eye Model	XREchoDefaultEyeModel
Not Found Model	XREchoNotFoundModel
Play Pause Shortcut	P
Stop Shortcut	S

Figure 5: *Replay Manager* within Unity Editor

In-Application GUI

Figure 6 presents the GUI for the replay tab as a record is being played. From there we can select a specific record, within a project and a session. Clicking on the play button (or pressing the specified shortcut) will start the replay. Selected *Tracked Objects* will evolve in the scene as recorded. The player allows to go through the entire record, at normal, x2 and x8 speed or manually via the timeline. Default view for the replay of the record is a top-view in which we can move and zoom in or out. Clicking on the screen icon switches the view to a first-person view of the user. If eye-tracking data were recorded, checking *Gaze Visualization* shows the eye gaze modeled by a ray originated from the user's head towards the point of interest.


Record | Replay | Analyze

Project: project
Session: session
Record: 22-02-08-10-22-18

0:05 / 0:25

Gaze Visualization

Quit Application

Figure 6: Replaying tab in the application during replay

3.3 Visualizations

In addition to the generation of data and the replay of recorded session, our plug-in integrates two modes of visualization for trajectory data. While replaying a session, the *Analyze* tab can be accessed to toggle the visualizations as shown in figure 7.

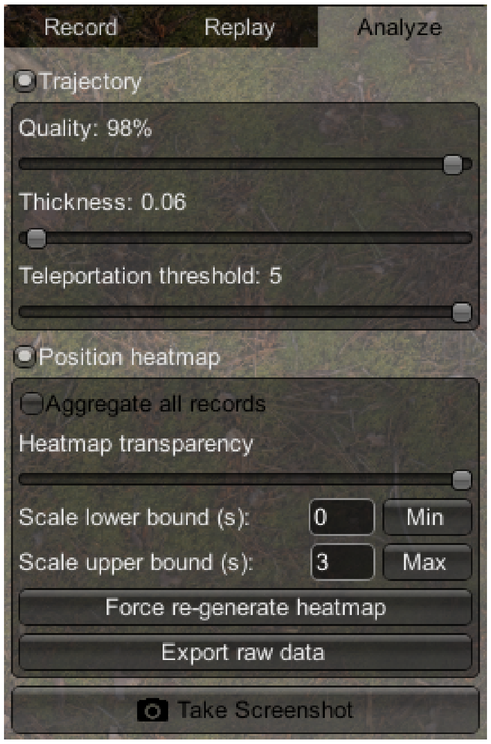
Trajectory

Visualizing trajectories directly from the application allows to see the paths taken by users in the scene without context loss. For example we can directly see users' teleportation frequency, if they made round trips or if they explored places they were not invited to. Checking *Trajectories* on the interface (figure 7) shows the user's trajectory as a break line (modeled by cylinders joined by spheres) between each recorded position. A color gradient is added to visualize the elapsed time directly on the trajectory (see Figure 1). During the execution of the application, the quality of the trajectory can be adjusted, using Unity's implementation of the Ramer Douglas Peucker algorithm[2]. The thickness of the line can be adjusted, as well as the teleportation threshold i.e. teleports of the user under the threshold (in meters) are modeled by a line and those above are not shown.

Spatial Heatmap

Displaying the users' spatial heatmap within the application allows to visualize the concentration of positioning in the scene without context loss. Heatmaps are useful to quickly identify movement patterns between users and can be used to feed behavioral models.

[2]Unity's documentation on the function used : https://docs.unity3d.com/ScriptReference/LineUtility.Simplify.html


Record | Replay | Analyze

Trajectory
Quality: 98%
Thickness: 0.06
Teleportation threshold: 5

Position heatmap
Aggregate all records
Heatmap transparency
Scale lower bound (s): 0 Min
Scale upper bound (s): 3 Max
Force re-generate heatmap
Export raw data

Take Screenshot

Figure 7: Visualization tab in the application

Checking *Position Heatmap* on the interface (figure 7) displays the spatial location of the user on the texture of a plane *GameObject*. This plane is divided by a grid of chosen resolution (e.g. 400 pixels per square meter) and each cell contains the duration spent on it by the user. The values from each cell are projected onto nearby cells via a gaussian of chosen standard deviation. As a default we chose the standard deviation to be one shoulder width. Then we apply a min-max normalization in order to obtain values between 0 and 1 making it easier to apply the color gradient. During the execution of the application, heatmap transparency is customizable as well as the values of the scale's lower and upper bounds. Raw and normalized heatmap data can be exported in a csv file for further analysis. Heatmaps from multiple users can be aggregated on a single heatmap to get an overview. Before the aggregation, heatmaps from every record of the current scene are computed and normalized. They are summed and averaged before the projection on the plane.

4 DESIGN AND IMPLEMENTATION

XREcho was made within Unity, a cross-platform Game Engine developed by Unity Technologies. Development within Unity is made in C# using the .NET platform and the engine can be used to create 2D, 3D or XR applications. Unity's ease of access and use makes it an obvious choice for researchers when looking to create XR experiences.

As presented above, our plug-in is able to record XR Sessions, replay them in real time and show meaningful visualizations for Unity applications. Its integration within Unity is made easy as it is delivered as a Unity Package and it can be integrated at any moment during development. Although there are multiple XR backends to choose from (OpenVR, Oculus, WMR, OpenXR) when developing an XR Application, we selected OpenXR[3] as our backend because it provides an application interface between most XR hardware and XR applications, hence making our plug-in as accessible as possible.

Our plug-in is ready-to-use if the application is using OpenXR as its plug-in provider within Unity and Unity's XR Interaction Toolkit[4] as its XR framework. Minor workarounds can be implemented in order to use a different backend or XR framework.

Regarding code architecture when developing a Unity application : C# is an object oriented language and Unity made the choice of a component based architecture. Unity developers rarely rely on inheritance to create complex behaviors like we would on a more traditional application. They create *GameObjects* which are the fundamental objects, then add components on those *GameObjects*, each component being a C# class inheriting from base class *MonoBehaviour*.

XREcho is made up of a multitude of components, each component being dissociated as much as possible in order to provide modularity. The choice of the singleton design pattern for the majority of components, including the main component, allows a certain flexibility and avoids possible conflicts between different instances. Here are the main components and classes of our plugin that can be set in the Unity editor :

XREcho: High-level Singleton containing general parameters of the plug-in such as the execution mode (automatic or manual), display of the GUI or selection of the monitoring camera.

XREchoConfig: High-level Singleton containing configuration parameters of the plug-in such as path names or data format.

Tracked Object: Contains information (GameObject, tracked parameters, tracking interval) on objects that are recorded.

Tracking Action: For each Tracked Object, we create one or more Tracking Action. A Tracked Action corresponds to a type of data that needs to be recorded, i.e. a position, rotation or camera parameters.

Recording Manager: Singleton that manages the recording of the tracked objects.

Recording Metadata: This class manages the storage of metadata on each record such as the start timestamp, the duration, the active scene, the number of tracked objects and so on.

Replay Manager : Singleton that manages the replaying of the tracked objects.

Trajectory Manager : Singleton that manages the computation and display of user trajectory.

Position Heatmap Manager : Singleton that manages the computation and display of user and aggregated position heatmaps.

GUI Manager: Singleton that manages the display of the graphical user interface (GUI).

5 CASE STUDY - STUDYING HUMAN BEHAVIOR AND PRESENCE IN VR

We used XREcho in the context of a study seeking to determine the influence of sound quality on user experience and behavior in virtual reality. For the purpose of our study we used a HTC Vive Pro 2 paired with Index controllers. Before the experiment, participants anonymously complete a demographic questionnaire about their age, gender, video-games experience and VR experience. On completion, participants are taught how to use the HMD and its controlers, then they freely navigate in a VR scene representing a rural 2-bedroom house surrounded by forest and mountains. Participants need to complete an item search guided by a compass quest which ends in an archery task against zombies. We designed three versions of this scene, each version having a different level of sound quality: non-diegetic ambiant music, spatialized ambiant music and sound effects, spatialized and geometry aware ambiant music and sound effects. Participants are placed in a delimited area of 2 meters by 2 meters, where they can move their arms freely. The restricted size of the area implies that participants need to teleport to move around the virtual world. Right after the experiment, participants are showed a top-view screenshot showing their trajectory computed by XREcho, giving them immediate feedback on their action. Finally they complete two questionnaires: one on spatial presence and one on cyberkinetosis in virtual reality. From start to finish, the experiment is approximately 15 minutes long. For this experiment, the size of the recording per participant revolves around 50MB. More generally, file size is dependent on the length of the recording, the number of recorded objects and the enabling of the eye-tracking recording.

XREcho may help us create associations between state of presence and/or cyber-sickness and activity inside the virtual world or explore the creation of behavioral models integrating the user profile. As an example of this possibility, figure 8 presents aggregated heatmaps for each condition of our experimentation. In addition to collecting useful data to answer our research questions, XREcho helps producing reproductible results since we can make sure participations are valid by quickly reviewing recordings. The review is made easier by possibility to change the point of view: top-view for an overview and personal view for a more detailed review.

6 CONCLUSION

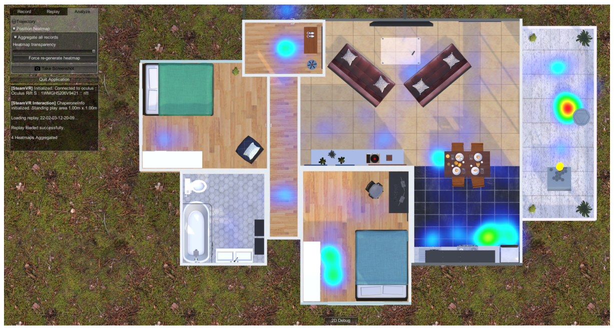
XREcho is a Unity plug-in allowing researchers to record user behavior and interactions within an XR application, such as headset and controllers movements, grabbed virtual objects and visual attention. Those interactions can be replayed in real-time and superposed with meaningful visualizations on user trajectory.

Our plug-in generates a massive quantity of complex data that needs new methods and algorithms to be analysed. XREcho already embeds several visualization tools that allow the production of preliminary analyses; we intend to add more visualizations, such as visual saliency maps.

Since we aim for XREcho to be used in the XR community, we plan to upgrade the plug-in continuously in order to stay compatible

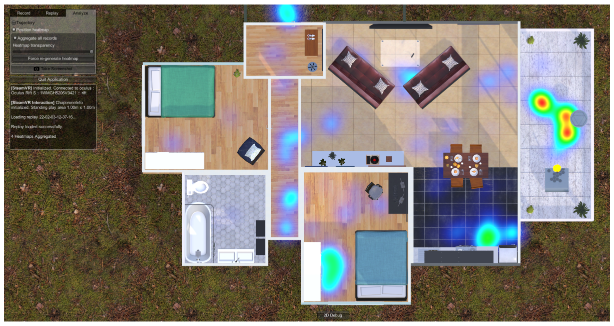
[3]From OpenXR website : https://www.khronos.org/openxr/ : "OpenXR is a royalty-free, open standard that provides high-performance access to Augmented Reality (AR) and Virtual Reality (VR) —collectively known as XR— platforms and devices." OpenXR is fully supported by Unity via package : https://docs.unity3d.com/Packages/com.unity.xr.openxr@1.3/manual/index.html

[4]From the package documentation : "The XR Interaction Toolkit package is a high-level, component-based, interaction system for creating VR and AR experiences. It provides a framework that makes 3D and UI interactions available from Unity input events. The core of this system is a set of base Interactor and Interactable components, and an Interaction Manager that ties these two types of components together." https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/index.html



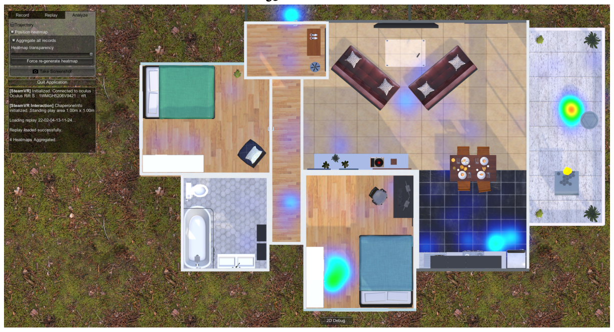
```
[SteamVR] Initialized. Connected to oculus : Oculus Rift S : 1WMGH5206V9421 :: rift

[SteamVR Interaction] ChaperoneInfo initialized. Standing play area 1.00m x 1.00m

Loading replay 22-02-03-12-20-09...

Replay loaded successfully.

4 Heatmaps Aggregated
```

(a) Aggregated heatmap of 4 participants from the *non-diegetic ambiant music* condition



```
[SteamVR] Initialized. Connected to oculus : Oculus Rift S : 1WMGH5206V9421 :: rift

[SteamVR Interaction] ChaperoneInfo initialized. Standing play area 1.00m x 1.00m

Loading replay 22-02-03-12-37-16...

Replay loaded successfully.

4 Heatmaps Aggregated
```

(b) Aggregated heatmap of 4 participants from the *spatialized ambiant music and sound effects* condition



```
[SteamVR] Initialized. Connected to oculus : Oculus Rift S : 1WMGH5206V9421 :: rift

[SteamVR Interaction] ChaperoneInfo initialized. Standing play area 1.00m x 1.00m

Loading replay 22-02-04-13-11-24...

Replay loaded successfully.

4 Heatmaps Aggregated
```

(c) Aggregated heatmap of 4 participants from the *spatialized and geometry aware ambiant music and sound effects* condition

Figure 8: Aggregated heatmaps for each condition of the case study

with the latest versions of Unity, OpenXR and the XR Interaction Toolkit. For the time being XREcho has been tested on PC VR applications using various devices. While our intent is for our plugin to be usable on any XR platforms, users should be aware that minor code modifications dependant on the hardware might be needed. Moreover, XREcho users are encouraged to create forks of the project as they might need to add specific functionalities such as monitoring physiological sensors.

ACKNOWLEDGMENTS

This work was partially supported by French National Research Agency as part of ANR-PISCo project (ANR-17-CE33-0005). The authors would like to thank Thibault Louis and Charles Javerliat for their contribution to the development of XREcho.

REFERENCES

[1] Philipp Agethen, Viswa Sekar, Felix Gaisbauer, Thies Pfeiffer, Michael Otto, and Enrico Rukzio. 2018. Behavior Analysis of Human Locomotion in the Real World and Virtual Reality for the Manufacturing Industry. *ACM Transactions on Applied Perception* 15 (July 2018), 1–19. https://doi.org/10.1145/3230648

[2] Dmitry Alexandrovsky, Susanne Putze, Michael Bonfert, Sebastian Höffner, Pitt Michelmann, Dirk Wenig, Rainer Malaka, and Jan David Smeddinck. 2020. *Examining Design Choices of Questionnaires in VR User Studies.* Association for Computing Machinery, New York, NY, USA, 1–21. https://doi.org/10.1145/3313831.3376260

[3] Florian Berton, Anne-Hélène Olivier, Julien Bruneau, Ludovic Hoyet, and Julien Pettré. 2019. Studying Gaze Behaviour During Collision Avoidance With a Virtual Walker: Influence of the Virtual Reality Setup. IEEE, 717. https://doi.org/10.1109/VR.2019.8798204

[4] Daniel Martin, Ana Serrano, Alexander W. Bergman, Gordon Wetzstein, and Belen Masia. 2021. ScanGAN360: A Generative Model of Realistic Scanpaths for 360° Images. (March 2021). https://arxiv.org/abs/2103.13922v1

[5] Patrice Robitaille and Michael J. McGuffin. 2019. Increased affect-arousal in VR can be detected from faster body motion with increased heart rate. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '19).* Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3306131.3317022

[6] Silvia Rossi, Irene Viola, Laura Toni, and Pablo Cesar. 2021. From 3-DoF to 6-DoF: New Metrics to Analyse Users Behaviour in Immersive Applications. *arXiv:2112.09402 [cs]* (Dec. 2021). http://arxiv.org/abs/2112.09402 arXiv: 2112.09402.

[7] Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. 2018. Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (2018), 1633–1642. https://doi.org/10.1109/TVCG.2018.2793599

[8] Emin Zerman, Radhika Kulkarni, and Aljosa Smolic. 2021. User Behaviour Analysis of Volumetric Video in Augmented Reality. In *2021 13th International Conference on Quality of Multimedia Experience (QoMEX).* IEEE, Montreal, QC, Canada, 129–132. https://doi.org/10.1109/QoMEX51781.2021.9465456