BeatriXR: Comprehensive and Adaptive Feedforward Support for Guidance in Virtual Reality

VALENTINO ARTIZZU, University of Cagliari, Italy
KRIS LUYTEN, Hasselt University - Flanders Make, Belgium
GUSTAVO ALBERTO ROVELO RUIZ, Hasselt University - tUL - Flanders Make, Belgium
LUCIO DAVIDE SPANO, University of Cagliari, Italy

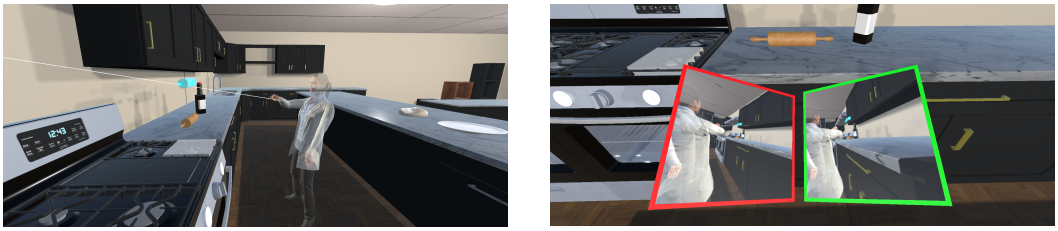
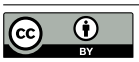


Fig. 1. BeatriXR supports different implementations of direct feedforward, which allows users to see the results of the possible interactions in a virtual environment. On the left part, an avatar demonstrates how to select an object from afar. On the right, two panels show two alternative completions for the same interaction.

Virtual Reality (VR) environments challenge users with varied input devices, interaction methods, and interface designs, resulting in a steep learning curve. Feedforward "informs the user about what the result of his action will be", and it allows to ease the process of learning of the end user by providing ways to represent the required action to perform, using contextualised previews that show how to complete a given interaction. Yet, creating effective direct feedforward without specialised tools remains a tedious process. We present BeatriXR, a flexible and adaptive toolkit that simplifies the creation of direct-feedforward configurations in VR. It enables users to build, visualise, and customise direct feedforward using virtual avatars, offering both in-world representations and on-screen comparisons of interaction options. Mapped to the recognised design-space by Muresan et al. (including Triggering, Previewing, and Exiting phases), BeatriXR streamlines development and helps ensure effective feedforward integration in VR environments. As additional support for the feedforward design task, BeatriXR features an LLM-powered decision-support layer that suggests possible configuration options. This guidance helps designers select optimal settings during development for adapting the presentation to user needs and the expected configuration of the interaction context. We conducted an exploratory review with XR domain experts who rated the UI for modifying feedforward settings, and assessed four LLM models in the task of suggesting possible configuration options. The participants' feedback was positive and provided valuable insights for improving both the user interface, which was generally perceived positively, and the quality of the LLM-generated responses.

Authors' Contact Information: Valentino Artizzu, Mathematics and Computer Science, University of Cagliari, Cagliari, Italy, valentino.artizzu@unica.it; Kris Luyten, Digital Future Lab, Hasselt University - Flanders Make, Diepenbeek, Belgium, kris.luyten@uhasselt.be; Gustavo Alberto Rovelo Ruiz, Expertise Centre for Digital Media, Hasselt University - tUL - Flanders Make, Hasselt, Belgium, gustavo.roveloruiz@uhaselt.be; Lucio Davide Spano, Department of Mathematics and Computer Science, University of Cagliari, Cagliari, Italy, davide.spano@unica.it.



ACM 2573-0142/2026/6-ARTEICS008
https://doi.org/10.1145/3816419

CCS Concepts: • **Applied computing → Interactive learning environments**; • **Human-centered computing → Virtual reality**; Graphical user interfaces; User interface programming.

Additional Key Words and Phrases: virtual reality, direct feedforward, avatars, education, LLM

ACM Reference Format:
Valentino Artizzu, Kris Luyten, Gustavo Alberto Rovelo Ruiz, and Lucio Davide Spano. 2026. BeatriXR: Comprehensive and Adaptive Feedforward Support for Guidance in Virtual Reality. *Proc. ACM Hum.-Comput. Interact.* 10, 4, Article EICS008 (June 2026), 32 pages. https://doi.org/10.1145/3816419

1 Introduction

In recent years, Virtual Reality (VR) has gained traction across various domains such as entertainment, education, and professional training. Research indicates that VR serves as an effective training tool, particularly for skill acquisition in high-stakes fields like medicine [20], manufacturing [32], and aviation [19]. VR allows novices to practice in realistic environments without real-world repercussions, enhancing learning through immediate feedback and facilitating the transfer of skills from virtual to actual tasks [37].

Interacting with VR systems can be challenging, as users might miss important cues due to the complexity of these environments. Such difficulties can lead to frustration or errors if users need to figure out the interactions before exploring the content.

To improve user interaction, incorporating feedforward mechanisms can be beneficial. Feedforward informs users about the consequences of their actions before execution [15, 38]. Muresan et al. [27] categorize feedforward into direct, which shows visual simulations, and indirect, which uses cues to suggest actions. Their design space outlines three phases: triggering, previewing, and exiting, aiding designers in structuring feedforward presentation [2, 40, 41].

Designing and implementing appropriate feedforward, particularly direct feedforward, can be tedious without dedicated tools to simplify their integration into virtual environments. In practice, developers must coordinate multiple interdependent aspects (e.g., when to trigger feedforward, how to visualize actions and outcomes, and how to transition back to the interaction state), which are often implemented in an ad-hoc manner within game engines. The lack of such support could limit the options designers consider or the techniques a single application could employ. The process would be easier with a toolkit that provides reusable feedforward components.

Even with a toolkit available, finding the right combination of visual and interaction features for a feedforward representation is challenging. The number of possible combinations within the design space makes systematic exploration difficult, especially for non-expert developers. Large Language Models (LLMs) could provide support in exploring this design space by considering the available configuration options together with contextual information.

This paper introduces BeatriXR, a VR toolkit supporting the authoring of direct feedforward while covering the scope of feedforward types proposed by Muresan et al. [27]. The toolkit enables domain experts to configure and deploy feedforward in VR applications selecting specific options over the full conceptual spectrum, providing comprehensive support for both design-time specification and runtime delivery within VR environments. By operationalizing this design space into reusable components and interface controls, BeatriXR reduces the effort required to implement and iterate on feedforward solutions. In addition, BeatriXR integrates an LLM-based module that assists domain experts in defining suitable feedforward configurations for different interaction types. This component is intended as a mixed-initiative aid to support configuration exploration.

The contributions of this work can be summarised as follows:

(1) A modular VR toolkit that supports the creation, visualization, and customization of direct feedforward representations using avatars and/or interface panels, mapped onto an established feedforward design space and aligned with the various phases of interaction (triggering, previewing and exiting).
(2) An LLM-driven decision support layer, in which generative models propose configuration alternatives, to assist designers during exploration of the design space.
(3) A proof-of-concept implementation, developed in Unity and integrated with MRTK3, demonstrating the use of BeatriXR in an immersive training scenario[1].
(4) An exploratory evaluation with XR experts assessing the toolkit, the usefulness of its configuration interface, and the role of LLM-generated suggestions in supporting feedforward design.

2 Related Work

Feedforward. Feedforward is a relatively unexplored concept that occupies the opposite end of the temporal spectrum from feedback; while feedback provides information about what an action did, feedforward allows users to anticipate "what the result of their action will be" before it is executed [38]. This mechanism is grounded in the ability to couple available actions and possible reactions across dimensions of time, location, direction, dynamics, modality, and expression [39]. Specifically, feedforward can be categorized as inherent (information on available actions and how to perform them), functional (the purpose or features activated), or augmented (supplemental information such as labels, icons, and UI elements). While prior research has extensively covered the theoretical foundations [8, 28, 29, 38, 39] and 2D interface applications [7, 14, 18, 36], Muresan et al. [27] provide a foundational design space for VR. Their framework identifies three key stages (*triggering*, *previewing* actions and outcomes, and *exiting* the preview) and establishes parameters for identifying appropriate techniques for 3D interaction.

Building on this, recent systems have begun to implement specific feedforward cues. ViRgilites [2] leverages these categorizations to support procedural adaptation in training via indirect cues like UI icons and object highlighting, focusing on automatic support rather than design exploration. Other systems address motor task guidance: Fennedy et al. [16] adapted the Octopocus technique [7] to 3D to show dynamic traces of gestures, while Just Follow Me and Lilija et al. [23] utilize "ghosting" techniques to superimpose virtual hands or bodies that anticipate required movements. Similarly, JollyGesture [30] supports gesture disambiguation via path previews, Yu et al. [41] propose a motion feedforward design space, and GeneratiVR [21] employs visual previews for object placement. Beyond motor tasks, tools like LightGuide [34] and RayCursor [4] use arrows and traces to suggest movement directions or interaction targets. While the existing work in the literature address specific aspects of providing feedforward in VR enviroments (e.g., onboarding [11] or performance effects [5]), none of them fully implement tthe breadth of Muresan et al.'s guidelines (a comparison table is available in Appendix A).

BeatriXR extends these ideas by offering selectable, direct feedforward techniques (e.g.: animated avatars) without requiring programming from domain experts. By integrating specialized LLMs, our system allows experts to define customized parameters for each action, which are then suggested to learners during execution. We utilize arrows and traces to guide users toward interactive objects and provide automatic teleportation to targets when necessary.

Avatar representations for education. Many feedforward configurations supported by BeatriXR exploit avatars. Their use is documented in the VR-based education literature and it enhances learning through immersive, personalized interactions, often guiding students with human-like

[1]The toolkit code is available at the following URL: https://github.com/cg3hci/BeatriXR

traits and adapting feedback to individual actions [17, 22, 25]. They support active learning via role-play and scenario-based exercises that build communication, decision-making, and critical thinking skills [3, 6, 10, 31]. Recent advancements integrate LLMs for real-time, tailored instruction [17], though reliability remains limited by issues like hallucinations.

Large Language Models for Decision Making. LLMs have become valuable tools for supporting informed decision-making in AI-powered conversational agents. In design and planning contexts, they assist domain experts by simplifying complex decisions across areas such as strategic planning, predictive modelling, and clinical support [33]. Research has also demonstrated their effectiveness in game-theoretic simulations [24], emergency management [12] and social dilemmas [9], with models like GPT-4 showing human-level or superior performance in economic rationality [13] and theory of mind tasks [35]. We explore the idea of using LLMs to guide designers in the complex design space proposed by Muresan et al. [27].

3 BeatriXR: An Integrated Toolkit for Adaptive Feedforward in VR

In this section, we describe the support offered by BeatriXR for both creating and experiencing feedforward in VR. From the user's interaction point of view, BeatriXR consists of three main elements.

The **User Interface** guides the user in the execution of the scene, containing information about the task steps, actions and possible interactions for each action, using indirect feedforward techniques. It also shows information on the progress and the next action he will perform on successfully completing the current one;

The **Avatar Representations** are enabled on user request, and perform a demonstration of the interaction the user will have to do, using direct feedforward techniques;

The **AI module**, an external component that can be queried either at run-time with its dedicated API (for suggesting on-the-fly changes through analysis of the context) or at design time (for suggesting alternative configurations to the domain expert).

In the remaining part of the section, we discuss the features and the interactions each one of these elements. We start introducing a scenario showing an example guidance for a procedure in a chemistry lab. Then, we detail the interactive control over the guidance system provided through the user interface and the feedforward techniques supported through the avatar. Finally, we discuss LLM-based adaptive guidance and the suggestions for the domain experts that define tasks. Details about the technical implementation are available in Section 3.6.

3.1 Example Scenario

We consider a training environment where a trainee is learning to handle hazardous materials in a chemistry lab. The task consists of the following steps:

(1) Use the computer to initiate the procedure by interacting with the mouse placed nearby in the scene;
(2) Pick up a pair of gloves from the table;
(3) Insert a vial containing hazardous waste into a beaker;
(4) Grab the hazardous beaker from the table;
(5) Place the beaker into the biohazard waste box;

The domain expert, before providing the environment to the user, proceeds to define the feedforward settings inside of it, using a special version of the UI seen in section 3.2.2 that includes two extra buttons at the bottom, "Save Defaults" and "Save Override for this step". BeatriXR already has a set of options that we have defined in advance, that we deemed to be generally good for most of the cases, but using the in-world interface they can be freely changed. In the same fashion,

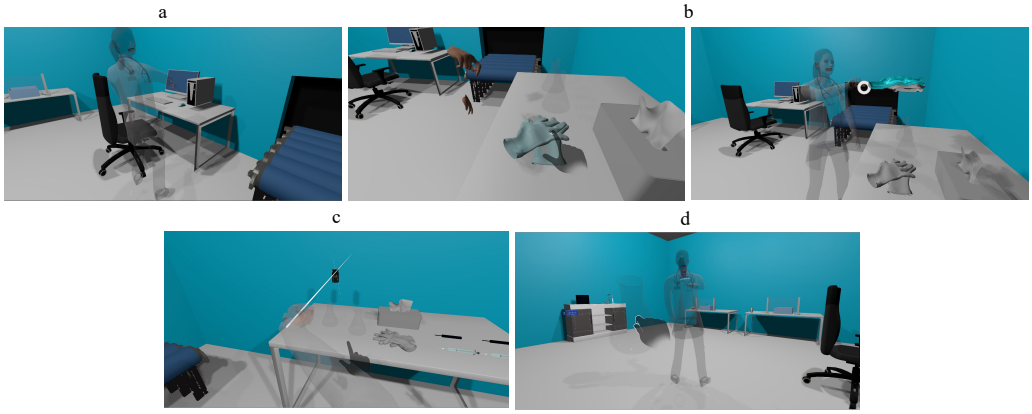


Fig. 2. Hazardous Waste Scenario: (A) Initiating program using the mouse, (B) grabbing gloves (before and after adjusting feedforward settings), (C) Transferring hazardous waste into a beaker, (D) Transporting the beaker to the biohazard waste box.

the domain expert can move between the steps and the interactions of the procedure using the interface in section 3.2.1 and once he defines the overrides he can save them. At the end of the setup, for the first step, a semi-transparent (ghosted) avatar will show the current step. For steps 2 and 3, animated virtual hands will automatically demonstrate the required actions. In the final two steps (4 and 5) the trainee will be given control to trigger the animation manually.

At the simulation start, the trainee stands at the room center and explores for interactive objects. Upon locating the mouse, a ghosted avatar moves toward it, touches it (Figure 2-A), and disappears. The trainee replicates the action and completes the step.

The Task Panel (Figure 2-B) directs the trainee to pick up gloves. An animation shows virtual hands grasping a ghosted glove model. Unsure about what to do, the trainee opens the Feedforward Settings panel (Figure 4), selects the *Full* ghosted avatar, and disables glove transparency. He then observes the avatar and mimics the gesture.

Next, the trainee must place a hazardous vial into a beaker. The AI retains the trainee's modified settings, overriding the expert's defaults. A ghosted avatar places the vial with laser pointing (Figure 2-C), then it resets. Wanting to act alongside the animation, the trainee selects the *Persistent* trigger, performs the task, and completes the step.

In the following step, the expert's settings require proximity-triggered feedforward, and the feedforward is automatically adjusted following the overrides defined by the domain expert. The user, upon approaching the beaker, triggers the feedforward representation that automatically begins, so he watches it and then uses his virtual hand to pick up the beaker (Figure 2-D).

Finally, to dispose of the beaker, the trainee replays the animation via the hand menu using the Playback control panel (Figure 5) and completes the task.

3.2 The User Interface

The scenario illustrates that the trainee utilizes a UI designed for task guidance, featuring two main functions: displaying task progress with available interactions and managing feedforward settings and playback in the virtual scene.

BeatriXR supports these features through two UI panels. The **Task Panel** allows the user to understand the progress inside the tasks execution (see Figure 3). The **Feedforward Settings**

Fig. 3. BeatriXR task guidance and feedforward comparison panels. The User Interface shows the progress of the user in the execution of the task, while also providing means for comparing interactions for a particular action.

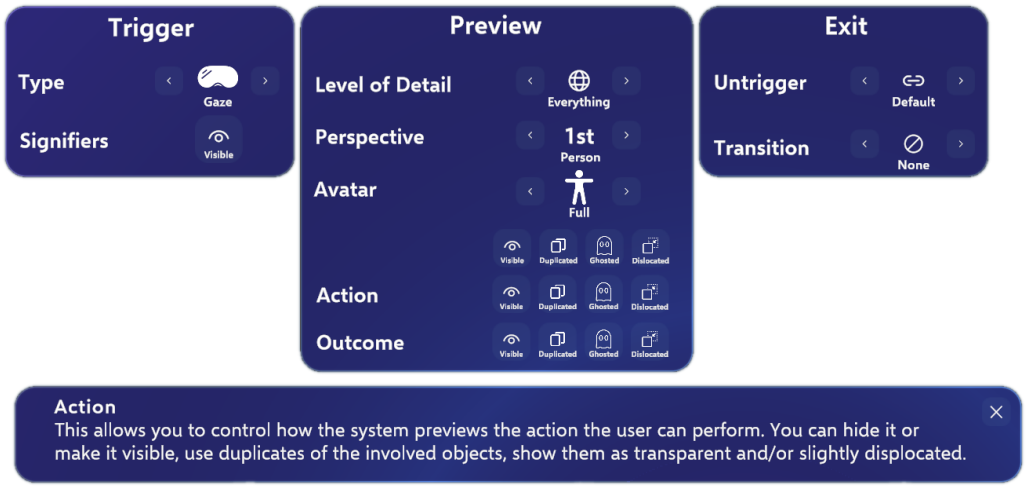


Fig. 4. BeatriXR shows the feedforward settings into three groups: *Trigger*, *Preview*, and *Exit*. Domain experts and trainees can browse setting options using arrow buttons (e.g., *Perspective* in *Preview*) or toggle flags on/off (e.g., *Visible* for *Action* to preview the required action). An explanation for the selected setting appears in the bottom panel.

Panel let the user tweak the direct feedforward experience by modifying various related settings (see Figure 4).

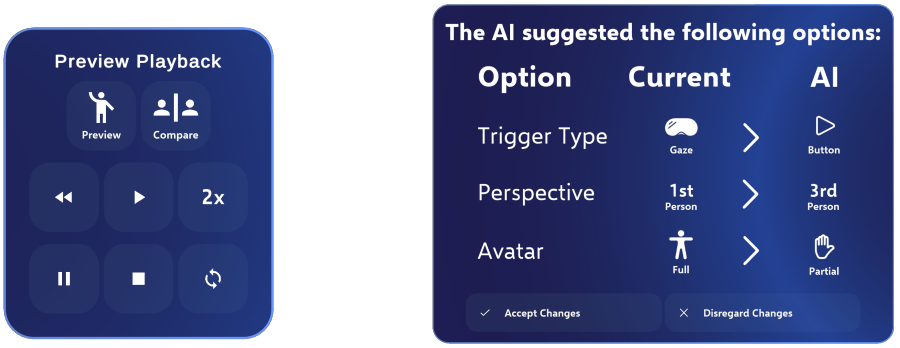


Fig. 5. BeatriXR panel for interacting with playback preview and AI-driven adjustments of feedforward settings.

3.2.1 The Task Panel. The Task Panel (see Figure 3) has three parts: the central part displays available actions and alternative interactions for completing them; each action can be performed through multiple interactions. The left panel shows the last successful action, while the right panel indicates the next expected action.

We represent alternative actions in the central panel as a stack of cards, each detailing the action instructions, required interactions, target objects, and a description. Users can navigate options by swapping the top card using arrow buttons. When an interaction is completed, it moves to the left panel, and the central panel displays possible next actions, indicating task progress. The left panel shows the most recent successful interaction. This design emphasizes actions over objects to focus feedback on the process rather than the specific object manipulated.

The right panel shows the user's next steps after completing the current action, including potential interactions. We used the algorithm in [2] to prioritize interaction modalities by scoring them based on static and dynamic parameters, such as fidelity, distance, modality switch cost, and context. Each interaction is assessed through a weighted dot product, with the highest-ranked option suggested to the user. In particular, the parameters taken in consideration are in part dynamic (e.g.: distance between the target object and the user, context information - considering factors like environment sensing, a trainee's past interactions, and unexplored options) and in part static (e.g.: interaction fidelity, cost of modality change - from one interaction type to another). In our proof-of-concept implementation, the information regarding the previous experiences of the trainee are verbally asked and then manually put inside the set of processed data that the system will analyse during the environment execution.

3.2.2 The Feedforward Settings. The user can change the feedforward configuration at any step during a task by looking at the palm of the non-dominant hand, which displays a menu with an activation button. Figure 4 presents options for configuring the feedforward, organized into three columns: Trigger (how to start animation), Preview (controlling the display settings), and Exit (animation ending). Each option controls the feedforward characteristics we better detail in Section 3.4 and 3.3. Users can select single-choice options (e.g., avatar type) by navigating with the previous and next buttons. Toggle buttons manage boolean flags (e.g., *visible* or *ghosted*), and interacting with these options displays explanatory text at the panel's bottom.

Another button in the non-dominant hand menu shows the *Animation Playback* panel (see Figure 5, left part), allowing control over the avatar animation playback, including options to start,

Fig. 6. The avatar representations of BeatriXR. On the left: the single representation (shown as the avatar demonstrating how to perform a touch interaction to a mouse); on the right: the multiple representations (shown as panels demonstrating grab and far interactions).

pause, resume, rewind, fast-forward, adjust playback speed, and loop the animation. Users can also manage feedforward animation with two buttons: the single avatar representation (see section 3.3.1) for a suggested interaction preview, and the multiple avatars representation (see section 3.3.2) for comparing different interactions for the same action.

3.3 The Avatar Representations

BeatriXR enables users to engage with the interface using virtual avatars that visually illustrate interactions. These avatars display gestures only when necessary. In BeatriXR, interactions can be represented by a single avatar for the current action (Figure 6, Left) or by a panel comparing two interactions simultaneously (Figure 6, Right).

3.3.1 Single Avatar Representation. The single avatar representation, activated through the Task Panel, disembodies from the user to demonstrate the interaction. Depending on the suggested interaction technique for the current step, the avatar may need to move closer (e.g., for *grabbing)* to the target object, or stays in the current position (e.g. for a *far interaction*). Based on the *Duplicated* flag, a copy of the target object may appear, with the avatar demonstrating the result on either the copied representation or the original object. The *Dislocated* flag determines if the copied object appears at the same location or is shifted for distinction. We set a distance to clearly differentiate the copied object from the original. After the animation, the copies of both the avatar and target object will disappear by default. It is important to emphasize that the avatar does not possess prior knowledge of the environment. Instead, it utilizes Inverse Kinematic[2] techniques to adapt its animation and orient itself towards the target object.

3.3.2 Multiple Avatars Representation. The multiple avatars representation in BeatriXR builds upon the single avatar mode, reusing most of its core functionalities while offering a different interaction flow. To activate this view, the user must select two different interactions (when available) and then press the *Compare* button to trigger the animations. In this mode, the avatars are not placed within the user's environment but are instead displayed in separate panels, each performing its animation independently (see Figure 6, right part). The animation technique remains the same as in the single avatar case, and each animation is viewed through a shoulder-mounted virtual camera.

[2]https://en.wikipedia.org/wiki/Inverse_kinematics

3.4 Triggering, Previewing, and Exiting: The Spectrum of BeatriXR's Guidance Capabilities

Both the settings and the avatar representations implement the design space of Muresan et al. [27] and makes it possible to customise the feedforward on-the-fly. In this section we briefly recall the different components of the design space and we discuss how they are supported in the BeatriXR. We adopt the Trigger–Preview–Exit decomposition as the core abstraction of BeatriXR, as it provides a clear separation of concerns between when feedforward is presented, how it is visualized, and how interaction resumes. This separation supports modular implementation and enables independent exploration of each dimension, which is difficult to achieve with monolithic feedforward designs.

Triggering stage. It is the mechanism of starting the preview of actions and outcomes in the Virtual Environment. Figure 7 shows the different options supported by BeatriXR for triggering the feedforward animation, which include all types defined in [27]: 1) *Gaze* (Panel A), when the user looks at the target object; 2) *Position* (Panel B), when the user moves nearby the target object; 3) *Timer* (Panel C), which automatically triggers the animation at regular intervals based on a predefined time setting; 4) *Action* (Panel D), which activates the animation through the dedicated button in the playback interface; 5) *Persistent* (Panel E), which loops the animation, restarting it immediately after it ends.

The *Signifiers* (Panel F) are hints guiding the attention of the users towards the target objects the users can decide to show (*visible*) or hide. BeatriXR uses arrows pointing to the object of interests.

Previewing stage. It is the the act of showing "the users a preview of some actions and their outcomes" after triggering it. Figure 8 shows BeatriXR *previewing* capabilities. The *Level of Detail* (Panel A), defines the complexity of the animation. It can either display the animation of the entire task from start to finish (*Everything*, first panel), a partial sequence showing only the key steps in the task (*Key Steps*), or a single animation focused exclusively on the current step (*Current Step*, second panel). The current step is based on user progress, while key steps are marked by the domain expert during task definition.

The *Targets* (Panel B) allow controlling the different parts of the preview. Each of them can be *visible* or not, could be temporarily *duplicated* for showing the preview or involve the original object (Panel G), could be *ghosted* (semi-transparent) or solid (Panel F), and it can be *dislocated*, i.e., translated a bit from the original position to distinguish it from the original version (Panel E). We have three types of targets:

- The *Avatar*, which is the representation of the user within the VR environment during the preview. BeatriXR provides different options for displaying it (Panel C). The *full avatar* (first panel) shows a humanoid character demonstrating what the trainee should perform in the virtual environment. The *partial avatar* (second panel) shows only the relevant parts of virtual character for the considered interaction (the hands in our example), and *real user* (third panel), where the humanoid character that performs the movements the user should do in real world (useful if the mapping between the tracking support and the virtual representation of the user is not direct).
- The *Action*, which is a visual representation of what the user has to perform. It is implemented as an arrow and a panel with the interaction modality attached to it. The arrow approaches and stays near the object as long as the interaction has to be performed, and it moves away when the interaction is completed;
- The *Outcome*, showing the effect of the action in the virtual environment.

The *Perspective* (Panel D) defines the point of view of the user, either *first person* (first panel) or *third person* (second panel);

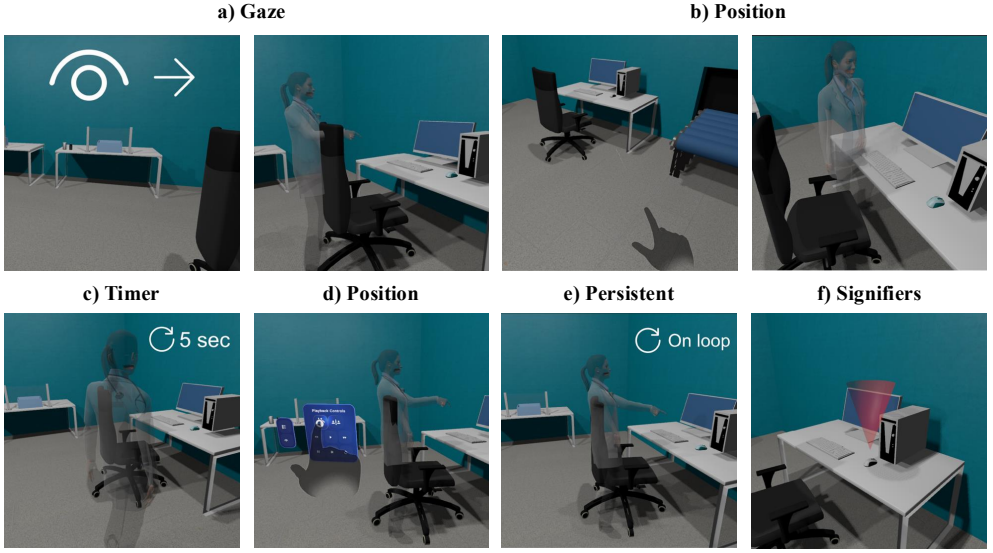


Fig. 7. Triggering capabilities for BeatriXR. The supported settings are: Gaze (Panel A), Position (Panel B), Timer (Panel C), Action (Panel D), Persistent (Panel E). Signifiers (Panel F) show the presence of a representation for the target object.

Exiting stage. It identifies two aspects: i) when to stop the previews and ii) how to stop them. Figure 9 shows the options for controlling how to exit from the preview and getting back to the current state of the virtual environment. It is possible to *Untrigger* the preview animation by *Gazing Away* from the target object (Panel A), *Moving Away* from the target object area (Panel B), or to end it *Automatically* as the animation finishes (Panel C). The *Transition* back to the current state of VR environment it is possible to *Rewind* the preview from the end to the start (Panel D), to *Instantaneously return* to the original state (Panel E) and to use a *Fade in to fade out effect* at the end of the preview (Panel F).

3.5 Artificial Intelligence for the Domain Expert

The design space outlined in [27] can involve numerous combinations, making it challenging to identify optimal configurations for each task step. In order to ease the exploration of the design space, we introduced in BeatriXR an LLM-based recommender as an independent module, which could be activated at design time, acting as a specialized assistant to the domain expert to streamline the training task workflow.

The LLM has been provided with knowledge about the design space through prompting and it aids domain experts by analysing their current setup and offering various viable options for implementation. It receives the information about the current feedforward settings through the Unity3D editor integration, together with the task sequences, represented in the format specified in [2]. In this way, the designer does not need to summarise information about the training procedure or the the knowledge about feedforward in VR. The LLM generates recommendations considering alternative configurations to enhance feedforward representation, presenting these overrides to the expert for potential adoption or further customization. Approved overrides are integrated into the

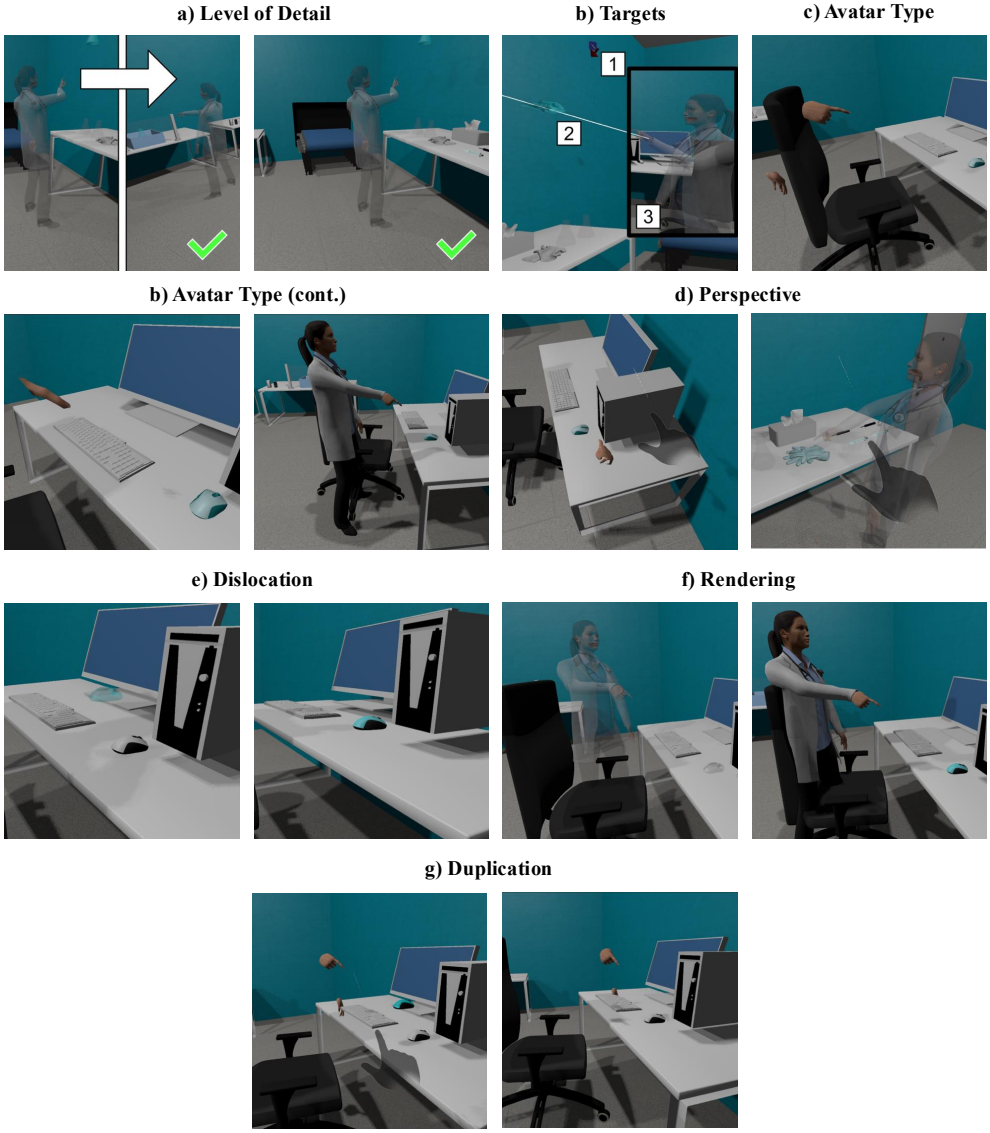


Fig. 8. Previewing capabilities for BeatriXR. The supported settings are: Level of Detail (Panel A), visibility of the Targets (Panel B, where 1 is the Action, 2 is the Outcome and 3 is the Avatar), Avatar Type (Panel C), Perspective (Panel D, it can be either first or third person), Dislocation (Panel E, it can be either in the same position of the original or in a slightly dislocated position), Rendering (Panel F, it can be either opaque or transparent), Duplication (Panel G).

task definition file, which contains default configurations and expert-linked overrides for specific actions.

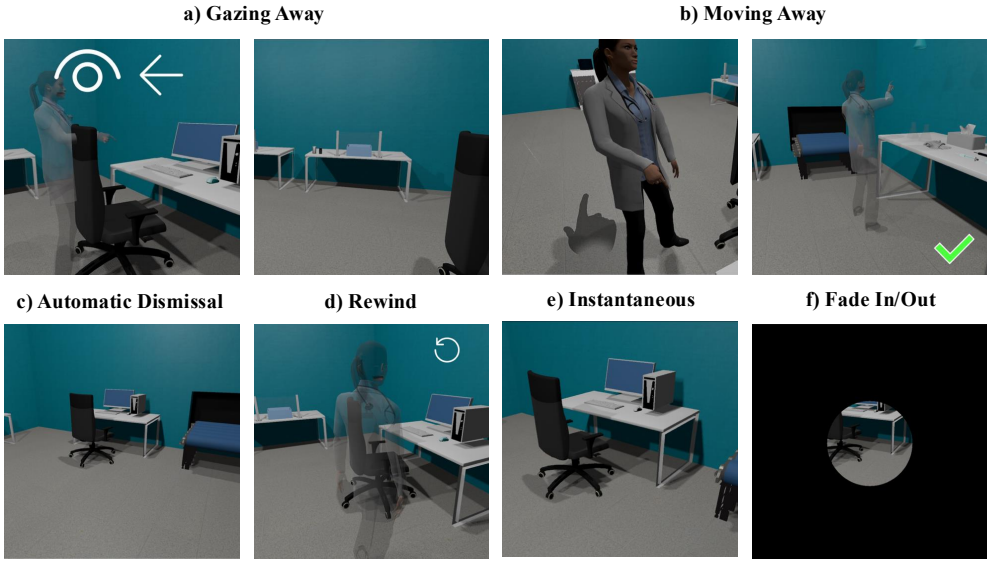


Fig. 9. Exiting Capabilities for BeatriXR. Untriggering settings support: Gazing Away (Panel A), Moving Away (Panel B), Automatic dismissal (Panel C). Exit Animation settings support: Rewinding of the animation (Panel D), Instantaneous return to the original state (Panel E), Fade in/Fade out effect (Panel F).

3.6 BeatriXR Technical Architecture and Modular Components

This section discusses the implementation architecture of BeatriXR. The system uses the Unity Game Engine and the MRTK 3 toolkit [26] for managing the environment and the user interactions.

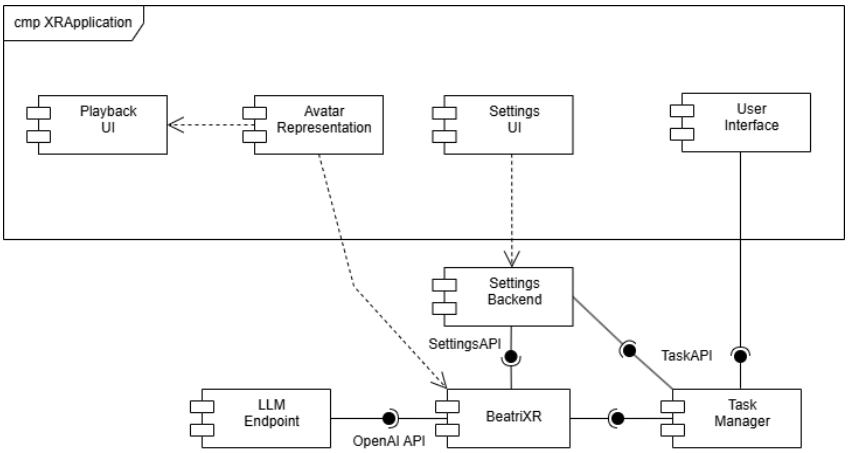


Fig. 10. UML component diagram for BeatriXR

MRTK allows BeatriXR to support complex VR interactions with minimal setup, requiring only the assignment of MRTK components to scene elements.

BeatriXR includes seven modules connected via APIs (see Figure 10), allowing developers to swap components with different implementations as long as API requirements are met, ensuring flexible and consistent results.

Task Manager. The *Task Manager* module manages tasks, tracks user progress, and updates actions as each is completed. Tasks are defined in a JSON file with an ordered list of actions, each described with alternative interactions. Each interaction specifies its description, modality, target object, and responsible user. The *User Interface* module accesses this information via the API for display.

The previous modules follow the ViRgilites [2] architecture for context-aware task guidance in immersive environments. We use a modular design with explicit task management, adaptive task selection based on context, and a user interface for choosing and executing required interactions.

BeatriXR. The *BeatriXR* module enables users and the task manager to customize the feedforward experience through dedicated interfaces, beyond simply displaying the avatar. Users can also control avatar playback. Our implementation integrates several modules: i) *avatar representation*, ii) *settings backend*, iii) *settings UI*, iv) *playback UI*, and v) *LLM endpoint*.

Avatar Representation. The *Avatar Representation* uses a Rocketbox model[3] with custom animations for each interaction type (e.g., Gaze, Grab, Far Interaction, Touch, Repositioning) created using Unity's animator. Supported by Inverse Kinematics, these animations are personalized on the fly. The avatar's mesh can appear opaque or transparent, fully or partially, based on selected settings.

Settings Backend and UI. The *Settings Backend* collects interaction type and target object data from the *Task Manager* via the TaskAPI to initiate the appropriate animation using Unity calls. The *Settings UI* allows configuration of how the animation is shown. In multiple-avatar mode, the same task is animated for each avatar, but avatars are hidden in the scene and displayed through separate camera renderers.

Playback UI. The *Playback UI* allows controlling the animation while the avatar is shown and playing it. This is done by connecting UI elements to Unity functions that modify the avatar's Animator's animation speed.

Avatar Representation. The *Avatar Representation* uses a model taken from the Rocketbox library[3] and we have created a set of animations that represents each interaction type (Gaze and commit, Grab, Far Interaction, Touch, Repositioning) using the built-in Unity model animator. As mentioned in section 3.3.1 the animations are also supported by Inverse Kinematics techniques, so they can be personalised on the fly, without an additional developer intervention. Depending on the selected settings the avatar receives variants of its default mesh, in order to get displayed either in an opaque or transparent appearance, partially or fully.

Settings Backend and UI. To accurately represent the intended animation, the *Settings Backend* module gathers essential data from the TaskAPI of the *Task Manager*, specifically the interaction type and the target object. Subsequently, the animation is initiated through Unity function calls, selecting animations that correspond to the specified interaction. Depending on the configuration settings available either from the start, or subsequently modified with the *Settings UI* module, the animation is then displayed in a manner tailored to these parameters. For the multiple avatar representation, in particular, the same task is done twice (one for each selected representation) but the avatars are not displayed directly into the scene. Indeed, they are hidden from the view of the

[3]https://github.com/microsoft/Microsoft-Rocketbox

user using Unity layers, and shown into two renderers that display two separate cameras that show those specific layers.

Playback UI. The *Playback UI* allows the control of the animation while the avatar is shown and playing the animation. This is done by connecting UI elements to Unity functions that modify the animation speed of the Animator of the avatar.

LLM Endpoint. The *LLM endpoint* enables the system to communicate with an external chatbot. In our implementation, we utilise the OpenAI-Unity package[4], modifying the endpoint and API key to connect to a local server managed by LM Studio[5]. The server is configured with a system prompt (see Appendix D) detailing the task, possible feedforward options, their pros and cons, and fictional weights to guide the chatbot's prioritisation. For each server call, the *LLM endpoint* includes the current feedforward settings representation. To ensure consistent and reliable responses from the LLM, we set inference options as follows: Top-p sampling (minimum cumulative probability for next token) to 0, Top-K sampling (limiting next token to the most probable) to 1, and Temperature (answer randomness) to 0. These settings maximise determinism, ensuring the model produces the same output for identical inputs and delivering stable, predictable results.

4 Validation and Expert-Based Evaluation

4.1 Validation

In this section, we demonstrate how BeatriXR operationalises and validates the design space proposed by Museran et al. [27]. Table 1 provides a detailed breakdown of the implementation status across all components and highlights the extent to which our toolkit realises each part of the framework. Overall, BeatriXR offers comprehensive support for the vast majority of the design space, with only two components—audio and haptic modifiers—remaining partially implemented. This limitation is deliberate: our research focuses on advancing visual cue technologies, and therefore prioritises depth and robustness in the visual dimension over full multimodal coverage. Despite this, the existing support demonstrates an extensive coverage of the feedforward configuration options.

4.2 Intermediate Feedback

Before creating the final version of the authoring interface we presented in Section 3.2.1, we assessed a preliminary version of its graphical representation through a mockup in a formative session. It involved 6 experts, 4 male and 2 female, in age range 25-41, all with one year of experience or more in the field of Extended Reality. The goal was understanding whether the experts can, through the proposed interface, identify feasible feedforward configurations for three different tasks:

- **T1)** Starting the program on a computer, using a touch interaction for clicking the virtual mouse in the VR scene;
- **T2)** Grabbing a beaker, using a laser-pointing technique, i.e., a ray starting from the hand or the VR controller allowing pointing from afar;
- **T3)** Turning on a sink, using a grab technique, i.e., the user joins the index and the thumb fingers in proximity of an object to start manipulating it.

For each task, we ranked the feasibility of all possible values of the feedforward configuration between 0 and 1. This allowed us to assess the proposed configuration in a $[0, 15]$ interval, where 0 is the worst solution and 15 the optimal. We report the rankings for each configuration attribute in Appendix B. Before completing the tasks, each participant watched a 5-minutes video explaining

[4]https://github.com/srcnalt/OpenAI-Unity/tree/master
[5]https://lmstudio.ai/

Muresan et al.'s design space. After watching the video, the experts defined their feedforward configuration for the three tasks. After that, we collected open-ended feedback on the activity.

The feedforward configurations proposed by participants show consistently high alignment with our rankings, with mean scores respectively of 11.85 ($SD = 1.08$), 12.33 ($SD = 1.03$), and 13.17 ($SD = 0.93$). Score variability was low across all tasks, indicating that all experts were successful in browsing, understanding and effectively using all the parts of the feedforward spectrum, even if it was the first time they were introduced to both the design space and the proposed interface to author it.

The open-ended feedback provided us with many minor suggestions to improve the presentation, but the overall interface organisation was assessed as usable and understandable. We changed the support for boolean flags (e.g., the visibility, duplication and ghosting for the avatar, actions and outcomes) from radio buttons to toggle buttons. In addition we modified the description of the avatar types in order to clarify what does it means to show a full avatar (a complete human representation) or partial (showing only the body parts involved in the interaction, e.g., the hands). We used this feedback for continuing the implementation work towards the final (and working) version of the authoring UI.

Table 1. Summary of the implemented features of Muresan et al's work in BeatriXR. Italics descriptions justify unimplemented elements.

		Muresan et al.	BeatriXR Implementation
Triggering	**Trigger Type**	**Persistent Triggers**	The animation will loop constantly, as soon as it finishes it it will start again right away.
		Location	When the trainee gets nearer the target object than a predefined distance, the animation will start.
		Events	We have selected a "timer" event, where every 5 seconds of the animation not playing, it will automatically appear.
		Gaze	The animation will start as soon as the trainee looks at the area where the target object is located.
		Action	The trainee can manually trigger the animation by pressing the "Preview" or "Compare" buttons in the left panel of Figure 5.
	Signifiers		A red cone will show over the target object, to identify the presence of feedforward associated to the object.
Previewing	**Level of Detail**	**All Actions**	The animation will queue all the animations from the task the trainee is learning through the environment and play them one after the other.
		A subset of actions	We have split this in two separate functions: one shows a selection of animations that the procedure creator flagged as important, then it shows them in a similar way to "All Actions"; the other function only shows the feedforward for the current interaction the trainee is tasked to do for moving in the next step of the task.
	Targets	**Avatar**	Either a full-body humanoid or a partial representation of it showing only the hands can be used for the feedforward representation.
		Actions	We have implemented this feature by creating an arrow with a panel attached to it (that shows the action the trainee needs to do), that represents in an abstract way how to interact with the object.
		Outcomes	The outcome can be either be the original object the trainee needs to interact with for completing the step or a replica of it, and they can be manipulated by the other targets during the feedforward preview.
	Avatar Type	**Full Avatar**	This feature has been implemented as a full humanoid character demonstrating what the trainee should perform in the virtual environment.
		Partial Avatar	This option shows only the hands of the avatar above.
		Real Device	In our implementation this feature overlapped with Real User, as we have focused our proof-of-concept to only hand tracking at first.
		Real User	This feature is similar to "Full Avatar", but the humanoid character performs the movements the trainee should do in real world (useful if the mapping between the tracking support and the virtual representation of the user is not direct).
		None	The avatar can be completely hidden from the view of the trainee.
	Perspective	**Third-Person**	The representation can be seen moving away from the trainee, and then it starts previewing away from the position of them, so they can see it from a different view.
		First-Person	The system will move the trainee alongside the avatar, so they can see the animation as if they were the avatar doing the animation themselves.
	Modifier	**Audio**	*We did not implemented audio features, since we have focused on visuals rather than audio cues.*
		Haptic	*Since we have focused on hand tracking, and the HMDs are not equipped with haptic feedback, we have not implemented this feature.*
		Transform	We have implemented this feature as an option that, if enabled, will move any of the targets slightly away from their original position, mainly for clear the view for the trainee, in the case they would like to follow a guide while interacting with the original target object.
		None	The transform modifier can be kept disabled.
	Rendering	**Ghosted**	The avatar and outcome targets can be rendered semi-transparent, but visible enough for the trainee to understand what the feedforward is showing.
		Original	The avatar and outcome targets can be left opaque and fully visible by the trainee.
	Representation	**Direct**	The feedforward representation shows visually what the user should do in order to complete the steps of the task.
		Indirect	The UI panels shown in Figure 3 display in a text form and with icons the instructions the trainee is tasked to follow in order to complete the current step and the next ones.
	Duplication	**Yes**	The avatar and outcome targets can be duplicated, so for each enabled option, the original ones (the hands of the trainee and the target object) can be left at their original positions.
		No	The system will use, for each disabled option, the hands of the trainee or the original target object for showing how to perform the steps of the task.
	Playback		Through the left panel of Figure 5 the trainee can manually pause, resume, stop, rewind or fast forward the feedforward animation
Exiting	**Untrigger**		The trainee, depending on the selected option, can either look away, move away from the target object or wait for its natural end to stop the feedforward preview.
	Exit Transition		The transition for dismissing the feedforward preview and going back to the original state can be either instantaneous (removing everything right away), by rewinding the preview back to the start and then removing it, or with a fade in/fade out animation that temporarily occludes the view of the trainee while the preview is being removed.

4.3 Exploratory Evaluation with Experts

In a second evaluation session with experts, we asked for feedback on the final implementation of the UI for structuring and configuring the feedforward in VR. In addition, we explored the idea of using LLMs as feedforward advisors by looking into the feasibility of their use and their alignment with the reasoning of the experts.

4.3.1 Participants and Procedure. We invited 11 experts (9 male, 2 female, age range 23-43) with either prior academia research or current or past full-time jobs in XR development. Among the participants, 7 of them had up to 2 years of experience in XR development, 2 of them had 3-4 years, and 2 of them had 5 or more years. The session consisted of the following parts.

Tutorial. The experts were introduced to the topic of feedforward by providing them with a document that they could browse during the whole session. It contained descriptions for all the possible options available in the system, and a quick explanation of the supported interactions in the system.

UI Evaluation. In this part, the experts were asked to evaluate the UI seen in Figure 4. Participants were presented with the same three scenarios used for the preliminary feedback, namely T1) clicking on a virtual mouse with touch interaction, T2) grabbing a beaker through laser pointing and T3) turning a sink with a grab technique. For each one, we gave participants a screenshot of the feedforward settings UI, which was set up to illustrate a specific feedforward configuration, together with different screenshots of the considered VR interaction and environment.

We assessed the participants' understanding of the UI meaning and functionality by asking them to answer a series of true/false statements. At the end of this part, participants rated their understanding of the UI on a subjective scale from 1 to 7 using a Likert scale: "Is the settings UI in the image comprehensible to you?" with "Totally incomprehensible" at the lowest and "Totally comprehensible" at the highest. Finally, open-ended questions explored potential understanding issues and described what the feedforward representation would show the user in the environment.

Assessing LLMs Suggestions. In the final part, we collected the expert's opinion about the design suggestions provided four LLMs: GPT OSS 120B, Mistral Small 3.2, Deepseek R1, GPT 5 Mini. We selected these models as the best performing candidates among a larger set of available LLMs, we report the selection procedure in Appendix C.

We asked the experts to consider again the three tasks proposed in the previous part, each one having a default configuration for the feedforward. For each task, experts reviewed four LLM answers (randomized to prevent model recognition) and suggested improvements to the default feedforward options. They then rated each answer on a 1–7 Likert scale for quality, usefulness of the suggested configuration, and overall feedforward quality. Finally, experts provided open-ended comments on each answer.

4.4 Feedback about the Feedforward Configuration User Interface

The data collected about the UI evaluation shows an overall positive rating of the feedforward options visual representation. Participants positively answered 85% of the questions, with each task scoring 76%, 91% and 88% respectively. This shows that the UI, despite being presented for the first time, achieves a satisfactory score indicating general understanding. Furthermore, the increase in percentages after the first task suggests that after an initial adjustment period participants became accustomed to the system and comprehended its functionality better. This is also supported by a similar trend on the subjective comprehensibility score, which was 5.82 (STD: 0.60), 6.36 (STD: 0.67) and 6.27 (STD: 0.79) out of 7 respectively.

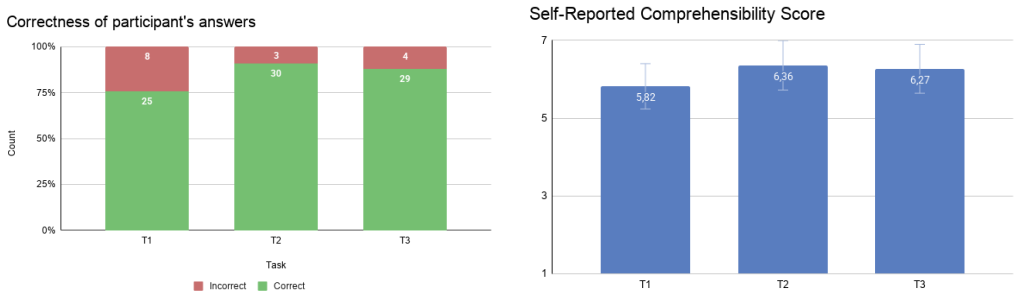


Fig. 11. Results from the UI evaluation, from left to right: 100% stacked bar chart showing the correctness of answers from the participants in the comprehension part of the tasks; self-reported comprehensibility scores from all the tasks.

We complement the reporting on the experts' scores with a qualitative analysis their open-ended feedback. Participants highlighted several significant strengths regarding the design of the feedforward authoring panel. Participants consistently reported that the UI was understandable and well designed. For instance, P6 stated: "I think the interface is quite clear and provides the user with all the tools they need. [...]", while P7: "I found the UI much cleaner and easier to understand.".

This was largely attributed to the logical subdivision of the workspace; by clustering options into meaningful phases (specifically the Trigger, Preview, and Exit panels) the system established a coherent structural hierarchy that guided them in understanding the different components of the feedforward model by Muresan et al. [27]. P1 commented "[...] The options are clustered in a meaningful way", while P5: "I appreciated the subdivision of the UI in phases", and P6: " [...] I appreciated the division of the three panels: trigger, preview and exit".

Despite the high information density required for the task, the interface successfully balanced comprehensiveness with spatial efficiency, providing all necessary tools and options that can be activated within a constrained footprint. P2 appreciated the space usage:"It summarizes in a short space all the elements necessary to understand how the simulation takes place". P3 commented "The UI is good. It definitely displays a lot of information in a small space.", and P10: "The UI is clean, functional and have clear goals.". The same visualization, however, also revealed several recurring usability challenges. The primary concern cited by participants involved iconographic ambiguity ($n = 7$), followed closely by issues regarding information density and interface clutter ($n = 6$). Users also noted that specific UI terminology lacked clarity ($n = 4$), which contributed to difficulties in navigating system settings ($n = 2$) and interpreting the current state of the interface ($n = 2$). A minor but notable issue was the cognitive load required to understand specific value combinations, such as the relationship between perspective views and avatar configurations ($n = 1$). To address these limitations, participants suggested several improvements, including the integration of visual previews for settings ($n = 3$) and a comprehensive overhaul of icons and labels ($n = 2$ each) to reduce semantic distance. Furthermore, a participant (P8) proposed a more modular layout to balance element distribution and suggested a progressive disclosure strategy to mitigate the learning curve, i.e., showing the panels separately at first and then to expose the user with all the three panels together.

Finally, technical enhancements such as dynamic setting visibility (where inapplicable options are hidden) and the implementation of "Quick Settings" presets were identified as high-value improvements for optimizing workflow efficiency. The set of options to be included in such a panel was different across the tasks, making them difficult to identify.

4.5 Feedback on LLMs' suggestions

Table 2 presents the final scores obtained by the LLMs during the evaluation process. The data indicates that all LLMs are capable of generating plausible configuration suggestions with a significant degree of reasonability and a satisfactory resulting feedforward representation. Additionally, the scores demonstrate a partial alignment between the responses of the LLMs and expert reasoning. We provide a more insightful interpretation of these results through the qualitative feedback discussed in the next section.

Table 2. Results of LLM performance across three criteria and three tasks (T1, T2, T3). Metrics are presented as: Average (Avg) / Standard Deviation (SD) / Median (Md). All scores are out of 7.

Criterion	Model	T1 (Avg / SD / Md)	T2 (Avg / SD / Md)	T3 (Avg / SD / Md)
Answer Quality	Deepseek R1	5.64/1.75/6	5.45/1.04/6	5.91/1.04/6
	GPT OSS 120B	5.64/1.12/6	4.55/1.37/5	5.82/1.17/6
	GPT 5 Mini	5.45/1.81/6	4.91/1.64/5	5.36/1.36/5
	Mistral Small 3.2	4.91/1.64/4	4.45/2.02/5	5.55/0.82/6
Realism or Reasonability	Deepseek R1	5.73/1.85/6	5.64/1.21/6	5.64/1.12/6
	GPT OSS 120B	5.82/1.47/6	5.18/1.33/5	6.09/0.83/6
	GPT 5 Mini	5.64/1.86/6	4.73/1.56/5	5.55/1.29/6
	Mistral Small 3.2	5.00/1.41/5	5.00/1.73/6	5.73/0.90/6
Representation Quality	Deepseek R1	5.55/1.75/6	5.09/1.22/5	5.27/1.10/5
	GPT OSS 120B	5.64/1.57/6	4.64/1.57/5	5.82/1.40/6
	GPT 5 Mini	5.18/1.99/6	4.64/1.57/5	5.18/1.54/5
	Mistral Small 3.2	5.18/1.72/6	4.73/1.79/5	5.36/0.92/5

Experts generally found LLM suggestions coherent and helpful, with shorter responses being clearest. Most models accurately assessed context configurations and feedforward options, offering concise, well-argued insights from various VR user perspectives.

On the negative side, some experts expressed general concerns in the post-test questions about the quality and structure of some answers. Some experts stated that the answers were sometimes messy (P1) or cluttered (P7), retained system prompt data, and contained repetitive, copy-pasted sentences (P7). Sometimes, the reasoning behind choices seemed unfounded (P3) or lacked clear justification for reconfigurations (P9). Finally, some suggestions were considered either irrelevant (P6) or, at times, disconnected in the same answer (P11).

By exploring the comments for each answer, we identified patterns that affected the evaluation of the suggestions. Experts avoided feedforward configurations that overrode user control in the virtual environment. For example, P2 objected to a first-person view in T2 from GPT 5 Mini, stating, "I'd feel sick because it moves without my control".

Participants preferred "partial avatars" for visibility and minimal occlusion, rejecting full-body avatars as cluttered and unhelpful for orientation. Manual triggers were favored over gaze-based ones, and "rewinding" animations were seen as less intuitive than instant or fade transitions.

Deepseek R1 was praised for clear, structured output and strong context awareness. In contrast, GPT OSS 120B was criticized for an "imperative" tone, Mistral Small 3.2 for being overly conservative, and GPT 5 Mini for introducing an unclear "weight system" without explanation or justification.

Participants consistently rejected certain LLM choices, notably the Fade In/Out Transition and problematic perspective settings. Negative feedback targeted GPT OSS 120B's third-person setup

with a non-duplicated avatar, while Deepseek R1 and GPT 5 Mini were criticized for their use of the first-person perspective.

5 Discussion

5.1 Advantages of a Toolkit for VR Feedforward Development

BeatriXR addresses a key practical challenge in VR development: implementing feedforward is complex, repetitive, and often tightly coupled with application-specific logic. Developers must handle multiple interdependent aspects, including spatial placement, timing of previews, embodiment through avatars, and transitions between interaction states. In the absence of dedicated support, these elements are typically implemented ad hoc, limiting reuse and increasing development effort.

The results of our expert evaluation suggests that BeatriXR provides a viable approach for structuring and implementing feedforward in VR. By operationalising the feedforward design space into modular components, BeatriXR enables a more structured development approach. The separation into Trigger, Preview, and Exit phases provides a clear and validated conceptual model, while the toolkit exposes these dimensions as configurable parameters that can be reused across applications. This allows developers to compose feedforward behaviours without re-implementing them from scratch, reducing both development time and technical overhead.

More importantly, the toolkit supports systematic exploration of alternative designs. Instead of committing early to a single implementation, developers can iterate over different configurations, compare them, and refine their choices. This shifts feedforward design from a one-off engineering task to an iterative and exploratory process, which is particularly valuable in VR contexts where interaction techniques are still evolving.

5.2 Advantages for Empirical Studies and Research

Beyond development, BeatriXR has implications for empirical research on VR interaction. Studying feedforward techniques in immersive environments is often challenging because implementing multiple experimental conditions requires significant effort. As a result, prior work tends to focus on isolated techniques or limited subsets of the design space.

During the evaluation, experts engaged with multiple configurations of feedforward and assessed their suitability for different interaction contexts. This ability to quickly explore and compare alternatives highlights how the toolkit can support experimental workflows. By providing a unified framework that covers a wide range of feedforward configurations, BeatriXR enables researchers to more easily instantiate and compare different conditions within controlled studies. The ability to switch between configurations without modifying the underlying implementation lowers the barrier to conducting systematic evaluations, including factorial designs that explore combinations of triggering mechanisms, preview types, and exit strategies.

In this sense, the toolkit can act as an experimental platform for investigating feedforward in VR. It supports reproducibility by standardising how configurations are defined and applied, and it facilitates rapid prototyping of study conditions. This can accelerate research progress by allowing more comprehensive exploration of the design space than would otherwise be feasible.

5.3 Usefulness of the Interface for Experts and End Users

The interface of BeatriXR is designed to support both domain experts (at design time) and end users (at runtime), reflecting feedforward's dual nature as both a design artefact and an interactive feature.

For domain experts, the interface provides direct access to the feedforward design space's parameters. Rather than encoding decisions in code, experts can configure and adjust feedforward

through dedicated UI panels. This lowers the technical barrier to authoring VR interactions and enables experts to focus on higher-level design considerations, such as task structure and user experience. The organisation of settings into Trigger, Preview, and Exit categories also supports a more systematic reasoning about design choices.

For end users, the interface enables interaction with feedforward during task execution. Users can trigger demonstrations, adjust settings, and compare alternative interaction techniques. The avatar-based previews provide an intuitive and embodied representation of actions, supporting learning through observation and imitation. Additionally, the possibility to modify feedforward parameters allows users to adapt the guidance to their preferences and level of expertise.

The expert evaluation suggests that the interface is generally perceived as usable and effective for exploring feedforward configurations. At the same time, the richness of the configuration space introduces a trade-off between flexibility and complexity. While experienced users may benefit from fine-grained control, less experienced users may require additional guidance, such as presets or simplified modes.

5.4 Usefulness of LLM-Based Assistance

The integration of a Large Language Model in BeatriXR aims to support the exploration of the feedforward design space rather than to automate decision-making. Given the large number of possible configurations and the context-dependent nature of feedforward design, identifying suitable settings can be challenging, especially for non-expert users.

In this context, the LLM acts as a mixed-initiative assistant, suggesting alternative configurations based on the current task and setup. It provides a flexible interface to the design space, allowing users to obtain recommendations without explicitly encoding decision rules. This can be particularly useful during early stages of design, where rapid exploration of alternatives is more important than optimisation.

The expert feedback indicates that such assistance can be valuable at design time, helping users consider options they might not have explored otherwise. However, the results also highlight that the LLM should not be interpreted as a replacement for structured approaches. In well-defined scenarios, rule-based or model-based systems may offer greater predictability and control.

Overall, the usefulness of the LLM lies in supporting exploration and ideation, rather than in providing optimal or authoritative solutions. Its role is therefore complementary to the toolkit, augmenting the design process without replacing human judgment.

5.5 Limitations

This work has several limitations that should be acknowledged.

First, the evaluation focuses on expert feedback and does not include task-based measures of performance. As a result, we cannot yet quantify the impact of BeatriXR on development efficiency, usability, or end-user learning outcomes. Future work should include controlled studies comparing the toolkit with baseline approaches, as well as evaluations involving end users in realistic scenarios.

Second, the role of the LLM remains exploratory. While the results suggest potential benefits for design-time assistance, we do not provide a systematic comparison with alternative approaches, such as rule-based systems. The use of LLMs for runtime adaptation is also not sufficiently validated and should be considered preliminary.

Third, BeatriXR currently focuses on visual feedforward and does not include other modalities such as haptic or auditory feedback. Extending the toolkit to support multimodal guidance would provide a more complete coverage of the feedforward design space.

Finally, while the richness of the configuration space is a strength, it may also introduce complexity for users. Additional support mechanisms, such as presets, templates, or adaptive interfaces, may be needed to make the system more accessible to a broader range of users.

6 Conclusions

We have presented BeatriXR, a toolkit for integrating feedforward information in any VR environment. The system takes advantage of previous research to implement a comprehensive tool for showing how to perform actions in a virtual environment through the use of avatars. By operationalizing an existing feedforward design space into reusable components, BeatriXR supports the structured authoring and configuration of feedforward across different interaction scenarios. BeatriXR also exploits recent advancements in Artificial Intelligence to support domain experts during the configuration process, suggesting plausible alternatives for feedforward visualizations, while leaving final design decisions under user control, serving as an assistive tool to help explore the space of possible configurations.

Exploratory data gathered during the expert reviews for BeatriXR provides initial insights into the usability of the toolkit and the usefulness of its configuration interface, as well as how LLM-generated suggestions can support the exploration of alternative design possibilities, particularly for novice domain experts. These findings highlight the potential of BeatriXR both as a practical development tool and as a platform for investigating feedforward in VR. Future work can build upon our first steps by strengthening the empirical evaluation and further exploring AI-assisted support for feedforward design, advancing research in this field.

Acknowledgments

Lucio Davide Spano acknowledges "GraphNet: models, computation and estimation in networks and graph analysis" funded by D.M. 737/2021 - Linea d'intervento Iniziative di ricerca interdisciplinare su temi di rilievo trasversale per il PNR (grant number F25F21002720001). Kris Luyten is supported by the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program, R-13509.

References

[1] 2025. IBM watsonx as a Service. https://www.ibm.com/docs/en/watsonx/saas?topic=prompts-model-parameters-prompting

[2] Valentino Artizzu, Kris Luyten, Gustavo Rovelo Ruiz, and Lucio Davide Spano. 2024. ViRgilites: Multilevel Feedforward for Multimodal Interaction in VR. *Proc. ACM Hum.-Comput. Interact.* 8, EICS, Article 247 (jun 2024), 24 pages. doi:10.1145/3658645

[3] Jeremy Bailenson, Kayur Patel, Alexia Nielsen, Ruzena Bajscy, Sang-Hack Jung, and Gregorij Kurillo. 2008. The Effect of Interactivity on Learning Physical Actions in Virtual Reality. *Media Psychology* 11, 3 (2008), 354–376. arXiv:https://doi.org/10.1080/15213260802285214 doi:10.1080/15213260802285214

[4] Marc Baloup, Thomas Pietrzak, and Géry Casiez. 2019. RayCursor: A 3D Pointing Facilitation Technique Based on Raycasting. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3290605.3300331

[5] Soumya C. Barathi, Daniel J. Finnegan, Matthew Farrow, Alexander Whaley, Pippa Heath, Jude Buckley, Peter W. Dowrick, Burkhard C. Wuensche, James L. J. Bilzon, Eamonn O'Neill, and Christof Lutteroth. 2018. Interactive Feedforward for Improving Performance and Maintaining Intrinsic Motivation in VR Exergaming. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3173574.3173982

[6] Ligia Batrinca, Giota Stratou, Ari Shapiro, Louis-Philippe Morency, and Stefan Scherer. 2013. Cicero - Towards a Multimodal Virtual Audience Platform for Public Speaking Training. In *Intelligent Virtual Agents*, Ruth Aylett, Brigitte Krenn, Catherine Pelachaud, and Hiroshi Shimodaira (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 116–128.

[7] Olivier Bau and Wendy E. Mackay. 2008. OctoPocus: A Dynamic Guide for Learning Gesture-Based Command Sets. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (Monterey, CA, USA) *(UIST*

'08). Association for Computing Machinery, New York, NY, USA, 37–46. doi:10.1145/1449715.1449724

[8] Jeremy Boy, Louis Eveillard, Françoise Detienne, and Jean-Daniel Fekete. 2015. Suggested interactivity: Seeking perceived affordances for information visualization. *IEEE transactions on visualization and computer graphics* 22, 1 (2015), 639–648.

[9] Philip Brookins and Jason Matthew DeBacker. 2023. Playing Games With GPT: What Can We Learn About a Large Language Model From Canonical Strategic Games? *SSRN Electronic Journal* (2023). doi:10.2139/ssrn.4493398

[10] Jay L. Caulfield and Felissa K. Lee. 2022. Digital Simulation: Applying Critical Thinking to the Practice of Ethical Decision Making. *Journal of Business Ethics Education* 19 (July 2022), 35–66. doi:10.5840/jbee2022193

[11] Edwige Chauvergne, Martin Hachet, and Arnaud Prouzeau. 2023. User Onboarding in Virtual Reality: An Investigation of Current Practices. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 711, 15 pages. doi:10.1145/3544548.3581211

[12] Minze Chen, Zhenxiang Tao, Weitong Tang, Tingxin Qin, Rui Yang, and Chunli Zhu. 2024. Enhancing emergency decision-making with knowledge graphs and large language models. *International Journal of Disaster Risk Reduction* 113 (2024), 104804. doi:10.1016/j.ijdrr.2024.104804

[13] Yiting Chen, Tracy Xiao Liu, You Shan, and Songfa Zhong. 2023. The emergence of economic rationality of GPT. *Proceedings of the National Academy of Sciences* 120, 51 (2023), e2316205120. doi:10.1073/pnas.2316205120

[14] Sven Coppers, Kris Luyten, Davy Vanacken, David Navarre, Philippe Palanque, and Christine Gris. 2019. Fortunettes: Feedforward about the Future State of GUI Widgets. 3, EICS, Article 20 (jun 2019), 20 pages. doi:10.1145/3331162

[15] Tom Djajadiningrat, Kees Overbeeke, and Stephan Wensveen. 2002. But how, Donald, tell us how? on the creation of meaning in interaction design through feedforward and inherent feedback. In *Proceedings of the 4th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (London, England) *(DIS '02)*. Association for Computing Machinery, New York, NY, USA, 285–291. doi:10.1145/778712.778752

[16] Katherine Fennedy, Jeremy Hartmann, Quentin Roy, Simon Tangi Perrault, and Daniel Vogel. 2021. Octopocus in VR: using a dynamic guide for 3D mid-air gestures in virtual reality. *IEEE Transactions on Visualization and Computer Graphics* 27, 12 (2021), 4425–4438.

[17] Maximilian C. Fink, Seth A. Robinson, and Bernhard Ertl. 2024. AI-based avatars are changing the way we learn and teach: benefits and challenges. *Frontiers in Education* 9 (2024). doi:10.3389/feduc.2024.1416307

[18] Tovi Grossman and George Fitzmaurice. 2010. ToolClips: An Investigation of Contextual Video Assistance for Functionality Understanding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 1515–1524. doi:10.1145/1753326.1753552

[19] Ryan Guthridge and Virginia Clinton-Lisell. 2023. Evaluating the Efficacy of Virtual Reality (VR) Training Devices for Pilot Training. *Journal of Aviation Technology and Engineering* 12, 2 (July 2023). doi:10.7771/2159-6670.1286

[20] Malte Issleib, Alina Kromer, Hans O Pinnschmidt, Christoph Süss-Havemann, and Jens C Kubitz. 2021. Virtual reality as a teaching method for resuscitation training in undergraduate first year medical students: a randomized controlled trial. *Scandinavian Journal of Trauma, Resuscitation and Emergency Medicine* 29, 1 (Feb. 2021), 27.

[21] Nicholas Jennings, Ananya Nandy, Xinyi Zhu, Yuting Wang, Fanping Sui, James Smith, and Bjoern Hartmann. 2022. GeneratiVR: Spatial Interactions in Virtual Reality to Explore Generative Design Spaces. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 366, 6 pages. doi:10.1145/3491101.3519616

[22] James C. Lester, Sharolyn A. Converse, Susan E. Kahler, S. Todd Barlow, Brian A. Stone, and Ravinder S. Bhogal. 1997. The persona effect: affective impact of animated pedagogical agents. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '97)*. Association for Computing Machinery, New York, NY, USA, 359–366. doi:10.1145/258549.258797

[23] Klemen Lilija, Søren Kyllingsbæk, and Kasper Hornbæk. 2021. Correction of Avatar Hand Movements Supports Learning of a Motor Skill. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. 1–8. doi:10.1109/VR50410.2021.00069

[24] Nunzio Lorè and Babak Heydari. 2024. Strategic behavior of large language models and the role of game structure versus contextual framing. *Scientific Reports* 14, 1 (09 Aug 2024), 18490. doi:10.1038/s41598-024-69032-z

[25] Ntima Mabanza and Lizette de Wet. 2014. *Determining the Usability Effect of Pedagogical Interface Agents on Adult Computer Literacy Training*. Springer Berlin Heidelberg, Berlin, Heidelberg, 145–183. doi:10.1007/978-3-642-41965-2_6

[26] Microsoft. 2024. Mixed Reality Toolkit Unity Plugin. https://github.com/MixedRealityToolkit/MixedRealityToolkit-Unity [Online; accessed 05-September-2024].

[27] Andreea Muresan, Jess McIntosh, and Kasper Hornbæk. 2023. Using Feedforward to Reveal Interaction Possibilities in Virtual Reality. *ACM Trans. Comput.-Hum. Interact.* (jun 2023). doi:10.1145/3603623 Just Accepted.

[28] Donald A Norman. 1999. Affordance, conventions, and design. *interactions* 6, 3 (1999), 38–43.

[29] Donald A Norman. 2008. The way I see IT signifiers, not affordances. *interactions* 15, 6 (2008), 18–19.

[30] Gun Woo (Warren) Park, Anthony Tang, and Fanny Chevalier. 2024. JollyGesture: Exploring Dual-Purpose Gestures and Gesture Guidance in VR Presentations. In *Proceedings of the 50th Graphics Interface Conference* (Halifax, NS, Canada) *(GI '24)*. Association for Computing Machinery, New York, NY, USA, Article 28, 14 pages. doi:10.1145/3670947.3670965

[31] Sandra Poeschl and Nicola Doering. 2012. Designing Virtual Audiences for Fear of Public Speaking Training – An Observation Study on Realistic Nonverbal Behavior. In *Annual Review of Cybertherapy and Telemedicine 2012*. IOS Press, 218–222. doi:10.3233/978-1-61499-121-2-218

[32] Amotz Perlman Rafael Sacks and Ronen Barak. 2013. Construction safety training using immersive virtual reality. *Construction Management and Economics* 31, 9 (2013), 1005–1017. doi:10.1080/01446193.2013.828844

[33] Elena Sblendorio, Vincenzo Dentamaro, Alessio Lo Cascio, Francesco Germini, Michela Piredda, and Giancarlo Cicolini. 2024. Integrating human expertise & automated methods for a dynamic and multi-parametric evaluation of large language models' feasibility in clinical decision-making. *International Journal of Medical Informatics* 188 (2024), 105501. doi:10.1016/j.ijmedinf.2024.105501

[34] Rajinder Sodhi, Hrvoje Benko, and Andrew Wilson. 2012. LightGuide: Projected Visualizations for Hand Movement Guidance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 179–188. doi:10.1145/2207676.2207702

[35] James W. A. Strachan, Dalila Albergo, Giulia Borghini, Oriana Pansardi, Eugenio Scaliti, Saurabh Gupta, Krati Saxena, Alessandro Rufo, Stefano Panzeri, Guido Manzi, Michael S. A. Graziano, and Cristina Becchio. 2024. Testing theory of mind in large language models and humans. *Nature Human Behaviour* 8, 7 (01 Jul 2024), 1285–1295. doi:10.1038/s41562-024-01882-z

[36] Michael Terry and Elizabeth D. Mynatt. 2002. Side Views: Persistent, on-Demand Previews for Open-Ended Tasks. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology* (Paris, France) *(UIST '02)*. Association for Computing Machinery, New York, NY, USA, 71–80. doi:10.1145/571985.571996

[37] Hasan Mahbub Tusher, Steven Mallam, and Salman Nazir. 2024. A Systematic Review of Virtual Reality Features for Skill Training. *Technology, Knowledge and Learning* 29, 2 (June 2024), 843–878.

[38] Jo Vermeulen, Kris Luyten, Elise van den Hoven, and Karin Coninx. 2013. Crossing the Bridge over Norman's Gulf of Execution: Revealing Feedforward's True Identity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 1931–1940. doi:10.1145/2470654.2466255

[39] S. A. G. Wensveen, J. P. Djajadiningrat, and C. J. Overbeeke. 2004. Interaction Frogger: A Design Framework to Couple Action and Function through Feedback and Feedforward. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (Cambridge, MA, USA) *(DIS '04)*. Association for Computing Machinery, New York, NY, USA, 177–184. doi:10.1145/1013115.1013140

[40] Xingyao Yu, Benjamin Lee, and Michael Sedlmair. 2024. Design Space of Visual Feedforward And Corrective Feedback in XR-Based Motion Guidance Systems. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 723, 15 pages. doi:10.1145/3613904.3642143

[41] Xingyao Yu, Benjamin Lee, and Michael Sedlmair. 2024. Design Space of Visual Feedforward And Corrective Feedback in XR-Based Motion Guidance Systems. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 723, 15 pages. doi:10.1145/3613904.3642143

A Support for Guidelines by Muresan et al. in the literature

Table 3. Comparison between the guidelines of Muresan et al., ViRgilites, JollyGesture, and the design guidelines of Yu et al.

		Muresan et al.	ViRgilites	JollyGesture	Yu et al.	GeneratiVR
Triggering	**Trigger Type**	**Persistent Triggers**				✓
		Location				
		Events		✓		
		Gaze				
		Action				
	Signifiers					✓
Previewing	**Level of Detail**	**All Actions**			✓	
		A subset of actions	✓		✓	
	Targets	**Avatar**			✓	
		Actions	✓	✓	✓	
		Outcomes				✓
	Avatar Type	**Full Avatar**			✓	✓
		Partial Avatar				
		Real Device				
		Real User				
		None	✓	✓		
	Perspective	**Third-Person**			✓	
		First-Person	✓	✓	✓	✓
	Modifier	**Audio**				
		Haptic				
		Transform				
		None	✓			
	Rendering	**Ghosted**				✓
		Original	✓			
	Representation	**Direct**			✓	✓
		Indirect	✓			
	Duplication	**Yes**			✓	✓
		No	✓			
	Playback					
Exiting	**Untrigger**					
	Exit Transition					

B Feedforward Options Ratings in the Intermediate Evaluation

The following are the rating we used to assess the feedforward configuration proposed by experts in the intermediate evaluation. The table reports a column for each task.

Feedforward	T1	T2	T3
Gaze	0.5	0.7	0.6
Position	0.6	0.6	0.6
Event	0.4	0.4	0.4
Action	1	1	1
Persistent	0.2	0.4	0.5
Signifiers	1	1	1
No Signifiers	1	0.8	1
Full Animation	1	1	1
Parts of Animation	1	0.9	1
Action Visible	1	1	1
Action Invisible	1	1	1
Outcome Visible	1	1	1
Outcome Invisible	0	0	0
Avatar Visible	1	1	1
Avatar Invisible	0	0	0
Full Avatar	1	1	1
Parts of Avatar	0.6	0.6	0.6
Real User	1	1	0.9
First Person	0.8	0.9	0.9
Third Person	1	1	1
Dislocated	0	0	0.8
Not Dislocated	1	1	1
Transparent Avatar	1	0.8	1
Opaque Avatar	0.9	1	0.8
Transparent Object	0.8	0.5	0.7
Opaque Object	1	1	1
Duplicated Avatar	1	1	1
Singular Avatar	0.3	0.6	0.9
Duplicated Object	1	1	0.5
Singular Object	0.9	0.9	1
Gaze Away	0.4	0.7	0.6
Move Away	0.3	0.6	0.6
Automatic	1	1	1
Rewind	0.2	0.8	0.1
Instant	1	1	1
Fade In	0.8	0.5	0.2

Table 4. Ratings of all the possible feedforward options in the intermediate evaluation tasks.

C LLM model selection

The quality of suggestions from the AI component depends on the underlying LLM, but it was not feasible for our participant to assess the output of all the LLMs available when we prepared

Table 5. Performances of the considered LLMs in the preselection phase.

Vendor	Model	Param.	Quant.	Answer Completeness	Answer Quality (avg.)	Answer Plausibility (avg.)	Total (sum out of 10)	Selected?
Llama	3.1 Instruct	8B		✗	-	-	-	-
	3.3 Instruct	70B		✗	-	-	-	-
	4 Maverick	17B	FP8	✗	-	-	-	-
OpenAI	GPT OSS	120B		✓	5	4	9	✓
	GPT 5			✓	3	5	8	
	GPT 5 Mini			✓	5	5	10	✓
Alibaba	Qwen3	32B		✓	5	3	8	
		235B		✓	3	3	6	
Deepseek	R1			✓	5	4	9	✓
Google	Gemini 2.5 Flash			✓	5	1	6	
NVIDIA	Nemotron Nano	9B		✓	5	2	7	
Mistral	Small 3.2	24B		✓	5	4	9	✓

the evaluation, which is currently increasing. Therefore, before gathering expert feedback, we identified representative models for generating suggestions. We started from a set of popular models recognized in both research and public discourse. This selection, while not exhaustive, includes models with significant attention due to widespread adoption, performance, and community interest. We considered 12 models available between Abacus.ai and OpenRouter, created by different companies, considering various parameters and quantization methods (when available, the models have been chosen with a "Instruct" dataset fine-tune[6]). We report the list of models and the parameters/quantisation: LLama 4 Maverick Instruct 17B FP8, LLama 3.1 Instruct 8B, LLama 3.3 Instruct 70B, Qwen3 32B, Qwen3 235B, Deepseek R1, Gemini 2.5 Flash, GPT OSS 120B, GPT 5 Mini, GPT 5, NVIDIA Nemotron Nano 9B, Mistral Small 3.2 24B.

All the models have been setup using the following settings: i) temperature set to 0, ii) top-k sampling set to 1, iii) top-p sampling set to 0. These settings aim to minimize randomness in responses and ensure determinism in the tests [1]. We did this because hallucinations are intrinsic in every model up to these days [7]. The next section details the selection process through a technical evaluation. We used the top-performing models for expert evaluation.

Test Environment All the LLMs have been prepared to accept the system prompt and the user prompt in Appendix D.

The resulting answers from each LLM have been analysed by the authors who agreed on rating them using the following criteria:

- **Completeness**: Whether the LLM model returns an answer that discussed the suggested setting for each parameter involved in the submitted request (binary);
- **Quality**: How the suggestions together may match in an actual feedforward implementation (1 to 5 scale);
- **Plausibility**: taken the LLM answer as valid, how coherent is the justification of the answer (1 to 5 scale).

If the model returned no answer on one or more parameters, then the model was considered excluded from subsequent scoring calculations. The scoring of the remaining models was determined by averaging the scores of *Answer Quality* and *Answer Plausibility* across two answers for each queried model. To proceed with the expert evaluation we have selected the top 4 models, which are: GPT OSS 120B, GPT 5 Mini, Deepseek R1 and Mistral Small 3.2 24B. Table 5 shows the performances of the considered LLMs in the pre-selection phase.

[6] Reinforcement Learning from Human Feedback (RLHF) - Wikipedia, last accessed: 20/01/2025
[7] OpenAI - Why Language Models Hallucinate - Last accessed 04/12/2025

D System and User Prompt for the LLMs

D.1 System Prompt

You are tasked with suggesting zero or more alternative new options for an animation representation in a virtual reality environment. You should only analyze these options as a controller for the animation, as the user is not influenced by the change of any of these options, the user will only see the animation set up with the settings in this prompt.

Instructions:

1. **Read** the descriptions of the environment settings below. 2. **Note** the weight of each item in parentheses (1 = negligible importance, 5 = extremely important). 3. **Evaluate** the significance of each setting based on its weight and description. 4. **Understand** what the variables mean, so you don't misinterpret them when the user prompt is issued. 5. **Read** the prompt of the user, which will contain a **description of the step** and the **set of default options** for the representation. 5. **Decide** on possible new settings that could be better - you can also suggest no changes for any or all options, if you evaluate it is better to do so. 6. **Give** your decision in the same line as the option, so it's clear which option you are referring to. 7. **DO NOT** talk about values outside this list, since it's not necessary and out of scope for your task. 8. **MANDATORY:** When you decide to recommend a change for an option, your decision must be **certain, specific, and definitive**. **Vague alternatives** or **conditional suggestions** (e.g., "choose X or Y,", "choose X unless [reason]" or "maybe Z") **are strictly forbidden**. Provide only the single, final recommended value.

—

Assumptions

- When talking about the "Target" the system is referring to the animation representation of the real target object. So for example, if the target is transparent then it will be transparent over the real representation of the object, if the settings say that the target is duplicated. If the target is not duplicated then the animation representation will act as the real object (even though the real object is just hidden) - You are helping a developer in setting options for modify an animation that will show how the user have to perform its interaction (like a tutorial). The action you are going to analyze has nothing to do with the action the user has to perform to complete the steps, it is a separate function. So for example, "Trigger Mode" to "Action" means that the user pushes the "Activate Animation" button to play the animation, that helps the user learn the action he needs to do. None of these settings affect how the user acts to accomplish the action, only the appearance and behaviour of the animation itself. - **The developer has no control over implementation of features different than the ones described here**, the only choice he has is to set different options for the animation representation. - When you decide to recommend a change, give a **certain and final choice**, **not** vague alternatives. **Your recommendation must be the exact value** (e.g., "Trigger Mode" to "Continuous Loop," not "a faster triggering mode"). - Always be clear on whether you are recommending a change or not for each option, and specify when you are analyzing the user's current choice or your recommended new choice.

STRICT TERMINOLOGY GUIDANCE

DO NOT CONFUSE STEPS WITH ACTIONS. 1. **Step:**: A complete, high-level phase of the overall task. * *Example Steps: Preheating the oven, grabbing some gloves, putting on shoes. 2. **Action**: A single, physical or direct input operation performed *within* a Step. These are too granular to be considered Key Steps. * *Example Action (Interaction):* **aim ray**, **click**, **drag**, **select/grab**, **move**

LLM Model Analysis Request

Please analyze all the options in the system prompt and return the analysis for each one. The analysis **MUST** include all of the following options:

1. **Trigger Mode** 2. **Signifiers** 3. **Level of Detail** 4. **Perspective** 5. **Avatar Type** 6. **Avatar Options** 7. **Action Options** 8. **Outcome Options** 9. **Untriggering** 10. **Exit Transition**

Available Settings

Trigger Mode (Weight: 4)

Determines how the user activates the animation. Options include:

- **Gaze**: Allows hands-free activation by looking in the direction of the target object, but may lead to accidental triggers. - **Position**: Allows hands-free activation by moving nearby the target object, but may lead to accidental triggers. - **Timer**: Useful for time-sensitive tasks but may be frustrating if the user is not ready. - **Action**: Provides deliberate control, by letting the user activate the animation through the dedicated button in the playback interface, it obviously requires a voluntary intention from the user. - **Persistent**: Convenient but may be disruptive if the user is not expecting it.

Signifiers (Weight: 2)

Hints indicating where the animation is available:

- **Enabled**: Aids in discovery with hints guiding the attention of the users towards the target object. It may clutter the environment. - **Disabled**: Creates a cleaner look but may make features less apparent.

This option may be important for ”Gaze” and ”Position” Trigger Modes. The other ones do not rely heavily on this option, if not for understanding where the user needs to look when the animation is triggered.

Level of Detail (Weight: 3)

Specifies the amount of detail shown in the animation of the Task Workflow.

- **Everything**: Provides comprehensive information by displaying the animation of the entire task, phase by phase, from start to finish. This option covers all necessary major steps and minor steps alike, but may overwhelm users due to its length. - **Key Steps**: Provides a partial sequence showing only the **essential, major steps** required to complete the task. This option focuses on critical milestones. Keep in mind that this option does not show exclusively how to complete the current step, but it’s a ”task designer”-curated selection of steps from everything, broader than the single animation of ”Current Step” but narrower than ”Everything”. - **Current Step**: Simplifies understanding by showing a single animation focused exclusively on the current step the user is engaged in, but may omit helpful contextual details about the overall task flow.

Perspective (Weight: 3)

The experience can be viewed from:

- **First-person Perspective**: Increases immersion but limits overall view. - **Third-person Perspective**: Offers broader perspective but may reduce personal connection.

Avatar Type (Weight: 3)

If the avatar is visible, it can appear as:

- **Full Avatar**: a humanoid character demonstrating what the trainee should perform in the virtual environment. - **Partial Avatar**: shows only the relevant parts of virtual character for the considered interaction (only the hands in this system). - **Real User**: the humanoid character that performs the movements the user should do in real world (useful if the mapping between the tracking support and the virtual representation of the user is not direct).

Detailed avatars enhance immersion but may increase complexity. Simplified representations reduce cognitive load but might lessen engagement. **Every option** offers an **accurate and

realistic representation of the required user's actions** for completing the step. No option is fantasy-like; it always represents either the real user's controllers or a realistic human representation. The avatar is a realistic human representation that shows what the user should do like if another person was showing the user how to do it.

Avatar Options (Weight: 4)

These options are either *true* or *false* (in the parentheses you can find the possible values, first one is the "true" option, the second one is the "false" option), and they control the following:

- **Visible** (Visible / Not Visible): Whether the avatar will be seen during the preview. - **Duplicated** (Duplicated / Not Duplicated): If duplicated it will be shown alongside the user's one, if not the system will user the user's one. - **Ghosted** (Ghosted / Not Ghosted): Whether if the avatar used for the preview will be transparent (ghosted) or opaque. - **Dislocated** (Dislocated / Not Dislocated): Whether if the avatar will be slightly dislocated with respect to the original position.

IMPORTANT! * Although these four settings (**Visible, Duplicated, Ghosted, Dislocated**) are grouped under one category, you must treat them as four entirely separate and independent, yet connected boolean variables (as they modify the same part of the preview).** **Simultaneous Change:** You are permitted to recommend changing **any number** of these options at once (from zero to four) if you determine it is necessary to achieve the optimal VR preview experience. For example, recommending 'Visible: true' and 'Dislocated: false' simultaneously is a valid outcome. * If your optimal recommendation for **Visible** is **false**, the other variables in this group become irrelevant, so skip directly to the next category.

Your evaluation for this category must explicitly consider the optimal state for *each* of the four options individually.

Note: Each option must be treated separately in your assessment, while keeping the

Action Options (Weight: 4)

These options are either *true* or *false* (in the parentheses you can find the possible values, first one is the "true" option, the second one is the "false" option), and they control the following:

- **Visible** (Visible / Not Visible): Whether the representation of the action (an arrow with a panel that shows which interaction needs to be performed) will be seen during the preview. - **Dislocated** (Dislocated / Not Dislocated): Whether if the action representation from the preview will be slightly dislocated with respect to the original position.

IMPORTANT! **Although these four settings (**Visible, Dislocated**) are grouped under one category, you must treat them as four entirely separate and independent, yet connected boolean variables (as they modify the same part of the preview).** * If your optimal recommendation for **Visible** is **false**, the other variables in this group become irrelevant, so skip directly to the next category. **Simultaneous Change:** You are permitted to recommend changing **any number** of these options at once (from zero to two) if you determine it is necessary to achieve the optimal VR preview experience. For example, recommending 'Visible: true' and 'Dislocated: false' simultaneously is a valid outcome.

Outcome Options (Weight: 4)

These options are either *true* or *false* (in the parentheses you can find the possible values, first one is the "true" option, the second one is the "false" option), and they control the following:

- **Visible** (Visible / Not Visible): Whether the target object will be seen during the preview. - **Duplicated** (Duplicated / Not Duplicated): If duplicated it will be shown alongside the original target object, if not the system will simulate the use of the original (it will visually hide the original object and set the copy to be seen as the original one). - **Ghosted** (Ghosted / Not Ghosted): Whether if the target object from the preview will be transparent (ghosted) or opaque. - **Dislocated** (Dislocated / Not Dislocated): Whether if the target object from the preview will be slightly dislocated with respect to the original position.

IMPORTANT! **Although these four settings (**Visible, Duplicated, Ghosted, Dislocated**) are grouped under one category, you must treat them as four entirely separate and independent, yet connected boolean variables (as they modify the same part of the preview).** * If your optimal recommendation for **Visible** is **false**, the other variables in this group become irrelevant, so skip directly to the next category. **Simultaneous Change:** You are permitted to recommend changing **any number** of these options at once (from zero to four) if you determine it is necessary to achieve the optimal VR preview experience. For example, recommending ‘Visible: true‘ and ‘Dislocated: false‘ simultaneously is a valid outcome.

Untriggering (Weight: 2)

Determines how the animation is dismissed:

- **Gazing Away**: Offers quick dismissal but may be unintentionally triggered. - **Moving Away**: Requires deliberate action but may be cumbersome. - **Automatically**: Convenient but may disrupt the user’s flow.

Exit Transition (Weight: 4)

Transition back to the initial state can be:

- **Instantaneous**: Quick disappearance of the changes made during the preview. It may feel abrupt. - **Rewinding from End**: Provides context showing the animation in reverse, but takes longer. - **Fade Out and In**: Offers smoothness with a animation that, using a vignette, occludes the user’s sight for a moment and goes right back in the original state. It may delay return.

All the Exit Transition options are automatically triggered, the only difference is the appearance of the return of the initial state

—

Environmental Factors

Possible factors in displaying different representations of a virtual avatar when certain conditions are detected:

Noisy Environment (Weight: 1)

Background noise can interfere with understanding the instructor, making it difficult for the user.

Controller Unavailable (Weight: 2)

Without controllers, users need alternative interaction techniques like hand tracking, which may not be as precise.

Experience Level (Weight: 5)

- **Little to No Experience with VR** - **Some Experience with VR** - **A Lot of Experience with VR**

Inexperienced users may need detailed representations; experienced users may need only hints or reminders.

Difficulty Preference (Weight: 5)

- **Prefers Most Help Available** - **Moderate Amount of Help** - **Little to No Help**

Users preferring less difficulty may need the experience adjusted for lower complexity.

Realism (Weight: 3)

User preference for environment representation:

- **More Realistic**: Provides immersive experience but requires more resources. - **More Abstract**: More efficient but may not be as engaging.

D.2 User Prompt

The task you’ll evaluate is “Start a program” with the interaction “Click the mouse” as action and “Touch” as interaction

The settings are:

Trigger Mode: Button-triggered
Signifiers: Disabled
Level of Detail: Full-Animation
Visible Elements : Target Object, Avatar
Avatar Type: Puppet Representing the User
Perspective: Third-person Perspective
Target Position: Correct Position
Avatar Rendering: Transparent
Target Rendering: Transparent
Avatar Duplication: Duplicated
Target Duplication : Duplicated
Dismissal Mode: Automatically
Exit Transition: Instantaneous