


# StudyFramework: Comfortably Setting up and Conducting Factorial-Design Studies Using the Unreal Engine

Jonathan Ehret<sup>1\*</sup> 

Andrea Bönsch<sup>1</sup> 

Janina Fels<sup>3</sup> 

Sabine J. Schlittmeier<sup>2</sup> 

Torsten W. Kuhlen<sup>1</sup> 

<sup>1</sup> Visual Computing Institute, RWTH Aachen University, Germany

<sup>2</sup> Work and Engineering Psychology, RWTH Aachen University, Germany

<sup>3</sup> Institute for Hearing Technology and Acoustics, RWTH Aachen University, Germany

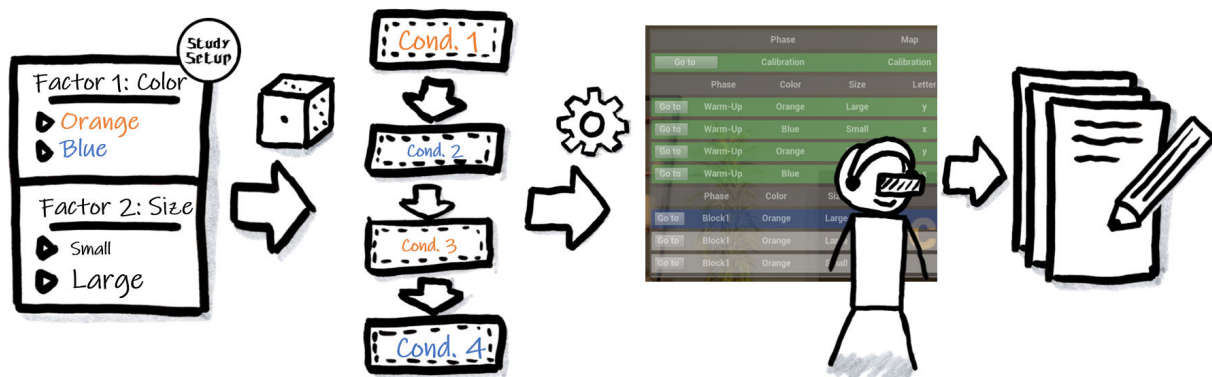


Figure 1: The pipeline of the *StudyFramework*: The factorial-design setup is randomized into an ordered list of conditions. This list can then be executed in VR, providing an helpful control interface to the experimenter. Finally, all gathered data is carefully logged.

## ABSTRACT

Setting up and conducting user studies is fundamental to virtual reality research. Yet, often these studies are developed from scratch, which is time-consuming and especially hard and error-prone for novice developers. In this paper, we introduce the *StudyFramework*, a framework specifically designed to streamline the setup and execution of factorial-design VR-based user studies within the Unreal Engine, significantly enhancing the overall process. We elucidate core concepts such as setup, randomization, the experimenter view, and logging. After utilizing our framework to set up and conduct their respective studies, 11 study developers provided valuable feedback through a structured questionnaire. This feedback, which was generally positive, highlighting its simplicity and usability, is discussed in detail.

**Index Terms:** General and reference—Empirical studies; Human-centered computing—Virtual reality; Human-centered computing—User studies; Software and its engineering—Frameworks;

## 1 INTRODUCTION

When developing new techniques or interfaces for virtual reality (VR), it is essential to assess their effectiveness through controlled user studies to confirm their optimal performance and meaningful enhancement over existing methods. Moreover, conducting user studies is essential to acquire a comprehensive understanding of human behavior. Here, VR is an emerging tool in, for example, psychological research where it can help to conduct controlled experiments in settings closer to real life than classical lab experiments. However, setting up those studies is time-consuming and holds a

lot of potential caveats with respect to experimental design and data management, such as counterbalancing randomized condition orders and thoroughly storing all relevant data. If setup errors go unnoticed until the study conductance or data evaluation, there is a substantial risk of data corruption, making parts or all of the data unusable.

Modern game engines like the *Unreal Engine* or *Unity* are readily available and are increasingly often used as they provide a lot of helpful tools to simplify setting up immersive virtual environments (IVEs) and also to implement complex interactions. However, their complex internal architectures can introduce numerous challenges when trying to implement a robust study that, for example, should include different virtual scenes, especially for engine novices. On top of that, the required code for setting up such studies is often very similar. Consequently, identical code has to be reproduced over and over again, potentially introducing unwanted side effects.

To facilitate the development and execution of factorial-design studies using the Unreal Engine, we present the *StudyFramework*. Factorial design thereby means that developers specify different factors to be varied across defined levels, and the system automatically generates and counterbalances the required conditions per participant (further details in Section 3).

Next, we explore related approaches. We then define our contributions by outlining our goals and detail the core components of our framework, employing an illustrative 2-factorial study, where the visibility of displayed letters is judged based on color and size (see Fig. 1 and supplemental video). Following this, we present an evaluation based on feedback from 11 independent study developers — computer science students or researchers in acoustics or psychology — who have used our framework. Rather than conducting an artificial study to compare our framework with existing ones or none at all, we deliberately chose to gather real-world experiences and insights from those actively using our framework. This approach allowed us to enhance our solution based on practical usage scenarios. The paper concludes with a discussion of these results.

\*e-mail: ehret@vr.rwth-aachen.de

## 2 RELATED WORK

Several frameworks and tools exist that should support researchers from diverse research areas to implement, set up, and conduct experiments in VR using various rendering engines. Most of them are built on top of Unity. One of these is the *Unity Experiment Framework (UXF)* [6]<sup>1</sup>, which allows to specify experimental orders a priori or progressively using a session-block-trial model and also supports remote experiments outside the lab. Another possibility is *BMLtux* [3]<sup>2</sup>, which allows to implement and conduct factorial-design experiments, aiding experimenters in visually keeping track of the progress of each session. Further frameworks are the *Virtual Reality Scientific Toolkit (VRSTK)* [32]<sup>3</sup> and the *Unified Suite for Experiments (USE)* [31]<sup>4</sup>, which both provide more integrated sensing capabilities, e.g., for brain activity. USE does that specifically by introducing a hardware device (*USE SyncBox*) to integrate measurements of electrophysiological recording devices with high-precision timing. VRSTK offers advanced features enabling experimenters to virtually immerse themselves within the IVE for enhanced interaction. Furthermore, it facilitates session replay and analysis. A recent framework that enables running distributed experiments with the potential for multiple remote participants is *Ubiq-Exp* [27]. It also supports conducting experiments both with and without an experimenter overseeing the process. Finally, *EVE* [19]<sup>5</sup> allows the integration of a commercial plugin (*MiddleVR*) so that experiments can be run in cave automatic virtual environment (CAVE) systems, as Unity natively only supports VR using head-mounted displays (HMDs). As this concludes our discussion on Unity frameworks, interested readers can find a detailed feature comparison of Unity frameworks for user study design and execution in [32].

Considering other frameworks besides Unity, *vexptoolbox* [23], provides more experimental control to the commercial VR platform *Vizard* (World-Viz, Santa Barbara, CA, USA) or a commercial solution for conducting VR experiments and exposition therapy: *CyberSession*<sup>6</sup>. Lastly, *R2VR* [30] should also be mentioned, which allows to build simple VR experiments directly in *R*, a well-known statistical software environment. For experiments not requiring VR, often python-based frameworks are used, like *PyEPL* [17] or, even more often, *PsychoPy* [22], which provides a versatile graphical user interface and is especially prominent for low-latency stimuli presentation and measurements. The latter was confirmed by Bridges et al. [4], performing a large-scale study comparing stimuli timing and latency for desktop-based experimental frameworks.

While the aforementioned frameworks support implementing and conducting an experiment to various degrees, there are also frameworks tailored to specific research domains, requiring minimal customization only. For example, *VREX* [29]<sup>7</sup> aids in setting up experiments in the field of experimental psychology and neuroscience in complex virtual indoor scenes. There are several toolkits to build navigational studies, e.g., *PandaEPL* [25], *Landmarks* [26], *NavWell* [9], or *DeFINE* [28], which require little to no coding. *VREVAL* [2], an Unreal-Engine-based tool, facilitates the efficient setup and execution of studies aimed at evaluating architectural models. Another Unreal tool is *DomeVR* [24], which was specifically designed to run experiments with rodents but also humans in a dome-shaped display device. For acoustical research, *Oticon Medical Virtual Reality (OMVR)* [21] was developed and provides a variety of valuable virtual scenes.

To the best of our knowledge, there exists no general, open-

<sup>1</sup><https://github.com/immersivecognition/unity-experiment-framework>

<sup>2</sup><https://github.com/BioMotionLab/TUX>

<sup>3</sup><https://github.com/ixperience-lab/VRSTK>

<sup>4</sup><https://github.com/att-circ-contnl/use>

<sup>5</sup><https://cog-ethz.github.io/EVE/>

<sup>6</sup><https://www.cybersession.info/>

<sup>7</sup><https://vrex.mozello.com/>

source framework to design and conduct factorial-design studies using the Unreal Engine, e.g., comparable to BMLtux. Additionally, many of the mentioned tools, due to their specificity limiting adaptability, lack the modularity required for an easy and seamless integration into a research prototype for evaluation. Finally, Aguilar et al. convincingly argue that the auditability and reproducibility of studies are paramount, emphasizing the current limitation of many tools and frameworks in effectively addressing these crucial aspects [1] (preprint)."

## 3 STUDYFRAMEWORK

Addressing the aforementioned gap, we introduce a new framework, called *StudyFramework*<sup>8</sup>, based on the Unreal Engine (currently developed for version 4.26 and 4.27 and 5.3). The core idea is to provide a light-weight solution that supports developing and conducting user studies with the following main aspects:

- easy setup of factorial-design studies
- out-of-the-box solutions for randomization/counterbalancing
- thoroughly tested and redundant data logging
- simple graphical interface facilitating monitoring and controlling user studies
- focus on VR but also a possibility for desktop studies
- support for multiple VR platforms, like HMDs and CAVE

In academia, a lot of studies are implemented and conducted by students evaluating their thesis projects, for example, having implemented a new interaction metaphor. Thereby they often lack both experience in the engine used and ample time to implement and thoroughly test their user study. Additionally, as mentioned above, researchers from other domains, like psychology, can benefit greatly from conducting VR user studies using game engines, while they potentially do not have a software development background. Consequently, one primary goal is to create a framework that is particularly user-friendly for individuals new to game engines, specifically within the context of Unreal Engine, as well as study design. While it is comparably simple to setup small interactive scenes in the Unreal Engine, mastering all intricacies to reliably conduct experiments (e.g., fading between scenes, having full control especially when something unexpected happens, reliably logging data, or counterbalancing orders) adds an extra burden onto novice developers. To this end, we provide the aforementioned functionality and made it accessible from both code (in this case *C++*) and also the visual scripting language provided by Unreal (called *blueprints*). The framework has been implemented as an Unreal plugin, ensuring an easy and seamless integration into any Unreal project. Another objective is to keep the framework lightweight, emphasizing its core purpose of facilitating the creation and execution of factorial-design studies. As a deliberate choice, we thus did not incorporate features unrelated to the study design or data management, such as immersive questionnaires. Users seeking this functionality can seamlessly integrate them through other plugins, like [16], or embed web questionnaires into the IVEs.

These mentioned aspect separate this approach from most of the existing tools mentioned above. Gröbel argues in a recent paper [18] that there are already sufficient experimental frameworks. However, we are convinced that our framework can contribute to a crucial gap. First, due to its modular design, we avoid the issue of overly specialized and overloaded frameworks, allowing novice as well as experienced developers to use it. Second, to the best of our knowledge, it is the first of its kind for the Unreal Engine, setting it apart from the aforementioned plethora of Unity-based solutions. This is significant, because Unreal, in contrast to Unity, enables VR application for CAVEs [10] through its native *nDisplay* plugin, thus

<sup>8</sup><https://git-ce.rwth-aachen.de/vr-vis/VR-Group/unreal-development/plugins/unreal-study-framework>

ParticipantID	Gender	Phase	Color	Size	Letter	Map	Visibility	Time
0	male	Warm-Up	Orange	Large	y	LivingRoom	good	4.19
0	male	Warm-Up	Blue	Small	x	LivingRoom	bad	8.84
0	male	Warm-Up	Orange	Small	y	LivingRoom	good	9.04
0	male	Warm-Up	Blue	Large	x	LivingRoom	good	30.49
1	male	Warm-Up	Orange	Large	y	LivingRoom	bad	10.26
1	male	Warm-Up	Blue	Small	x	LivingRoom	good	7.44

Table 1: Excerpt of an example phase log file, here for the example study setup used in Fig. 2 and Fig. 3. The csv file format is split into columns here for visibility. It contains the participant ID, independent variables (here: Gender), the phase name, factor levels (here: Color, Size, Letter, Map), dependent variables (here: Visibility) and the duration of the condition.

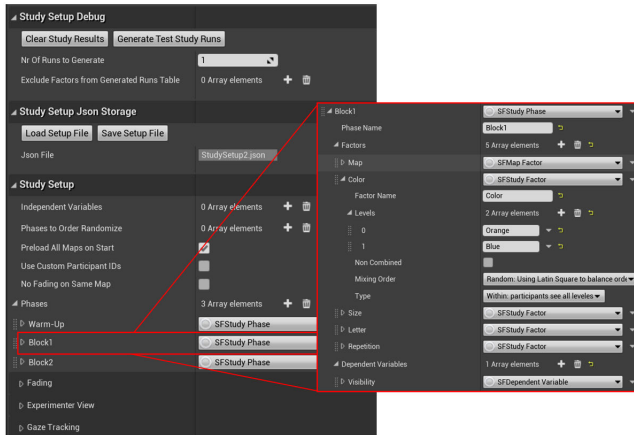


Figure 2: The *Details* section of a *StudySetupActor*. In the third section, multiple phases are specified. The specifics of one of the phases is expanded on the right, specifically also expanding one of the factors (*Color*) details, so its levels can be seen.

enabling a larger range of VR display settings for the studies. This is especially important for us, as we run a CAVE with a 49-node cluster [20]. Third, Unreal is open-source and its user-friendly visual-scripting blueprints provide an accessible programming interface, particularly suited for novice developers.

### 3.1 Components

The core idea of this framework is to develop **factorial-design** studies (similar to [3]). Factorial design is an experimental setup that consists of two or more independent variables also known as factors, with each factor having multiple levels. With a full factorial design, all possible combinations of the levels of a factor can be studied against all possible levels of other factors (in contrast to fractional factorial designs, systematically only showing a fraction of these conditions to each participant, but which are not directly covered here). Per condition (combination of specific levels per factor), data for one or multiple dependent variables (DVs) is gathered. Similar to blocks in the session-block-trial model (cp., [6]), we structure the experiment in different phases for which specific factors and DVs are defined individually (see Fig. 2). Sometimes a task is repeated multiple times in the same condition, for example, if multiple selection tasks are performed consecutively, and data beyond a mean performance score should be collected. Therefore, in addition to the previously specified DVs collecting one value per condition, we introduced special multiple-trial DVs. Lastly, also independent variables (IVs), going beyond the aforementioned factors, can be specified for which the data is collected at the very beginning of the experiment by means of input prompts shown on the experimenter screen (multiple-choice or text, for example, participant-specific data that should be reacted on). Similarly, participant IDs can be

set at the start, but for counterbalancing a sequential number is also always stored internally.

Since one key idea of the *StudyFramework* is to require as little programming background as possible for study developers, we tried to utilize graphical user interface (GUI) elements of the Unreal Editor. To create a new study, developers first drag-and-drop a *StudySetupActor* (actor is the Unreal term for any object within a map) into an empty map. Then the details panel (see Fig. 2) is used to configure all crucial design aspects of the experiment, which will be described below. Thereby, the configuration is directly stored and updated into a configuration file in human-readable *json* format. Developers can switch those files, for example, to assess different study configurations during development (see the second section in Fig. 2). When starting the study, the setup information is parsed and one singleton object (derived from the *GameInstance* class of the Unreal architecture) is created which holds all interfaces, e.g., for logging or controlling the study. A reference to this object is easily accessible both in C++ and blueprints.

However, before a session for a specific participant is started, the order of the individual conditions has to be **randomized and balanced** according to the setup. The framework supports both within- and between-subjects factors (cp., [11]). When multiple within-subject factors are specified for a phase, the conditions are created by combining each level of one factor with all the levels of the other factors. This process results in the Cartesian product of all factors (see Fig. 1). The default case would be to balance the order of these generated conditions using Balanced Latin Squares [12] so that the position at which each condition is presented and the condition after which each condition is presented is counter-balanced over participants to avoid potential position and order effects. To achieve this, we use the sequential number of the participants to pick an appropriate row from the Balanced Latin Square. Additionally, we shift the picked row of the Latin Square by the phase ID to avoid potential identical randomization in two repeated phases with the same factors. Moreover, our framework allows factors to always be presented *in order* (i.e., first all conditions with the first level and so on) or at least such that the same levels of one factor are presented *en bloc*. For instance, if the virtual scene is varied, all conditions within the same scene can be shown back-to-back, to minimize frequent scene transitions. Obviously, only one factor per phase can be specified as either of the two. This option can also be used to implement repetitions by defining a repetition factor and setting it to “*in order*” so that all first repetitions are finished before the second repetitions start. Furthermore, sometimes there should be balancing, e.g., of a task, which is not a factor to be examined. This is possible by defining *non-combined* factors, which do not contribute to the Cartesian product and are potentially randomly mapped to the aforementioned conditions. As a last resort, there is also a specific callback function that can be implemented by developers and gives the possibility to reorganize or filter generated conditions. Further documentation and examples can be found



Figure 3: The experimenter view overlay displayed over a demo study scene. Additionally to the status bar (top) and log section (bottom left), the condition list is currently shown (can be de/activate by the top button on the left). In the list completed conditions are marked in green and the current condition is highlighted in blue. Already gathered data is displayed.

in the project Wiki<sup>9</sup>. As a tool for checking the setup balancing for correctness, we added the possibility to generate the condition lists of an arbitrary number of participants and store them into a single text file for further inspection (see “Generate Test Study Runs” button at top of Fig. 2). In the shown example, *Size* and *Color* are 2-level factors, while *Letter* is a non-combined, randomly assigned factor with different levels in the *Warm-Up* and the *Block1* phase. Additionally, *Block1* has a third factor *Repetition* with two levels so that each combination of the first two factors is presented twice.

The framework also includes **logging** for positional data, data gathered for DVs, and potential events, all with timestamps. An event can be, for example, a participant interacting with a specific object, for which developers can log an arbitrary text, such as “*Object A picked up*”, using provided interfaces. Adding new actors of which position and orientation should be logged at each frame of the application, or less often if specified, is straight-forward by just adding a special logging component to these actors. Additionally, this component also allows to log custom data frame-wise, like the status of the actor or whatever is required in the specific use case. For each study phase a table in csv-format (comma separated values) is created, holding data collected for the DVs as well as the duration of each condition for all participants. These tables are created in long format, holding one line per condition and participant, as opposed to one line per participant, so that they can be easily loaded into statistics tools like R (see Tab. 1). The aforementioned multiple-trial DVs, do not fit into this format and therefore create one csv-file per variable with a line per recorded data point, so potentially multiple lines per condition and participant. Furthermore, all data is also logged redundantly per participant and session into a separate text file. In general special care was taken that all data is stored safely to avoid potential data losses.

For an immersive study, a virtual scene is mandatory and is sometimes also varied as part of the study. It is therefore also formalized as a factor in our framework. When switching scenes, we recommend to use **fading**, i.e. transitioning to a predefined color and after a while back to the new scene, to not confuse participants by immediately changing their entire surroundings and potential lags due to loading. To this end a configurable fading is implemented that works in VR and desktop mode and also defines callbacks that allow developers to react to a new level being loaded or having faded in, e.g., by starting a task only once the new scene is faded in.

To support experimenters, we added an **experimenter view**. It

<sup>9</sup><https://git-ce.rwth-aachen.de/vr-vis/VR-Group/unreal-development/plugins/unreal-study-framework/-/wikis>

- Q1 Experienced in Unreal
- Q2 Experienced in C++ development
- Q3 Experienced in factorial study design
- Q4 Easy usage of study setup
- Q5 Randomization options were clear
- Q6 Wiki was helpful
- Q7 C++/Blueprint interfaces sufficient
- Q8 Needed to look in source code freq.
- Q9 Needed a lot of help
- Q10 The experimenter view helped
- Q11 Felt in control conducting study
- Q12 Felt confident with recovery options
- Q13 Used “Show Conditions” regularly

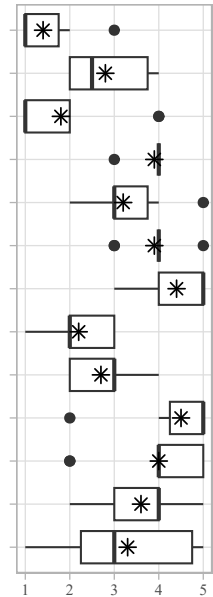


Figure 4: The answers by  $n = 10$  developers given to the individual statements as box plots on a scale from 1 (*Strongly Disagree*) to 5 (*Strongly Agree*). The stars indicates the mean while the boxes show the quartiles with whiskers extending to the full range and outliers being displayed as dots. Statements are given here as shortened versions, full-length statements can be found in Appendix A.

contains a status bar (see Fig. 3), that always shows what status the application is in and which condition is currently presented, and a log section showing the latest log messages. Developers can decide for logged messages specifically whether they should also be shown in the log section, to give the experimenter all relevant information and simultaneously not overload this log so that relevant information might be missed. On clicking the “Show Conditions” button in this experimenter view a scrollable condition list can be displayed (see Fig. 3). There, on top of seeing which condition is currently active and which are already finished, recorded data of DVs is shown. Furthermore, to simplify experiment development and debugging, specific conditions can directly be started there without jumping through previous conditions. This functionality can also be used during study execution to restart a specific condition, e.g., if the participant was distracted by something else happening and missed the start of a condition or wants to repeat a familiarization phase. This restarting is obviously also noted in the participant’s log file, but should be used very carefully during execution, potentially having confounding effects. The experimenter view can be shown as an overlay on what the participant sees in the HMD or in a separate window potentially on a second screen in desktop mode. This study control functionality is expanded by the possibility of recovering failed study sessions. If on starting the study an unfinished previous study run is detected, the experimenter can choose whether to continue the study from the last unfinished condition or start with a new participant from the beginning. This is helpful for quick and clean recovery if the software crashes unexpectedly during study execution.

### 3.2 Evaluation

The *StudyFramework* was already successfully used in 14 experiments (e.g., [7, 8, 13–15]) and successively updated and improved in that process. All developers of these studies (if not the first author of this paper) were asked to fill out a short subjective evaluation ques-



tionnaire after conducting their respective study. This questionnaire contained general questions with regard to the experience of the developers (see Appendix A), specific questions for features of the framework, and the System Usability Scale (SUS) questionnaire [5].

In total we received filled-out questionnaires from 11 different study developers over the course of one and a half years of which one had to be excluded due to incompleteness. Of the remaining  $n = 10$  projects, three were bachelor thesis and four master thesis projects. The remaining three were in the context of different PhD projects in the realm of acoustic, psychology, and VR research. Answers to the questions regarding prior experience, ease of development, and confidence during study execution, which were rated on a 5-point Likert scale between 1 (*Strongly Disagree*) and 5 (*Strongly Agree*), can be found in Fig. 4. Additionally, when asked what the most helpful feature was, five study developers (50%) stated the status bar, three (30%) the “Next Condition” button, and one each stated the output log (10%) and “all of them” (10%). When asked for the least helpful feature seven participants (70%) answered “None”, while the “Next Condition” button, the “Show Conditions” button, and the output log were picked by one developer (10%) each.

Evaluating the results of the SUS yielded a mean score of 76.5 ( $SD = 16.6$ ) on a scale of 0 to 100, which is considered a “good” usability score. When looking closer at the individual scores, in five cases the framework was rated above 80, which constitutes an excellent score, however, in one case it was even rated as low as 40 (while all other ratings stayed above 65, which is the average SUS rating). Following up on the free field comments in this particular case did not yield any insights on a specific shortcoming.

However, when looking at the free comments of all developers regarding implementation ease, there are a few inconveniences and feature requests mentioned. Some of them (like logging custom data or debugging functionality requested) were already solved during further development of the framework, but two still persist. One study developer stated that *“The nested design of the study setup (phases etc.) was at times hard to digest, partly also because of the small fonts”* (see Fig. 2). Another comment hinted at the missing possibility to dynamically insert, e.g., *“a phase only between specific conditions of another phase”*, which goes beyond standard factorial design and was thus consciously not added. However, also several comments stated, e.g., *“the framework makes creating a study very easy, especially if you are new to Unreal”* or that *“it was easy to use for a beginner in study design and actually helped to understand the structure of studies with (in)dependent variables, factors and conditions”* and that the StudyFramework’s *“blueprints were really nice and easy to use.”* Looking into the comments regarding conducting the actual study, the only negative comments regarded shortcomings that were already fixed in the meantime or feature requests that would go beyond the targeted scope of this framework. One developer, for example, requested the possibility to reset the participant’s position which is very specific and should therefore rather be implemented for a study individually. Apart from that, there were positive comments like *“the possibility to restart the study where the participant left of as well as the good logging came in very handy”* and *“during execution of the study, it was pretty good. I could not think of something to immediately improve.”*

## 4 DISCUSSION

When looking at the results of questions **Q1 - Q3**, we can conclude that we had rather novice users: While some of them were more proficient in C++ development, nearly all of them were very new to Unreal and study design. Although the framework should also benefit more proficient developers in not having to write study code from scratch, the plugin was especially designed with Unreal novice users in mind. Therefore, it is very encouraging that they found *“the usage of the StudySetupActor clear and easy”* (**Q4**,  $M = 3.9$ ,  $SD = 0.3$ ). The “different randomization and ordering options” in

this setup, however, were apparently experienced as less clear (**Q5**,  $M = 3.2$ ,  $SD = 0.9$ ). This is also illustrated by the comment regarding the complexity of the nested design, probably referring to the nested setup of phases, factors, and levels. Partly, this might already have been improved, since we simplified the condition creation and balancing during further development, e.g., removing dedicated repetition functionality which could just as well be implemented with an additional factor. At the same time also more documentation and several examples were added to the Wiki of the respective git-project<sup>9</sup>. However, there might also be room for improvement with regard to the clearness of the randomization and balancing. The Wiki in general and the provided interfaces were rated well above average (**Q6 & Q7**). This can also be seen in the low ratings to the statement “I had to look into the source code frequently to understand what was going on” (**Q8**,  $M = 2.2$ ,  $SD = 0.8$ ). However, looking at the answers to “I needed a lot of help to develop the study” (**Q9**,  $M = 2.7$ ,  $SD = 0.7$ ) reveals that still some additional help was required, albeit that these ratings drop over time (the only “4”-rating was at the beginning of the evaluation, while the last ratings were “2”). That potentially hints that the quality improved over time, but could also be caused by other factors. Furthermore, the statement does not clearly differentiate between needing help to understand the *StudyFramework* or with study design in general.

These observations together with the relatively high system usability score bring us to the conclusion that the developed framework is well usable by novices to set up factorial-design user studies, which was one of our main goals. The pursuit of simplicity inevitably limits the ability to accommodate more intricate configurations, such as those requested by a developer to dynamically adjust conditional orders based on participants’ performance or choices. But overall the provided functionality seems sufficient and user-friendly.

Another important aspect of the framework is the support for experimenters during the execution of a study by means of a GUI. Developers were very grateful for the experimenter view and agreed that it was helpful (**Q10**,  $M = 4.5$ ,  $SD = 1.0$ ) and greatly agreed with the statement “I felt in full control over the study” (**Q11**,  $M = 4.0$ ,  $SD = 1.2$ ). Only one developer, using an early version of the framework, rated both with 2. Reported problems were subsequently fixed. Another participant asked for the possibility to restart an already started condition in the conditions list (using the “Go to” button, see Figure 3), which was initially deactivated during study runs but consequently provided to grant experimenters full control. A similarly enabling functionality is the recovery on crashed study runs (**Q12**,  $M = 3.6$ ,  $SD = 1.1$ ). However here, experimenters did not feel just as confident. This might have come, because they did not test it thoroughly before starting the study and did not implement it themselves, so they were not entirely sure what would happen. We, however, implemented this feature in a way that no data loss can appear, because potentially removed data from an already started, but not finished, condition would be backed up before removal. Generally, the features within our experimenter view often go unimplemented in many studies that are developed from scratch. Due to time constraints, essential functions are prioritized over convenience features, relegating these valuable additions to the bottom of the priority list. This is again a strong argument for using our proposed framework, as feeling in control while conducting a user study is a very reassuring feeling for the experimenter and potentially also increases the number of valid and useful data sets gathered. Which particular feature was useful for which study had a larger spread, this can be seen in the answers to **Q13** but also in the answers to the most and least helpful feature. While the status bar was liked most by a majority of experimenters, there is no clear preference for a least helpful feature that potentially could be removed, hinting that all implemented features are well integrated and supportive.

However, the evaluation is based on a limited number of subjective responses ( $N = 10$ ) and can therefore only be generalized

with caution. Furthermore, as stated, the framework was developed continuously during data gathering so that some features might have become more helpful in the process. Nevertheless, we are confident that the presented *StudyFramework* provided value to the surveyed developers and facilitated their development as it will for future study developers.

For future developments of our framework, our focus will be on enhancing user-friendliness, enabling new as well as experienced developers to quickly and effortlessly set up new studies, minimizing the likelihood of unforeseen issues arising during or after study execution. Thus, our strategy revolves around optimizing and streamlining our current functionality based on incoming developer feedback and future Unreal versions, while we will only incorporate new features that align with our vision of a light-weight, general framework in response to user requests. Additionally, we appreciate the initiative of Aguilar et al. [1] to increase the reproducibility of experiments. While our *StudyFramework* already contributes to this goal, we believe that the use of widely accepted common formal descriptions could further enhance reproducibility. Consequently, we are open to engaging in discussions to determine the appropriateness of our current *json* file implementation or explore whether other data formats might offer superior compatibility and reproducibility benefits.

## 5 CONCLUSION

We presented the *StudyFramework*, an open-source light-weight Unreal plugin specifically designed for novice developers to easily set up and conduct factorial-design user studies. The framework thereby primarily focuses on VR experiments in HMDs and CAVEs, while also being suitable for desktop-based studies. To date, the framework has been utilized for 14 VR-based user studies. An evaluation of responses from 10 involved study developers has confirmed its ease of use, streamlining the implementation process, and providing experimenters with a sense of control during study execution. We are committed to continuously enhance the framework's functionalities and compatibility with newer engine versions, with our current efforts directed toward making it compatible with Unreal Engine 5.3.

## ACKNOWLEDGMENTS

This research was funded by the German Research Foundation (DFG) within the project "Listening to, and remembering conversations between two talkers: Cognitive research using embodied conversational agents in audiovisual virtual environments", which is part of the DFG Priority Program "AUDICTIVE" (SPP 2236). Special thanks goes to Marius Schmeling, Patrick Nossol, and Malte Kögel for their support in developing this framework. Additionally, the authors want to thank Daniel Zielasko and Isabel Schiller for the fruitful discussion in conceptualizing this work.

## REFERENCES

- [1] L. Aguilar, M. Gath-Morad, J. Grübel, J. Ermatinger, H. Zhao, S. Wehrli, R. W. Sumner, C. Zhang, D. Helbing, C. Hölscher, and E. Zürich. Experiments as Code: A Concept for Reproducible, Auditable, Debuggable, Reusable, & Scalable Experiments. *arXiv preprint arXiv:2202.12050*, 25:2022, 2 2022. doi: 10.48550/arXiv.2202.12050
- [2] G. Bailey, O. Kammler, R. Weiser, S. Schneider, and E. Fuchkina. Integrating Immersive Virtual Environment User Studies into Architectural Design Practice: A Pre-Occupancy User Study of Train Station Waiting Preferences With VREVAL. *Proceedings of the 2022 Annual Modeling and Simulation Conference, ANNSIM 2022*, pp. 644–655, 2022. doi: 10.23919/ANNSIM55834.2022.9859371
- [3] A. O. Bebko and N. F. Troje. bmlTUX: Design and Control of Experiments in Virtual Reality and Beyond. *i-Perception*, 11, 7 2020. doi: 10.1177/2041669520938400
- [4] D. Bridges, A. Pitiot, M. R. MacAskill, and J. W. Peirce. The timing mega-study: Comparing a range of experiment generators, both lab-based and online. *PeerJ*, 8:e9414, 7 2020. doi: 10.7717/peerj.9414

- [5] J. Brooke. SUS-A quick and dirty usability scale. *Usability Eval. Ind.*, 189, 1995.
- [6] J. Brookes, M. Warburton, M. Alghadier, M. Mon-Williams, and F. Mushtaq. Studying human behavior with virtual reality: The Unity Experiment Framework. *Behavior Research Methods*, 52:455–463, 4 2020. doi: 10.3758/s13428-019-01242-0
- [7] A. Bönsch, T. Sittart, J. Ehret, and T. W. Kuhlen. Where Do They Go? Overhearing Conversing Pedestrian Groups during Scene Exploration. In *ACM International Conference on Intelligent Virtual Agents (IVA '23)*, p. 3. ACM, 2023. doi: 10.1145/3570945.3607351
- [8] A. Bönsch, L. B. Zimmermann, J. Ehret, and T. W. Kuhlen. Whom Do You Follow? Pedestrian Flows Constraining the User's Navigation during Scene Exploration. In *ACM International Conference on Intelligent Virtual Agents (IVA '23)*, p. 3. ACM, 2023. doi: 10.1145/3570945.3607350
- [9] S. Commins, J. Duffin, K. Chaves, D. Leahy, K. Corcoran, M. Caffrey, L. Keenan, D. Finan, and C. Thornberry. NavWell: A simplified virtual-reality platform for spatial navigation and memory experiments. *Behavior Research Methods*, 52:1189–1207, 6 2020. doi: 10.3758/S13428-019-01310-5
- [10] C. Cruz-Neira, D. Sandin, R. V. Kenyon, and J. C. Hart. The cave - audio visual experience automatic virtual environment. *Communications of The ACM - CACM*, 1992.
- [11] D. W. Cunningham and C. Wallraven. *Experimental Design: From User Studies to Psychophysics*. A K Peters/CRC Press, 2012. doi: 10.1201/b11308
- [12] A. L. Edwards. Balanced Latin-Square Designs in Psychological Research. *The American Journal of Psychology*, 64:598–603, 1951. doi: 10.2307/1418200
- [13] J. Ehret, A. Bönsch, P. Nossol, C. A. Ermert, C. Mohanathanan, S. J. Schlittmeier, J. Fels, and T. W. Kuhlen. Who's next? Integrating Non-Verbal Turn-Taking Cues for Embodied Conversational Agents. In *ACM International Conference on Intelligent Virtual Agents (IVA '23)*, 2023. doi: 10.1145/3570945.3607312
- [14] J. Ehret, A. Bönsch, I. S. Schiller, C. Breuer, L. Aspöck, J. Fels, S. J. Schlittmeier, and T. W. Kuhlen. Audiovisual Coherence: Is Embodiment of Background Noise Sources a Necessity? In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW): "Workshop on Virtual Humans and Crowds in Immersive Environments (VHCIE)"*, 2024.
- [15] C. A. Ermert, J. Ehret, T. W. Kuhlen, C. Mohanathanan, S. J. Schlittmeier, and J. Fels. Audio-Visual Content Mismatches in the Serial Recall Paradigm. In *49. Jahrestagung für Akustik, Hamburg, Germany, DAGA 2023*, pp. 1429–1430, 2023.
- [16] M. Feick, N. Kleer, A. Tang, and A. Krüger. The Virtual Reality Questionnaire Toolkit. *UIST 2020 - Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pp. 68–69, 10 2020. doi: 10.1145/3379350.3416188
- [17] A. S. Geller, I. K. Schleifer, P. B. Sederberg, J. Jacobs, and M. J. Kahana. PyEPL: A cross-platform experiment-programming library. *Behavior Research Methods*, 39:950–958, 2007. doi: 10.3758/BF03192990
- [18] J. Grübel. The design, experiment, analyse, and reproduce principle for experimentation in virtual reality. *Frontiers in Virtual Reality*, 4:1069423, 4 2023. doi: 10.3389/frvr.2023.1069423
- [19] J. Grübel, R. Weibel, M. H. Jiang, C. Hölscher, D. A. Hackman, and V. R. Schinazi. EVE: A framework for experiments in virtual environments. In *Spatial Cognition X - 13th Biennial Conference, KogWIS 2016*, vol. 10523 LNAI, pp. 159–176. Springer Verlag, 2017. doi: 10.1007/978-3-319-68189-4\_10
- [20] T. W. Kuhlen and B. Hentschel. Quo vadis CAVE: Does immersive visualization still matter? *IEEE Computer Graphics and Applications*, 34:14–21, 9 2014. doi: 10.1109/MCG.2014.97
- [21] R. L. Pedersen, L. Picinali, N. Kajs, and F. Patou. Virtual-Reality-Based Research in Hearing Science: A Platforming Approach. *Journal of the Audio Engineering Society*, 71:374–389, 6 2023. doi: 10.17743/JAES.2022.0083
- [22] J. Peirce, R. Hirst, and M. MacAskill. *Building experiments in PsychoPy*. Sage, 2022.
- [23] I. Schuetz, H. Karimpur, and K. Fiehler. vextoolbox: A software toolbox for human behavior studies using the Vizard virtual reality

- platform. *Behavior Research Methods*, 55:570–582, 2 2023. doi: 10.3758/S13428-022-01831-6
- [24] K. A. Shapcott, M. Weigand, I. Glukhova, M. N. Havenith, and M. L. Schölvinck. DomeVR: A setup for experimental control of an immersive dome virtual environment created with Unreal Engine 4. *bioRxiv*, 2022. doi: 10.1101/2022.04.04.486889
- [25] A. Solway, J. F. Miller, and M. J. Kahana. PandaEPL: A library for programming spatial navigation experiments. *Behavior Research Methods*, 45:1293–1312, 12 2013. doi: 10.3758/S13428-013-0322-5
- [26] M. J. Starrett, A. S. McAvan, D. J. Huffman, J. D. Stokes, C. T. Kyle, D. N. Smuda, B. S. Kolarik, J. Laczko, and A. D. Ekstrom. Landmarks: A solution for spatial navigation and memory experiments in virtual reality. *Behavior Research Methods*, 53:1046–1059, 6 2021. doi: 10.3758/S13428-020-01481-6
- [27] A. Steed, L. Izzouzi, K. Brandstätter, S. Friston, B. Congdon, O. Olkkonen, D. Giunchi, N. Numan, and D. Swapp. Ubiq-exp: A toolkit to build and run remote and distributed mixed reality experiments. *Frontiers in Virtual Reality*, 3:912078, 10 2022. doi: 10.3389/frvir.2022.912078
- [28] K. Tiwari, V. Kyrki, A. Cheung, and N. Yamamoto. DeFINE: Delayed feedback-based immersive navigation environment for studying goal-directed human navigation. *Behavior Research Methods*, 53:2668–2688, 12 2021. doi: 10.3758/s13428-021-01586-6
- [29] M. Vasser, M. Kängsepp, M. Magomedkerimov, K. Kilvits, V. Stafinjak, T. Kivisik, R. Vicente, and J. Aru. VREX: An open-source toolbox for creating 3D virtual reality experiments. *BMC Psychology*, 5:1–8, 2 2017. doi: 10.1186/S40359-017-0173-4
- [30] J. Vercelloni, J. Peppinck, E. Santos-Fernandez, M. McBain, G. Heron, T. Dodgen, E. E. Peterson, and K. Mengersen. Connecting Virtual Reality and Ecology: A New Tool to Run Seamless Immersive Experiments in R. *PeerJ Computer Science*, 7:1–14, 6 2021. doi: 10.7717/PEERJ-CS.544
- [31] M. R. Watson, B. Voloh, C. Thomas, A. Hasan, and T. Womelsdorf. USE: An integrative suite for temporally-precise psychophysical experiments in virtual environments for human, nonhuman, and artificially intelligent agents. *Journal of Neuroscience Methods*, 326:108374, 10 2019. doi: 10.1016/j.jneumeth.2019.108374
- [32] M. Wölfel, D. Hepperle, C. F. Purps, J. Deuchler, and W. Hettmann. Entering a new Dimension in Virtual Reality Research: An Overview of Existing Toolkits, their Features and Challenges. In *Proceedings - 2021 International Conference on Cyberworlds, CW 2021*, pp. 180–187, 2021. doi: 10.1109/CW52790.2021.00038

## A FULL QUESTIONNAIRE STATEMENTS

Nr.	Full Statement	Left Anchor (1)	Right Anchor (5)
Q1	Experience with Unreal Engine (UE) before starting the project:	Completely new to UE	Very familiar with UE
Q2	Experience with C++ programming before starting the project:	Completely new to C++	Expert in C++
Q3	Experience with factorial study design before starting the project:	Complete novice	Expert in study design
Q4	I found the usage of the study setup actor clear and easy.	Strongly Disagree	Strongly Agree
Q5	The different randomization and ordering options were clear to me.	Strongly Disagree	Strongly Agree
Q6	The Wiki helped in finding the information I needed.	Strongly Disagree	Strongly Agree
Q7	The C++/Blueprint Interfaces provided were sufficient.	Strongly Disagree	Strongly Agree
Q8	I had to look into the source code frequently to understand what was going on.	Strongly Disagree	Strongly Agree
Q9	I needed a lot of help to develop the study.	Strongly Disagree	Strongly Agree
Q10	The provided control screen helped conducting the study.	Strongly Disagree	Strongly Agree
Q11	I felt in full control over the study.	Strongly Disagree	Strongly Agree
Q12	I felt confident that I could use the provided recovery options to handle every possible situation.	Strongly Disagree	Strongly Agree
Q13	I used the "Show Conditions" Option regularly.	Strongly Disagree	Strongly Agree

Table 2: Full Statements and their scale's anchors used in the questionnaire to evaluate the *StudyFramework*. Additionally, the System Usability Scale (SUS), two rankings for most and least helpful features, free feedback fields regarding implementation and execution, as well as demographics, were part of the questionnaire.