Can You Move These Over There? Exploring an LLM-based VR Mover to Support Natural Multi-object Manipulation

Xiangzhi Eric Wang*
The Hong Kong Polytechnic University
Hong Kong SAR, China
xiangzhi.wang@connect.polyu.hk

Zackary P. T. Sin*†
The Hong Kong Polytechnic University
Hong Kong SAR, China
zackary-p-t.sin@polyu.edu.hk

Ye Jia
The Hong Kong Polytechnic University
Hong Kong SAR, China
ye-aimmeng.jia@connect.polyu.hk

Dan Archer
University College London
London, United Kingdom
daniel.archer.18@ucl.ac.uk

Wynonna H. Y. Fong
Heep Yunn School
Hong Kong SAR, China
s221001@hys.edu.hk

Qing Li
The Hong Kong Polytechnic University
Hong Kong SAR, China
qing-prof.li@polyu.edu.hk

Chen Li
The Hong Kong Polytechnic University
Hong Kong SAR, China
richard-chen.li@polyu.edu.hk

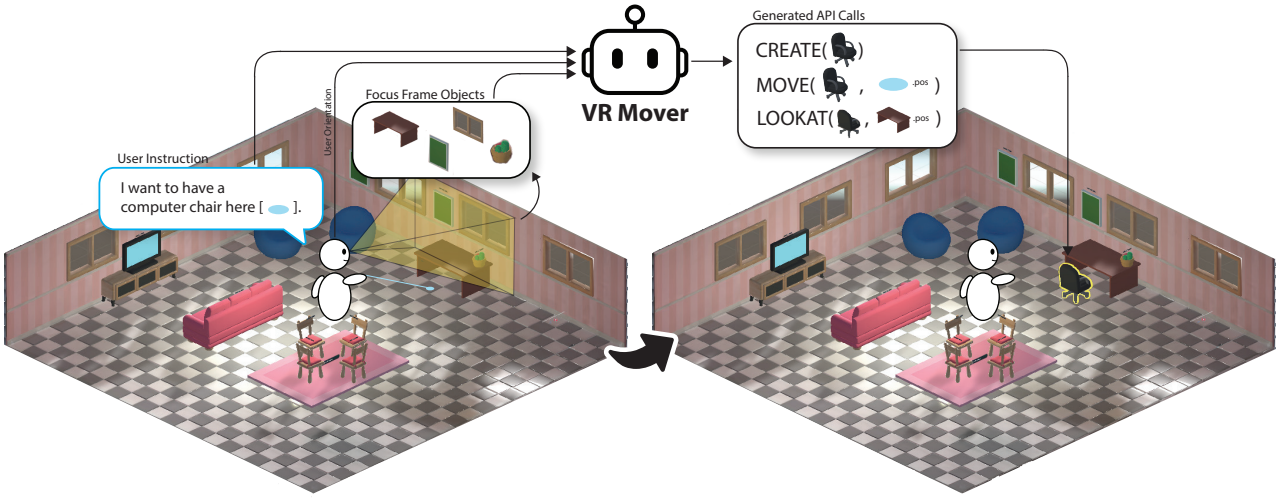

Figure 1: We propose *VR Mover*, an LLM-based interface for supporting natural object manipulation. It aggregates user-centric information, such as what the user is saying, seeing, and pointing at, to decide how to assist them in the placement of objects.

Abstract

In our daily lives, we naturally convey instructions for spatially manipulating objects using words and gestures. Transposing this form of interaction into virtual reality (VR) object manipulation can be beneficial. We propose *VR Mover*, an LLM-empowered multimodal interface that can understand and interpret the user's vocal instructions combined with gestures to support object manipulation. By simply pointing and speaking, the user can command the LLM to manipulate objects without structured input. Compared to classic interfaces, our user study demonstrates that *VR Mover* enhances user usability, overall experience, and performance on multi-object manipulation, while also reducing workload and arm fatigue. Users prefer the proposed natural interface for broad movements and may complementarily switch to gizmos or virtual hands for finer adjustments. These findings are believed to contribute to design implications for future LLM-based object manipulation interfaces, highlighting the potential for more intuitive and efficient user interactions in VR environments.

CCS Concepts

• **Human-centered computing** → **User studies**; **Gestural input**; **Virtual reality**.

Keywords

LLM, Object Manipulation, VR Mover, Natural User Interface

ACM Reference Format:
Xiangzhi Eric Wang, Zackary P. T. Sin, Ye Jia, Dan Archer, Wynonna H. Y. Fong, Qing Li, and Chen Li. 2025. Can You Move These Over There? Exploring an LLM-based VR Mover to Support Natural Multi-object Manipulation. In *The 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25), September 28–October 01, 2025, Busan, Republic of Korea.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3746059.3747673

*Co-first authors; both authors contributed equally to this research.
†Corresponding author

UIST '25, Busan, Republic of Korea

ACM ISBN 979-8-4007-2037-6/25/09
https://doi.org/10.1145/3746059.3747673

1 Introduction

In virtual reality (VR), object manipulation refers to the task of moving and manipulating 3D objects. It is a commonly used task in many applications and is generally associated with scene editing, for example, level design in VR game development, or customizing a personal space in the metaverse. However, despite its importance, the complexity of object manipulation interfaces may present challenges in terms of **usability**, especially for novice users who are less familiar with VR interactions. Typical interfaces require a sequence of actions — selecting an object, adjusting its transform (position, rotation, or/and scale) and confirming the modifications [77]. This three-step procedure may induce a learning curve, and make interactions less accessible for novice users, as they tend to focus more on exploring an environment rather than accomplishing tasks [32]. Thus, by improving **learnability**, which is a fundamental attribute of usability [54] that refers to how easily a user grasps the interface, it can generally improve the experience of users, particularly for those who are new to VR.

Beyond usability, another concern with object manipulation techniques is **fatigue**. Many VR interfaces rely on mid-air gestures or direct hand-based interactions, which can lead to the so-called "gorilla-arm effect" [29, 42]. Continued usage of these interaction methods introduces arm fatigue, limiting the time users can comfortably engage in object manipulation tasks [36, 44, 77]. Prior research [6, 77] has attempted to introduce gaze-based techniques to mitigate arm fatigue, but the approaches require specialised devices, introduce additional effort to learn, and can induce eye fatigue [57]. Aggregating the above, we can see that **workload** is also a noticeable concern. Object manipulation interfaces can be complex to use due to their high degree of freedom, and physical fatigue can further compound the perceived workload. As such, an interface that is simple but flexible to operate can be beneficial. Regardless, it is believed that **fatigue, workload, and usability** remain a challenge in the current context, and need to be addressed.

In addition, these concerns are particularly true in instances involving **multi-object manipulation**. There are many situations where the user needs to move multiple objects together. There seems to be a lack of discussion on how to effectively handle multi-object manipulation in VR, as most research focuses on improving manipulation in isolation — one object at a time [77]. When handling multiple objects, it is generally assumed that the user should first select several objects in tandem and then manipulate them as a group. While this is a valid strategy, it presumes that all grouped objects require identical manipulation, which is often not the case.

To address the issues raised earlier, we take note of how we perform "object manipulation" tasks in reality; this provides us with inspiration on how to ground a **natural interface**. For example, when we move residence or decorate rooms, it is possible to instruct others to help with moving objects. Humans tend to do so by naturally combining speech and gestures for spatial communication tasks [2]. The importance of this behaviour has also been observed in a virtual environment [40]. In addition, humans can provide unstructured and context-driven instructions that may involve multiple objects simultaneously. Given that this innate spatial communication is effective and intuitive, our goal is to translate it into VR, creating the effect of a virtual mover who assists with object manipulation tasks. This form of interaction recommends a natural interface in VR, where users can issue commands expressively and flexibly without the constraints of traditional methods. Thus, we reason that such an intuitive interface will be easy to use, quick to learn, and capable of mitigating fatigue and workload.

Inspired by how people communicate spatial manipulation instructions in the real world, as well as multimodal works such as Put-that-there [9], we propose leveraging a large language model (LLM) to realise a *VR Mover* that supports the user in object manipulation (Figure 1). There are recent related LLM works; however, their focus is on other domains, such as scene editing (e.g. LLMR [16]), rather than designing effective interfaces for object manipulation. Their models seem to lack user-centric consideration and exhibit response times that are too slow for object manipulation (15 seconds to several minutes). In order to realise an **LLM-based interface** that is suitable for object manipulation, we propose a specialised user-centric object manipulation interface. This interface is embedded with an atomic object manipulation function set, can enable multi-object interaction, has a much quicker response time (~2 seconds), and supports expressive multimodal input. We will later discuss this approach in greater detail in the technical methodology section. In addition, we will later show that the interaction with *VR Mover* aligns with how human visual working memory operates, making it theoretically more natural and intuitive to use.

To evaluate *VR Mover* — and, by extension, the properties of an LLM-based interface — we prepared the user study with two experimental settings. The first experiment measures how effective an interface is for single mid-air object manipulation and multi-object manipulation. The second experiment provides the users with a free-to-create environment to furnish a room, aiming to determine whether the interface supports practical and creative use cases. We compared our LLM-based interface with two other object manipulation interfaces. Primarily, we compared it with a commonly used controller-based interface where the user manipulates an object via gizmos [18] and virtual hands [77], both prevalent in popular software. Secondly, we compared it with a traditional rule-based speech system (i.e. an LLM-removed voice command variant) to highlight the necessity of LLM integration. To the best of our knowledge, we are the first to compare an LLM-based interface with classical object manipulation variants. As such, the empirical data provides findings on the advantages of LLM-driven object manipulation, particularly when multiple objects are involved.

Briefly, the contributions of this paper are as follows:

- *VR Mover*, a novel LLM-based multimodal interface for object manipulation that can handle unstructured, incomplete, and contextualised instructions, while enabling multiple-object manipulation from a user's perspective.
- The technical implementation of an LLM-based interface that features significant speed improvement (~2 seconds), and contextualises user-centric perception and multimodal signals.
- A user study demonstrating that an LLM-based interface like *VR Mover* can improve usability and user experience, as well as reduce arm fatigue and workload, compared to classical interfaces (i.e. gizmos, virtual hand, and voice

command), while further demonstrating the feasibility and benefit of multi-object manipulation as a novel dimension.
- A set of design recommendations for LLM-based interfaces derived from our implementation and user study findings.

2 Related Works

Here, works that are related to *VR Mover* are presented. Table 1 provides a quick summary on the uniqueness of our work in relation to recent LLM-based VR interface works. To our knowledge, we are the first to develop an LLM-based interface specifically for VR (multi-)object manipulation and compare it with classical variants (i.e. gizmos, virtual hand and rule-based voice command).

Table 1: Comparison with recent LLM-based interface works.

	LLMR [16]	**Dream CodeVR** [25]	**VR Copilot** [78]	***VR Mover*** (Ours)
Objective	Scene Edit	Scene Edit	Layout Authoring	Object Manipulation
LLM Purpose	Code Generation	Code Generation	User Intent	User Intent+ Manipulation
Gestural Input	✖	✖	✔	✔
Response Time	~20s	>15s	Unspecified	~2s
Compare Interfaces	✖	✖	Internal Designs	w/ External Interfaces

2.1 Object Manipulation in VR

Object manipulation is a long-standing and important topic in VR research, with hand-based interfaces being among the earliest methodologies for this purpose. Commonly, virtual hand [77] is recognised as the go-to method for hand-based manipulation. GO-GO [58] by Poupyrev et al. improves the virtual hand technique by employing a metaphorical extension of the user's arm. Mendes et al. developed MAiOR [48], which provides distinct translation and rotation functionalities. Gloumeau et al.'s PinNPivot [26] maps hand gestures to facilitate manipulation tasks. Alternatively, TabletInVR [65] shows it is also possible for hand gestures to be further supported by an additional touch-surface interface. Regardless, even if the mapping of hand and object can be direct and seamless, and an integrated control device achieves "well-coordinated movements" [42], the selection-adjustment-confirmation workflow seems to complicate the interaction. Especially the case of novice users, who pay much more attention to exploring the environment than accomplishing the task [32]. The usability can be affected since such interfaces tend not to be easy to learn, while learnability is a fundamental attribute of it [54].

Moreover, prolonged use of complex hand-based manipulation techniques is widely known to induce arm fatigue, which can adversely affect the user experience [36, 77]. Consequently, scholars have explored alternative approaches such as gaze-based methods. Jacob [35] proposed that eye movements could function as an input method. Yu et al. [77] developed several techniques that integrate gaze and hands for object selection and manipulation. Bao et al. [6] introduced several methods that utilise gaze not only for object selection but also to facilitate object movement. Despite the promise of gaze-based approaches, gaze tracking is not yet a standard in VR head-mounted devices; this may stall the broad application of gaze-driven techniques. Furthermore, although gaze-based approaches for object manipulation can reduce arm fatigue, they also introduce other issues such as eye fatigue [57] and ease-of-use (users need to learn how to use gaze to control movement, which is not common). As such, it is believed that object manipulation is not well-solved and warrants further investigation.

2.2 Voice-command Interface

Voice-command interface is a popular approach in VR, as it can have advantages [31]. It relates to our work as we also transcribe speech to text for processing. The scope of research in voice-enabled VR interfaces is extensive; there have been demonstrations on how to utilise it for design [52], 3D modelling [17], interactive dialogue [27] and navigation [33]. There are also voice-command works that focus on object manipulation. Aziz et al. [4] introduced several voice-controlled techniques to assist users with physical impairments in manipulating graphical object dimensions. Push-that-there [70] is a tabletop multi-robot object manipulation system. Zhou et al.'s research [79] elicits multimodal gesture and speech interaction for AR object manipulation. Whitlock et al. [73] investigated various modalities for manipulating objects at different distances in AR. Fernandez et al. recently developed Hands-Free VR [20], a voice interface that leverages advanced deep learning models to enable precise text-to-command translation. Also utilizing pointing like ours, WorldPoint [39] by Kim et al. proposed a multimodal interface for mobile applications that can be initiated by pointing and confirmed by voice command. Clay and Wilhelms [14] developed a system for specifying relationships in space, time, and motion between objects with predictability and extensibility in mind. Closest to our work, however, is the classic Put-that-there [9] by Bolt et al. that manipulates objects on a 2D screen. They proposed an interface that listens to pre-specified keywords and replaces pronouns with targets indicated by pointing. Like most voice-command interfaces, it predetermines a list of rules to execute procedurally. A problem with a voice-based interface is that it has been known to introduce obstacles to the user, namely that users have to familiarise themselves with the keywords or specific ways to trigger the commands [53]. Furthermore, it is believed that it can only execute relatively simple tasks. If the task cannot be directly mapped to the predefined rules, it will not function. We address this problem by introducing an LLM as the intelligent part of the interface that can interpret the users' intentions without the need for any predefined rules. To our knowledge, we are also the first work to conduct a user study that explicitly demonstrates the advantages of an LLM-based interface compared to a voice-command one for VR object manipulation.

2.3 LLM-based Interface

LLMs have captivated the global research community due to their demonstrated efficacy across various domains; some examples are

aiding programming tasks [38], facilitating the development of multimodal applications [75] and enhancing AR story-telling [13]. Recently, several studies have contributed to the development of LLM interfaces in VR. Bayat et al. [8] and VirtuWander [71] by Wang et al. investigated how LLM can improve a VR museum. Wan et al. [68], Shoa et al. [63], and John et al. [37] investigated LLM-based virtual avatars or agents that improve interaction with humans. GazePointAR [43] by Lee et al. proposed a multimodal question-answering assistant that can be interacted with gaze and pointing signals. Aghel Manesh et al. [1] have conducted an elicitation study that extracted some behaviours on how people will use an LLM for VR scene editing. For LLM works, the closest to ours are research involving in-VR scene editing, such as LLMR [16] by De La Torre et al. and DreamCodeVR [25] by Giunchi et al. LLMR focuses on introducing a generalised framework that enables the LLMs to execute various activities in VR, while DreamCodeVR focuses on in-VR programming. As such, there is a lack of investigation into exactly what role LLM can play, specifically, in complex multi-object manipulation scenarios. Although technically LLMR and DreamCodeVR can perform object manipulation, in practice, their response time of 15 seconds to several minutes is too slow as a functional manipulation interface. They also lack a specialised multimodal interface to allow the user to quickly control the manipulation naturally. Our work, by contrast, streamlines the system to enable *VR Mover* to respond in ~2 seconds and accept multimodal signals, including gestural cues and user perspective. We note GesPrompt [34] as a concurrent work similar to ours; their work focuses more on hand gesture identification and processing, while ours focuses more on multi-object manipulation and, we believe, presents a more complete user study that compares with classical object manipulation interfaces.

2.4 Layout Generation

A somewhat related area of research to object manipulation is layout generation, which focuses on the placement of 2D layouts. For instance, Miguel et al.'s GAUDI [7] is a generative model that facilitates both unconditional and conditional generation of 3D scenes. Another example is Handa et al.'s SceneNet [28], which is a framework designed to generate annotated 3D scenes. With the advent of LLM, Feng et al.'s LayoutGPT [19], Fu et al.'s AnyHome [22], Ocal et al's SceneTeller [55], and Wang et al.'s Chat2Layout [69] have emerged to utilise LLM's intelligence to generate layouts. Very recently, VRCopilot [78] by Zhang et al. proposed a layout authoring system for VR. It uses a trained model to generate the layout. Our work has several differences from theirs; aside from our work focusing on object manipulation, their work also limits the LLM's role solely to interpreting user intentions.

3 Research Objectives

To explore an LLM-based object manipulation interface, we have set the following research objectives. For clarity, we also indicate where they are addressed in the paper.

- (**R1**) Investigate a set of **natural interactions** for object manipulation, along with its intuition and theoretical motivation (Section 4).
- (**R2**) Propose a **technical methodology** that allows using an LLM to support the object manipulation interactions (Section 5).
- (**R3**) Investigate the **benefits and drawbacks** between an LLM-empowered natural interface and a classic manipulation interface (Section 6-8).
- (**R4**) Investigate the **benefits and necessities of an LLM** for object manipulation tasks (Section 6-8).
- (**R5**) Identify **design recommendations**, based on our findings, for future LLM-based VR interfaces (Section 9).

4 VR Mover: A Supportive Natural User Interface for Object Manipulation

Here, we start with the first research objective **R1**. In this section, we discuss the rationale and cognitive theory behind *VR Mover*, and how they affect its interaction design. First, let us visualise how we usually communicate spatial manipulation by imagining that we have movers to help us move into a new house; we may do the following: (1) point at a chair and ask the mover to move it to a specific location by pointing where we want it to go; (2) verbally ask the mover to move the dining chairs and table to a general location by pointing; (3) verbally ask the mover to move a table decoration on top of a dining table; (4) gesture in one direction to ask the mover to move an object in that specific direction; (5) use the surroundings to describe where to place an object; and, (6) once the mover has finished moving an object, we may fine-tune the final position ourselves.

We will later show how we achieve similar interactions with *VR Mover*. In the next subsection, we will discuss the cognitive theory behind some of the expressions and communication we used here.

4.1 Visual Working Memory

Whether we want to manipulate an object in VR or attempt to communicate spatial manipulation instructions, it is most likely that we have a mental image and then we manipulate it internally first. This human cognitive process is part of our visual working memory (VWM) [67]. VWM is how humans hold, manipulate and interpret information for a variety of everyday tasks [5, 45]. How VWM affects our ability to manipulate and handle memory for object location and movement tasks is of particular interest to us. Two governing principles in VWM can provide insight into object manipulation interfaces.

• **Chunking:** Humans have a tendency to group objects together to facilitate comprehension and communication. In the context of VWM, we have a process called chunking, which encodes information into larger perceptual chunks [51, 66]. Generally, humans may chunk objects together based on similar colours, locations, and shapes (e.g. chairs of the same set) [3]. Alternatively, when chunking an environment, larger environments may be organised as nested sub-environments [59]; that is, objects and their locations may be chunked together into memories represented as an area (e.g. dining area). Thus, in the context of object manipulation, we may infer that people will follow this tendency to group related objects during a manipulation task. Thus, a convenient method to select or refer to a group of objects is a topic worth exploring for object manipulation.

• **Coarse-to-fine Processing:** Another tendency of human nature is to process visual information in a coarse-to-fine manner

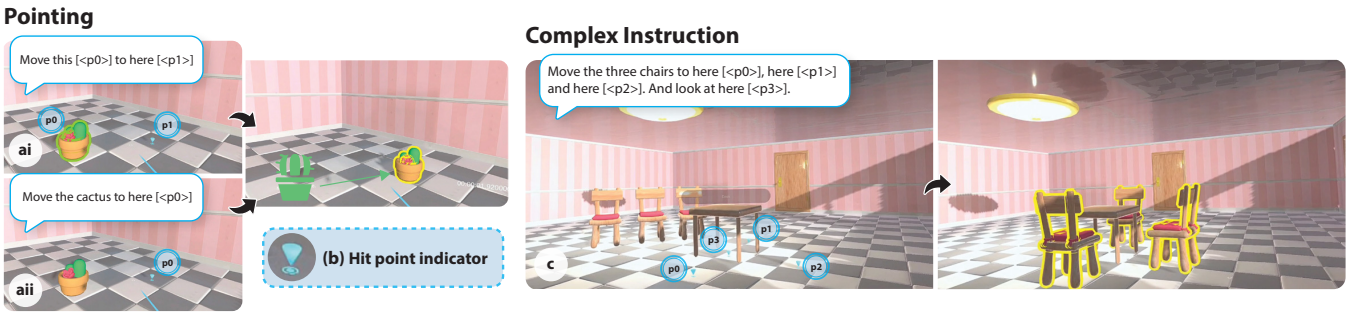

Figure 2: To move an object, (ai) the user can first specify an object and its target by pointing. However, as *VR Mover* is aware of what the user is seeing, (aii) the user can directly use speech to refer to the cactus. Note that, `p0` and `p1` are the first and second hit points from pointing, and (b) the hit point is visualised. (c) *VR Mover* can also handle complex instructions such as asynchronous multi-object manipulation, where different operations are applied to objects (e.g. moving and rotating).

[64], which is referred to as the so-called coarse-to-fine theory. Recent works have shown that VWM also follows this coarse-to-fine process, where the mental image is constructed gradually [23, 24]. It should be noted, however, another work has pointed out that the coarse-to-fine process is not the only cognitive pathway in VWM [76]. Still, it has been shown that coarse information takes precedence over detailed information [24]. Thus, based on these VWM works, it can be inferred that object manipulation may also involve a coarse-to-fine process. We should consider a fast coarse placement initially, followed by a fine-tuned adjustment later.

4.2 Interaction Design

Here, we discuss how the user can interact with the *VR Mover*.

• **Pointing:** In the real world, people naturally use pointing as a method to communicate a point of interest or indicate an object of interest [15]. In the case of *VR Mover*, the user can point at an object and then point at a position to either move the object by saying "Move the chair [point] to here [point]" (Figure 2*a*); or to make the object rotate towards that point by saying "Make the chair [point] look at here [point]". It should be noted that our current implementation requires the user to actively press A on the Meta Quest hand controller to indicate that they are pointing at something and [point] indicates when the user has pointed.

• **Lining:** Aside from pointing, we have different ways to gesturally indicate spatial information. Another method is to draw a line (lining) to indicate direction or a line of interest. Similar to pointing, lining is simple and expressive (e.g. lines can express direction and magnitude), and people generally already understand them. For example, the user can say "move the object that way [line]" (Figure 3*a*), "I want 4 pictures along the wall here [line]" (Figure 3*b*) or "make the object face that direction [line]". Note that the user needs to press (release) B to start (end) drawing a line in our environment.

• **User-centric:** Similar to a human mover, a *VR Mover* also tries to understand the requirement by trying to see the manipulation from the user's perspective (Figure 2*aii*). For example, in the case of a chair positioned in front of the user, they can say "move the chair away from me".

• **Asynchronous Multi-object Manipulation:** As discussed in the VWM discussion earlier (subsection 4.1), humans tend to group

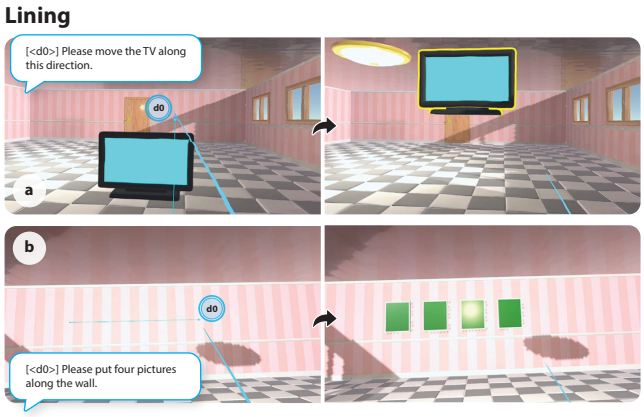

Figure 3: By drawing a line (lining), a user can express different manipulations. Here, we show the user using (a) a line (`d0`) to represent a moving vector, and (b) a line to indicate where the pictures should roughly be placed.

things together. With VR Mover, it is possible to quickly refer to multiple objects. They can be explicitly addressed by pointing, or they can be implicitly addressed by saying "move the three chairs to here [point], here [point] and here [point]" (Figure 2*c*). We referred to this as asynchronous multi-object manipulation because the manipulation applied to each object is different.

• **Spatial Common Sense:** As *VR Mover* is powered by an LLM, it is embedded with a degree of common sense. For example, when asked to place a dining area in a scene, it understands that it involves adding chairs and a table. Furthermore, *VR Mover* will also turn the chairs to face the table (Figure 4*a*).

• **Context-aware:** As a smart interface, *VR Mover* is aware of the current context. The user may first ask the chair to move to a particular location and then say "move it back". *VR Mover* will still understand what "it" is and where the original location to move back to is as well (Figure 4*b* and 4*c*).

• **Compound Instruction:** It is also possible to stack multiple related or unrelated instructions together. For example, we can say "move the chair to here and then make it face the window".

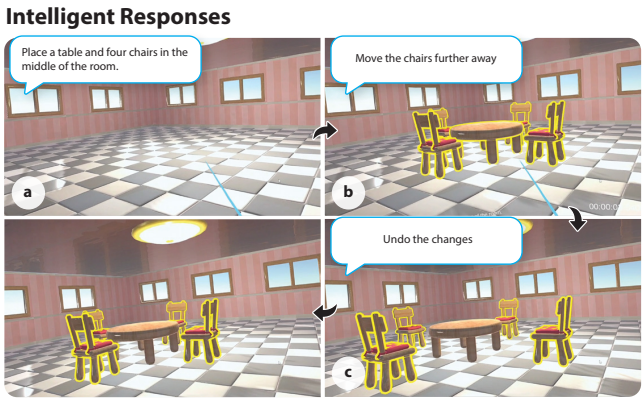

Figure 4: *VR Mover* has intelligent responses; (a) it has spatial common sense such that it knows the chairs should face the table, (b) it is aware of the user's context (e.g. what is referred to implicitly) and (c) it is adaptive enough to fulfil a user's undo request despite that there is no undo function.

• **Manipulation Fine-tuning:** Finally, it should be noted that the intelligent part of *VR Mover* is not designed to complete the entire manipulation autonomously. Just as in real life, where we may ask the mover to move the intended objects to an approximate location, and later fine-tune the exact placement. Thus, *VR Mover* should be coupled with classic techniques such as gizmos or virtual hands to let the user perform fine-tuning, forming a coarse-to-fine manipulation process.

However, as *VR Mover* is an intelligent interface, there are multiple ways to interact with it. More interactive examples, such as area-drawing, are provided in Appendix *A.1* and later in Section 9.2.

5 Methodology of VR Mover

LLMs have shown promise in spatial arrangements, complex task sequences, and as specialised agents [8, 16, 69, 71]. However, similar LLM-integrated VR research projects suffer from long response times, ranging from 15 seconds to several minutes [16, 25] (but note that they used older and probably slower models), while interactive VR interfaces crucially rely on real-time responses. We believe there is currently no LLM solution that is suitable for object manipulation, and thus, here we address the second research objective **R2** to provide such a solution. We propose that object manipulation can be deconstructed into an atomized function set that is quick to generate. On the other hand, there is currently a lack of investigation on how to design an LLM model that is user-aware (e.g. understand instructions from the user's perspective). To this end, we also propose a user-centric module that can quickly inject the user's perception and multimodal signal into the prompt. In total, *VR Mover* has four key components (Figure 6): the *scene modelling* (Section 5.1), which encodes objects into a text format that is readable by LLM; the *user-centric augmentation module* (Section 5.2), which processes speech, perception and gestural data; the *LLM processing module* (Section 5.3), which accepts the augmented prompt and generates manipulation command with atomized functions; and, the *scene update module* (Section 5.4), which processes the API calls generated by the LLM to update the scene.

5.1 Scene modelling

To enable the LLM to process objects, 3D spatial information is converted into a text-based format while preserving key features. Following the taxonomy [21], the scene elements are categorised into environmental objects (static) and manipulatable objects (dynamic and interactive). For example, if the virtual environment is a large empty room with furniture to manipulate, the scene elements should be floor, walls, and windows and the manipulatable objects should be the furniture. The spatial information of the objects is embedded as oriented bounding boxes (OBBs) [61] (cuboids with position, rotation, dimension). Additional metadata includes the object's name and its description (which can be generated by an image-to-text model). The objects (specifically, their OBBs and metadata) are then embedded into a JSON format and fed to the LLM. Figure 5 illustrates the JSON expression of manipulatable prefabs, manipulatable objects, and environmental objects, as well as where they are referenced or updated. We provide more details on how they are described to the LLM (via system prompt) in Appendix *E*.

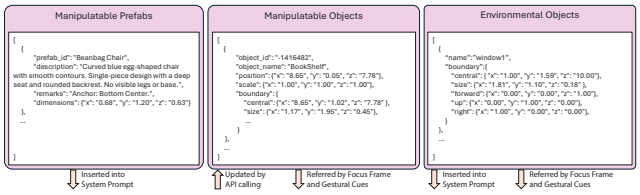

Figure 5: JSON expression of the manipulatable prefabs, manipulatable objects, and environmental objects.

5.2 User-Centric Augmentation Module

This module enables *VR Mover* to manipulate based on the user's perspective. In the following, we discuss how the model processes what the user is saying (Section 5.2.1), seeing (Section 5.2.2), and indicating (Section 5.2.3), as well as how to efficiently embed them (Section 5.2.4).

5.2.1 Processing Speech. For speech-to-text, we used Microsoft Azure's cloud service. In our implementation, it listens to voice signals from the microphone. The speech-to-text model needs to return the transcript with metadata, specifically, the timestamp of each spoken word, which is needed for time serialisation later (Section 5.2.4). If the user has stopped talking, it also needs to return a signal to trigger a packed request to the LLM. Note that interjections (e.g. "umm", "OK") are filtered and ignored.

5.2.2 Processing User Perception with Focus Frames. It is beneficial if the LLM can "see" what the user is seeing. Thus, we send the objects in the view frustum to the LLM to provide a perspective of the user's point of view. However, continually recording the head motion or objects in the view frustum generates a huge amount of data, and it is not straightforward for the LLM to identify the object of interest. We observed that users tend to look in the general direction of the objects of interest when referring to them. As such, we introduce focus-frames (Figure 7), which are groups of continuous

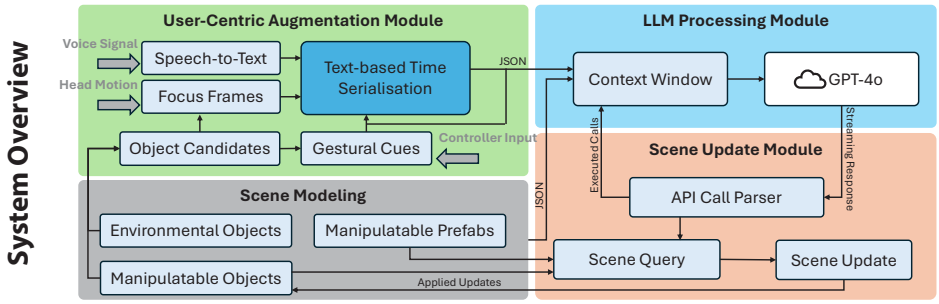

Figure 6: The system overview of *VR Mover*; it has four major modules: scene modelling, user-centric augmentation module, LLM processing module and scene update module. We used GPT-4o as the LLM's core.

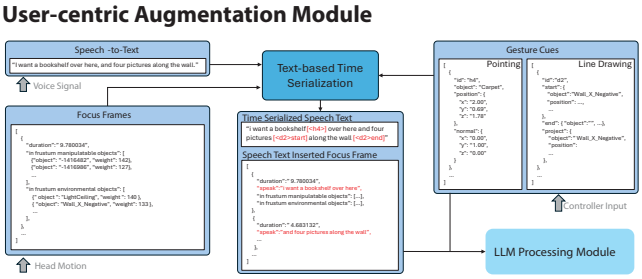

Figure 7: The User-centric augmentation module. The in-figure example shows the recognised speech has gestural cues inserted, and segmented into the focus frame groups. Changes after the time serialisation are highlighted in red.

viewports over a short period while the user is speaking. The system averages the viewports within the current focus-frame group, accumulating the objects that appear in the view frustum, and ranking them based on the screen distance to the centre. When the player's head motion changes beyond the threshold, or when their speech stops, the current group ends, and the current focus-frame moves to the next group. A focus-frame group with insufficient temporal duration will be filtered out. In each focus-frame group, the higher the ranking is, the more likely the user is to look at the related object during that period. Although it is possible that a user may not look at the objects of interest when referring to them, in practice, this does not seem to happen, and thus empirically, our focus-frame system seems to generally work.

5.2.3 Processing Gestural Cues. To enable multimodal instruction, it is necessary to capture and embed gestural cues (i.e. pointing and lining). A point gesture is defined as the intersection between the hand ray and an virtual object when the button is pressed. Similarly, a lining gesture is determined by the intersection points at the start of the button press, while the endpoint is computed by applying the hand's delta movement (i.e., the change in hand position during the button pressing) to the start point. In addition to the positions of the points, other supportive data, such as hit object(s), the surface normal at the intersection and the operation's timestamp, are included. Those data provide a richer context for interpreting user gestures within the virtual environment.

5.2.4 Text-based Time Serialisation for Efficient Signal Injection. Timing is an important consideration for multimodal instruction; it is important to pair the gestural cues and focus frames with the transcribed speech in such a way that it is clear to see how the multimodal signals work together. Previous methods use pronoun replacement via keywords [9, 43], but this necessitates rigid assumptions on the wording of the instructions, requiring users to familiarise themselves with it and restricting the ability to interact with multiple objects. In contrast, we propose an efficient signal injection scheme to quickly insert gestural cues via ID tags (Figure 7). For example, "`put the chair here [<p0>], and [<d0>start] I want four pictures to line [<d0>end] the wall`" showcase that "`p0`" and "`d0`" are, respectively, the IDs of the pointing and lining that are injected into the text transcriptions based on the timing. Then, to associate what the user was referring to when looking at a specific area, the transcription segments are inserted into the focus frames together as a JSON field.

5.3 LLM Processing Module

This module, of course, is the "brain" of the interface (Figure 8); it accepts the output of the user-centric augmentation module and then returns API calls to perform object manipulation automation. We first discuss how to prompt engineer the LLM model to behave as a mover that can support object manipulation. Then, we discuss how the LLM utilises the atomized object manipulation functions to execute the users' requests. For the LLM, we used GPT-4o [56] hosted on Microsoft Azure, similar to other works [16, 78].

5.3.1 Prompt Engineering. Prompting guides the behaviour of the LLM, and there are three types of prompts. In our system prompt, we provided a meta-prompt description to define its role and principles as an intelligent interface, which includes specifying the formatting for input and output. In addition, we also specified the list of available atomic function calls (5.3.2) for the LLM to generate the response, provided the environmental objects, and manipulatable prefabs for it to place and manipulate. Upon each user request, the user-centric augmentation module aforementioned generates the user prompt in JSON format, which will be processed by the LLM. To improve the object manipulation result, we also provide a one-shot example pair of user and assistant prompts. To stay within the context limit of LLMs, we only keep the latest 3 pairs

Table 2: The list of atomized object manipulation functions.

Function	Description
CREATE(string prefab_id);	Create an instance based on the prefab ID
MOVE(string object_id, float? x = null, float? y = null, float? z = null);	Set the position of an object
FORWARD(string object_id, float x = 0, float y = 0, float z = 0);	Set the forward direction of the object
LOOKAT(string object_id, float? x = null, float? y = null, float? z = null);	Set a position for the object to look at
SCALE(string object_id, float? x = null, float? y = null, float? z = null);	Set the scale of an object
DELETE(string object_id);	Delete an object by its ID
MESSAGE(string content);	Send a text message
EXPLAIN(string reason);	Send a debug text message

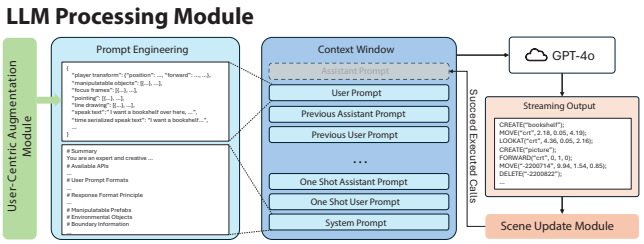

Figure 8: The LLM processing module. The context window manages conversation history. GPT-4o processes and generates streaming output, which will be passed to the scene update module.

of user-assistant prompts for context. To better demonstrate, we showcase the prompts and the generated result in Appendix *F*.

5.3.2 Atomized Object Manipulation Functions for Real-time Response. As mentioned, there was previous LLM work that focused on scene editing; research projects such as LLMR and DreamCodeVR are the closest to ours but they have a response time ranging from 15 seconds to several minutes, which is not believed to be sufficiently responsive for object manipulation. There is also layout generation work [71, 72], like Chat2Layout [69], that shares some similarities with our work, but their response time is also measured in minutes. It is believed that the main bottleneck lies in their generation strategy: they generate either a code script or JSON. For object manipulation, however, we speculate that a complex manipulation goal can be deconstructed into multiple atomic steps.

To achieve a significantly accelerated speed, we propose that a set of atomized object manipulation functions can be given to the LLM to weave a more complex object manipulation task. Instead of generating an entire script of code or JSON, the LLM is tasked with generating API calls (the full list of APIs are shown in Table 2), which are simpler and shorter, and therefore much quicker to generate. The LLM follows a line-wise API function calling in the format of "`<function name>(<param1>, <param2>, ...);\n`". Another advantage over code script and JSON is that the atomic functions can be executed individually as they arrive, instead of waiting for the entire output from the LLM. On average, the response delay (counted from the request sent to the LLM to the first line of the executed API call) is 2.29 seconds. We formally report the speed in the next section. Almost all users were satisfied with this response speed, which also enabled us to later compare an LLM-based interface with classic interfaces in the user study.

5.4 Scene Update Module

This module is in charge of parsing the object manipulation API calls from the LLM response and updating the scene. When a new line is received, the module is triggered and starts to extract the method name and corresponding parameters. The API call is then mapped onto the real runtime functions and invoked asynchronously. To ensure the robustness of the interface, any string sequence can be sent to this module, but only correctly recognised (matched by regular expressions) API calls will be performed. Thus, essentially, API calls from LLM are pushed to a buffer, and then a frame-by-frame function (e.g. OnUpdate() in Unity) combines multiple buffered calls to update the scene; this does not require recompilation because the changes occur at runtime. As nearly all the LLM-generated function calls are executable in practice (see next subsection), we did not explicitly handle the possible errors (e.g. non-existing function, wrong format, or invalid parameters) for simplicity's sake.

5.5 LLM Evaluation

Here, we performed some analysis on the performance of *VR Mover*. Aside from the default GPT-4o model, we also chose Llama3.1-405B and Llama3.1-70B [49] to include in our comparison. To quantitatively evaluate *VR Mover*'s speed and error rate, we prepared 35 prompts (Appendix *B.1*) for testing and repeatedly ran them 5 times.

• **Response Time:** As shown in Figure 9*a*, *VR Mover*, on average, responds in 2.29 seconds with GPT-4o as the LLM of choice. With Llama3.1-405B and Llama3.1-70B, the results are 5.42 and 2.35 seconds, respectively. As mentioned in subsubsection 5.3.2, this real-time performance mainly relies on the stream of the generated short atomic API calls, rather than the LLM of choice. Even though a slower LLM leads to a longer response time, the system is still faster than a code / JSON generation method. It should be noted that response time scales according to task complexity; however, we observed that with GPT-4o, in most cases, a manipulation required by a user takes a response time of around 2 seconds.

• **Error Rate:** While using GPT-4o, there is virtually no error when executing the returned API calls (Figure 9*b*). On the other hand, with Llama3.1-405B and Llama3.1-70B, 1.62% and 0.28% of the requests resulted in errors. However, we note that all errors came from the LLM generating incorrect prefab IDs, which should be easily fixable by searching for the closest match. Regardless, we believe our atomic object manipulation function set contributes to

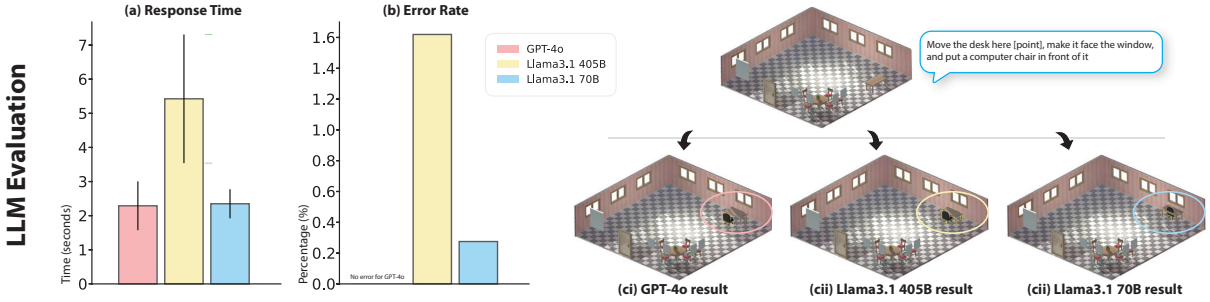

Figure 9: *VR Mover*'s (a) response time, (b) error rate and (c) placement examples given different models.

a significantly lower error rate compared to previous LLM work, which seemed to encounter error rates around 15%-20% [16, 25].

• **Reproducibility:** It is known that GPT-4o is not guaranteed to generate reproducible (deterministic) [50] results, although prompt engineering with system prompt design and common techniques like a one-shot example can improve consistency. With open-source models (like Llama), however, complete reproducibility can be achieved by locking the seed and using the same configuration. Regardless, in our experiment, we show that the three LLM models tested can reliably produce results that are reasonable and consistent for object manipulation (Figure 9*c*). More results are shown in Appendix *B.2*. We also show that for GPT-4o, even when we removed all one-shot and in-prompt examples, the model is still able to generate reasonable results (see Appendix *B.3*).

6 User Study

In this section, the user study setting is presented; its result and the subsequent discussion (which address objectives **R3** and **R4**) will be presented later.

6.1 Experimental Techniques

In this study, we compared the following three techniques. We hope to derive insights from how users use and view them differently.

• **Gizmos + Virtual Hand (*Control*):** This is the technique that acts as the control. It involves using the gizmos (Figure 10*a*) and virtual hand [47] (Figure 10*b*). The gizmos can be controlled via the interactive ray. For virtual hand, it can operate even when the user's hand is not next to the object (somewhat similar to a remote hand [77]). The combination of gizmos and virtual hand is picked as it is believed to be a sufficient representation of commonly-used object manipulation techniques, see previous research [18, 77] and current applications. Multi-object selection to perform synchronised manipulation (e.g. moving all objects in the same way) is possible by first selecting the objects (with a quick press of the A button). For brevity, we refer to this interface as the *Control* technique.

• ***Voice Command*:** To highlight the importance of an LLM that can understand the perspective and unstructured user instructions, we have separately developed a rule-based voice command variant, which is similar to Put-that-there [9]. Instead of an LLM process, a set of structured voice commands is directly mapped to the same set of atomic APIs shared by *VR Mover*. *Voice Command*'s implementation is similar to a recent voice-command rule-based locomotion technique [33], which utilises regular expressions and mapping. It follows a grammar of <verb, subject, (direction), (unit)>, where the brackets "()" indicate optional input. The most important commands are "move this here" and "rotate this here" in which the user can specify the object(s) to manipulate via selecting and the target of manipulation via pointing. For details of more operations, like directional voice command, please refer to Appendix *C.1*. It should be noted that this interface variant only includes pointing, but not lining, as the latter is a method to express ambiguous requests (inferring direction, area, and movement); thus, it is believed that there is no clear way to implement lining for *Voice Command*. Regardless, it is not expected to perform well in the fine-tuning of object placement. This *Voice Command* interface, therefore, also includes the same gizmos and virtual hand from the *Control* technique (Figure 10*c*). Lastly, note that comparing *VR Mover* with *Voice Command* can provide insights into why an LLM is needed to replace a rule-based voice-command-driven system.

• ***VR Mover*:** The proposed *VR Mover* is an LLM-based interface that can support the user in object manipulation. Similar to *Voice Command*, this technique includes gizmos and virtual hand for users' adjustments to complete the manipulation.

Figure 10: The (a) gizmos and (b) virtual hand are provided in all three experimental techniques. The *Voice Command* interface (c) is an LLM-removed rule-based variant. It only supported predefined commands such as "move this here".

6.2 Experimental Tasks

Similar to a previous VR object manipulation study [77], our user study includes two tasks. Task 1 is a performance-centric task to evaluate a user's ability to move object(s) given specific target(s). It is further divided into two sub-tasks, Task 1A and 1B. Task 2 is a creative-oriented task that allows the users to freely move objects to embellish the content of a VR room.

• **(Task 1A) Single Mid-air Object Manipulation:** The first sub-task (Figure 11*ai*) involves the user moving an object from source to target. A semi-transparent version of the manipulatable

target is used to indicate the goal. The scene involves a chair placed on the ground and a target in mid-air. The distance between the source object and the target is measured by the distances between the eight points of a bounding box. Once the average distance is below a threshold, the object is considered to have reached the target. In this sub-task, we aim to evaluate a technique's ability to handle mid-air manipulation. The chair is around one meter tall and both the object and target are three meters away from the user.

• **(Task 1B) Multi-object Manipulation:** The second sub-task (Figure 11*aii*) involves moving several manipulable objects to their targets. Similar to the previous task, semi-transparent targets are used to indicate the goals. This sub-task differs from the previous one in that it aims to evaluate a technique's ability in handling multi-object manipulation.

• **(Task 2) Sandbox Room for Object Placement:** In order to see how users use the proposed LLM-supported technique in a more realistic setting, this task (Figure 11*b*) places the users inside an empty room. Via a view-activated prefab menu in VR, the user can choose from an assortment of predetermined 3D models to decorate the room. A soft goal of the task is that the user should try to replicate the mini-room shown in the VR UI panel. However, they are encouraged to test the interaction technique as they see fit. It is believed this gives the user an opportunity to test the object manipulation workflow and provide subjective feedback. As there is no specific end goal for Task 2, each user is simply given 7 minutes to freely manipulate the objects in the room. This task's design follows a previous object manipulation research [77] to provide a real use case for evaluation.

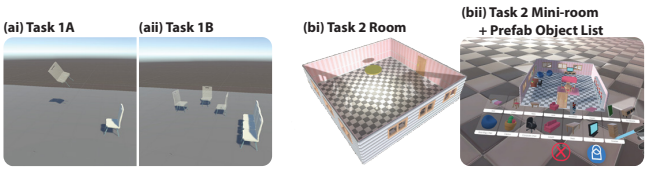

Figure 11: The left side is the environment of (ai) Task 1A and (aii) 1B; the goal of the user is to move the object(s) to the (semi-transparent) target(s). The right side is Task 2; given an (bi) empty room the user is instructed to populate it according to the (bii) reference mini-room with the prefab objects.

6.3 Participants and Procedure

24 participants were recruited for the user study via advertisements on campus (7 male, 16 female, 1 not specified); their ages ranged from 18 to 35 years (22.96 ± 3.83). Most of the users had prior experience with VR, but only used it sparingly (~62% and ~17% of the users only used VR a few times per year and month, respectively). Meta Quest 3 was used as the VR device of choice. For each participant, the following procedure was used: (Step 1) The user filled out a consent form and a basic information questionnaire (e.g. Age); (2) Steps 3a-3e were repeated until all three techniques have been tried; (3a) Based on the random order the user was assigned, the technique's overview was presented to the user; (3b) a practice session with an in-VR tutorial was played to familiarise the user with the interface; (3c) Task 1 was performed without a time constraint; (3d) Task 2 was performed with a 7-min time constraint; (3e) The user filled out a SUS, PQ, NASA-TLX, Borg C10 and UEQ-S after the completion of both tasks with a technique; (4) At this point, the user had completed all the trials and was then given the preference ranking to fill out, in addition to an interview session.

Experimenters observed user behaviour and recorded the content displayed in the headset. The entire study session for each participant lasted approximately 2 hours, including all tasks, evaluations, and interviews. The experimental virtual environments were implemented in C# on Unity. Ethical approval for the study was obtained from the institutional review board.

6.4 Measures

The following measures are used to evaluate the three techniques. Note that the arrow ↑ (↓) means the higher (lower) the better.

• **Coarse Manipulation Time (↓)** is the time it took for the user to move the object to a "near enough" position (in Task 1). As discussed in a previous paper, the user will spend significant time adjusting the object to fit the target more closely [77]. This measure provides insights into the time required for a user to move an object from the starting position to a location proximate to the target position. An object is considered to have hit the coarse target when the average distance of the eight points of the bounding box is smaller than 0.3m.

• **Fine Manipulation Time (↓)** is the total duration needed by the user to more closely fit the manipulable object to the target (in Task 1). For fine targeting, an object is considered to have reached its target when the average of bounding box points' distances is less than 0.12m. This threshold is chosen based on early testing that most users have significant difficulty reaching the target if it is lower.

• **Hand Movement Distance (↓)** is the total accumulated hand movement that occurred during Task 1.

• **Arm Fatigue (↓)** is measured by Borg C10 [10, 36], a rating of perceived exertion in the arm.

• **Usability (↑)** is quantified using a modified two-item System Usability Scale (SUS), adapted from the original SUS [11]. Previous work validated that this two-item design is highly accurate [60].

• **Presence (↑)** is measured by the Presence Questionnaire (PQ) [74], which comprises 12 carefully selected items across three pertinent subjects crucial for evaluating the experimental interfaces; they are realism, quality of interface and self-evaluation of performance. Each item is assessed using a 7-point Likert scale.

• **Workload (↓)** is retrieved by the NASA Task Load Index (NASA-TLX) [30], presented with a 10-point range.

• **Preference** from users is extracted from a preference ranking question where the user will be asked to rank their preference on technique from most favourite (1st choice) to least favourite (3rd choice).

• **User Experience (↑)** is measured via the short version of the user experience questionnaire (UEQ-S) [62] with a 7-point Likert scale. It captures overall, practical and enjoyable feedback.

Note that error rate is not used as users can use multiple manipulations to achieve the goal, while layout accuracy is unsuitable for Task 2 as it is a sandbox task with no fixed objective.

7 Results

When reporting a measure's mean (SD) for each technique, the order of reporting is always *Control* group, *Voice Command* and *VR Mover*. For each measure, the Shapiro-Wilk test is first conducted to check for data normality. If that is the case, typical repeated measures ANOVA and Student's t-tests are used. Otherwise, the Friedman test is performed and the paired test is done with Wilcoxon signed-rank. Bonferroni correction is performed on all post-hoc analyses. Note that we are only interested in comparing *VR Mover* with *Voice Command* and the *Control* group with the post-hoc tests. Effect size is shown with Cohen's d. The threshold for statistical significance was established at a $p-value$ of 0.05 for all analyses. In addition, we have recorded the audio in the structured interview. The interview records were transcribed for coding. Later in the discussion, we will present some of the qualitative user feedback.

7.1 Manipulation Time (Task 1)

For both subtasks of Task 1, the fine and coarse manipulation time are separately measured. For Task 1A, where the user needs to fit a single object to a mid-air target, the mean (SD) of coarse manipulation time (Figure 12*ai*) is, respectively, 58.142 (31.304), 73.620 (32.320) and 60.765 (28.295), for *Control*, *Voice Command* and *VR Mover*, and the fine manipulation time (Figure 12*aii*) is 70.682 (32.901), 86.860 (30.565) and 72.11 (28.155), respectively. There is a significant main effect for coarse ($p = 0.0102$, $d = -0.423$) and fine ($p = 0.0110$, $d = -0.478$) manipulation time. Compared to *Control*, *VR Mover*'s paired test shows no significance for both coarse ($p = 1.00$, $d = 0.0879$) and fine ($p = 1.00$, $d = 0.0728$) manipulation, although the test with *Voice Command* shows a significant difference for coarse ($p = 0.0102$, $d = -0.423$) and fine ($p = 0.0110$, $d = -0.478$) manipulation. This indicates that *VR Mover* does not improve the efficiency for single-object mid-air movement.

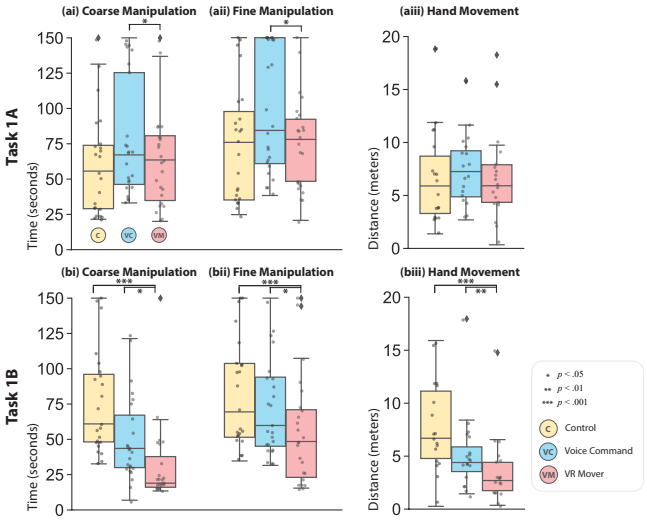

Figure 12: The upper row shows Task 1A's (ai) coarse manipulation time, (aii) fine manipulation time and (aiii) hand movement distance. The lower row shows Task 1B's (bi) coarse manipulation time, (bii) fine manipulation time, and (biii) hand movement distance. For each group, the left is Control, the middle is Voice Command and the right is VR Mover.

For Task 1B, the user needs to fit multiple objects to on-the-ground targets. The coarse manipulation time (Figure 12*bi*) for *Control*, *Voice Command* and *VR Mover* are 72.072 (28.862), 50.543 (29.532) and 29.852 (23.557), respectively, and the fine manipulation time (Figure 12*bii*) is 78.456 (30.067), 69.988 (29.652) and 52.636 (32.361). There are main effects for both coarse ($F(2, 23) = 21.583$, $p < 0.001$, $\eta_p^2 = 0.450$) and fine ($F(2, 23) = 8.769$, $p = 0.0125$, $\eta_p^2 = 0.183$) manipulation time. For coarse manipulation time, *VR Mover*'s paired tests show significant difference with both *Voice Command* ($p = 0.0130$, $d = -0.774$) and *Control* technique ($p < 0.001$, $d = -1.603$). Similarly, for fine manipulation time, *VR Mover* shows significance compared with both *Voice Command* ($p = 0.0342$, $d = -0.556$) and *Control* ($p < 0.005$, $d = -0.827$).

7.2 Hand Movement Distance (Task 1)

The total accumulated hand movement is also measured for Task 1A and 1B. Note that hand movement is recorded before the user has reached the fine target goal. For Task 1A, the mean of hand movement distances (Figure 12*aiii*) for *Control*, *Voice Command* and *VR Mover* are 6.400 (3.253), 6.367 (2.383) and 6.533 (3.305), respectively. There is no main effect when different techniques are used ($F(2, 23) = 1.083$, $p = 0.582$, $\eta_p^2 = 0.0226$). Thus, there is no need to show the paired tests.

On the other hand, for Task 1B, the hand movement distance is more differentiating (Figure 12*biii*). The mean movement distances are 7.503 (3.405), 5.148 (3.101) and 3.550 (2.740) for *Control*, *Voice Command* and *VR Mover*, respectively, and there is a main effect for technique ($F(2, 23) = 22.750$, $p < 0.001$, $\eta_p^2 = 0.474$). Compared to *Voice Command* and Control, *VR Mover* shows statistical significance. Furthermore, paired tests show that the hand movement distance in *VR Mover* is significantly different from that of *Voice Command* ($p < 0.005$, $d = -0.546$) and *Control* ($p < 0.001$, $d = -1.279$).

7.3 Arm Fatigue

Arm fatigue is measured by Borg C10. The average Borg C10s scores (Figure 13*a*) for *Control*, *Voice Command* and *VR Mover* are 4.792 (2.380), 3.521 (1.857) and 2.521 (1.461), respectively. There is a significant main effect for technique ($F(2, 23) = 20.840$, $p < 0.001$, $\eta_p^2 = 0.434$) and similarly, paired tests that compared *VR Mover* with the other two techniques, *Voice Command* ($p = 0.0197$, $d = -0.599$) and *Control* also show significance ($p < 0.001$, $d = -1.150$).

7.4 Workload

The mean NASA-TLX scores (Figure 14*a*), for *Control* group, *Voice Command* and *VR Mover*, are 5.965 (1.588), 4.729 (1.374) and 3.882 (1.797), respectively. There is a significant main effect for the technique ($F(2, 23) = 17.725$, $p < 0.005$, $\eta_p^2 = 0.198$). VR Mover's comparison with both *Voice Command* ($p = 0.0409$, $d = -0.530$) and *Control* group ($p < 0.001$, $d = -1.228$) both show statistical significance. To further investigate the constituent components of the users' workload, the result for each subscale is also presented. For mental demand (Figure 14*bi*), the scores for *Control*, *Voice Command* and *VR Mover* are 6.208 (2.121), 5.750 (1.984) and

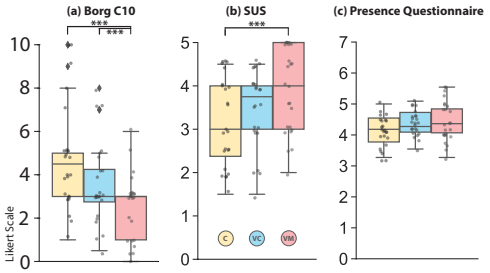

Figure 13: The (a) borg C10, (b) SUS, (c) presence results.

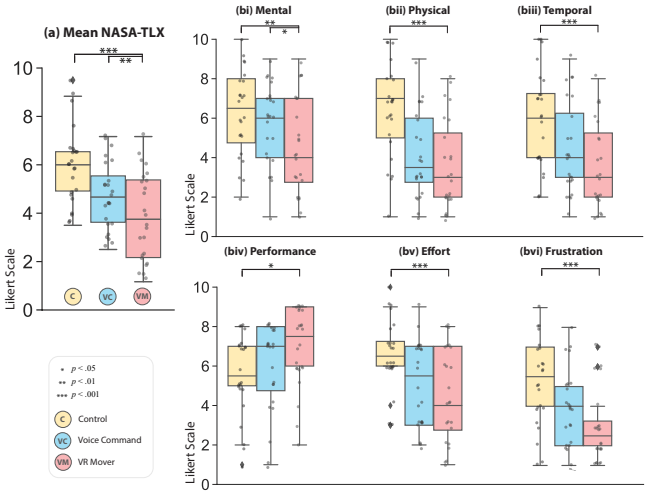

Figure 14: The (a) mean score of NASA-TLX and the subscales, (bi) mental demand, (bii) physical demand, (biii) temporal demand (biv) performance, (bv) effort, and (bvi) frustration.

4.542 (2.398), respectively, and it has a main effect ($F(2, 23) = 7.238$, $p = 0.0268$, $\eta_p^2 = 0.151$). *VR Mover* shows significance compared to both *Voice Command* ($p = 0.0478$, $d = -0.549$) and *Control* ($p < 0.01$, $d = -0.736$) in the paired tests. The scores for physical demand (Figure 14*bii*) are 6.375 (2.324), 4.125 (2.147) and 3.583 (2.253), and there is a main effect as well ($F(2, 23) = 18.025$, $p < 0.001$, $\eta_p^2 = 0.376$). The paired tests show that *VR Mover* has significance when compared with *Control* for physical demand ($p < 0.001$, $d = -1.220$), but not with *Voice Command* ($p = 0.457$, $d = -0.246$). Similarly, there are main effects for the other sub-scales (temporal demand, performance, effort and frustration), and paired tests show that *VR Mover* has significance when compared with *Control*, but not with *Voice Command* (Figure 14*biii-bvi*). For their detail, please refer to Appendix *D.1*. Overall, the considerably lower NASA-TLX scores of *VR Mover* are a strong indication that it can reduce workload.

7.5 User Experience

The overall UEQ-S scores (Figure 15*a*) for *Control* group, *Voice Command* and *VR Mover*, are 3.922 (1.273), 4.990 (0.953) and 5.729 (1.127), respectively. There is a significant main effect for the technique ($F(2, 23) = 26.053$, $p < 0.005$, $\eta_p^2 = 0.278$). The paired tests show VR Mover's comparisons with both *Voice Command*

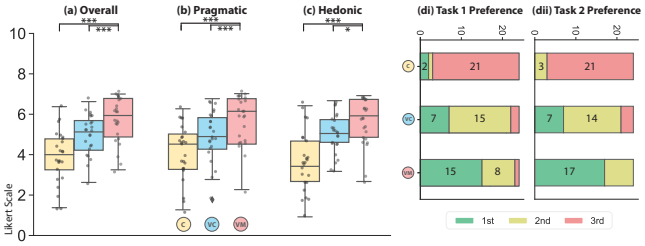

Figure 15: The left side is the (a) overall, (b) pragmatic, and (c) hedonic user experience results from UEQ-S. The right side is the preference ranking for (di) Task 1 and (dii) Task 2.

($p < 0.005$, $d = 0.699$) and *Control* group ($p < 0.001$, $d = 1.494$) have statistical significance. UEQ-S can be further separated into two meta-measures, practical and hedonic. The former indicates efficiency, perspicuity, and dependability while the latter indicates stimulation and novelty. For pragmatic quality (Figure 15*b*), the scores are 4.125 (1.342), 4.885 (1.235) and 5.719 (1.249), respectively. Similar to the overall score, there is a main effect ($F(2, 23) = 18.065$, $p < 0.001$, $\eta_p^2 = 0.376$) and the paired test shows significance when comparing *VR Mover* with *Voice Command* ($p < 0.005$, $d = 0.671$) and *Control* ($p < 0.005$, $d = 1.229$). For hedonic quality, the respective scores (Figure 15*c*) are 3.719 (1.485), 5.094 (0.935) and 5.719 (1.191), and it also has main effect ($F(2, 23) = 31.303$, $p < 0.005$, $\eta_p^2 = 0.652$). The paired tests that compared *VR Mover* with *Voice Command* ($p = 0.0183$, $d = 0.584$) and *Control* ($p < 0.005$, $d = 1.486$) both show significance.

7.6 Usability

The ease-of-use is measured by the SUS; the scores (Figure 13*b*), for *Control* group, *Voice Command* and *VR Mover*, are 3.167 (0.986), 3.479 (0.884) and 3.896 (0.957), respectively. There is a significant main effect for the technique ($F(2, 23) = 17.289$, $p < 0.005$, $\eta_p^2 = 0.360$). *VR Mover* only shows significance when compared to the *Control* ($p < 0.005$, $d = 0.750$) group. There is no significant difference with *Voice Command* ($p = 0.884$, $d = 0.452$).

7.7 Presence

Measured by the PQ, the average scores for presence (Figure 13*c*) are 4.121 (0.496), 4.398 (0.379) and 4.447 (0.624), for *VR Mover*, *Voice Command* and *Control* group, respectively. There is a significant main effect for the technique ($F(2, 23) = 4.748$, $p = 0.0134$, $\eta_p^2 = 0.0459$). However, there is no significance when comparing *VR Mover* with either *Voice Command* ($p = 0.598$, $d = 0.0954$) and *Control* ($p = 0.0651$, $d = 0.578$) in a paired test. Still, it is believed that given the existence of the main effect, the paired test comparing *VR Mover* shows the marginal significance and that the effect size is considerable, further investigation should be considered.

7.8 Ranking

At the end of the experiment, the users were asked to rank *Control*, *Voice Command* and *VR Mover* for both Task 1 (Figure 15*di*) and Task 2 (Figure 15*dii*). For Task 1's first choice, 2, 7, and 15 participants picked *Control*, *Voice Command* and VR Mover, respectively.

For Task 2, For Task 2, 17 participants picked *VR Mover* as their first choice while the remaining 7 picked *Voice Command*. Thus, generally, *VR Mover* is favoured by most participants.

7.9 Qualitative Feedback

In addition, we have recorded the audio in the structured interview. The interview records were transcribed for coding. The qualitative feedback from the users will be presented later in the discussion.

8 Discussion

Inspired by how humans convey spatial manipulation in the real world and empowered by the intelligence of an LLM, *VR Mover* aims to replicate this behaviour in VR. The *VR Mover* can listen to the user's instructions and take note of their gestural cues (via pointing and lining) to determine how to assist with coarse placement.

8.1 Performance

In physical reality, it is often advantageous when others can help you move objects, especially in the case of multiple objects. *VR Mover* provides an efficient interface to indicate several objects and their respective targets, which may be attributed separately per object. As such, *VR Mover* is expected to complete multi-object manipulation within a shorter time window. Some of the users (N=4) have commented that *VR Mover* is convenient: "I can manipulate multiple objects at a time, but others have to do it one by one. It is more convenient and quicker." (P18)

Objectively, the expectation and feedback that *VR Mover* is more convenient and efficient for multi-object placement is reflected in the manipulation time result of Task 1B; there, we can observe a significant difference between *VR Mover* and the other two interfaces, *Voice Command* and *Control*. For coarse placement, *VR Mover* can reduce at least 40% of the average time when compared to *Voice Command* and *Control*. However, its weaker effect on fine manipulation time suggests that *VR Mover* may not significantly benefit the fine-tuning process. This is expected as currently it is not embedded with a method for fine-tuning the placement of objects. Still, the result shows that the more efficient coarse placement can help the user complete the task faster.

On the other hand, it should be noted that *VR Mover* will not particularly benefit in a single mid-air object scenario, as a larger portion of the manipulation requires the classic gizmos + virtual hand combination to rotate and vertically move in order to fit the object to the final mid-air target. However, it can be argued that mid-air placement is not necessarily common in an object manipulation scenario (e.g. the sandbox environment in Task 2). Regardless, it can be inferred that *VR Mover* is best at supporting user performance when multiple objects are involved.

8.2 Arm Fatigue

When moving an object, a typical object manipulation interface will require a user to complete all necessary actions in the manipulation process. On the other hand, *VR Mover* can help with coarse placement, and then the user can take over to complete the fine-tuning. For situations where there are multiple objects, users can perform asynchronous multi-object manipulation. In addition, it should be noted that instructing *VR Mover* often involves a simple pointing gesture, which should require relatively little effort from the user. As some users (N=3) have pointed out, *VR Mover* is, "more relaxing, like having a conversation."(P20)

The fact that *VR Mover* induces less physical strain on the user is observed in several instances. Foremost, we can see that *VR Mover* has significantly less total hand movement for Task 1B (which involves multiple objects) when compared to *Control*. When compared to *Voice Command*, which also supports pointing, this significant reduction is likewise clearly visible. This can be explained by the fact that *VR Mover* is better at handling multi-object manipulation. Further, Borg C10, which measures user exertion, is observed to have a significant difference for *VR Mover* when compared to *Voice Command* and *Control*. The physical subscale of NASA-TLX also shows similar significance; this indicates that aside from the hand movement result, the users themselves also feel that the physical workload is reduced when using *VR Mover*.

Similar to our discussion on performance, it should be noted that when the manipulation involves less of the *VR Mover*, the reduction of fatigue will, in turn, be reduced. This expectation can be reflected by the hand moving distance in Task 1A where *VR Mover* group is virtually similar to the *Control* group. However, as indicated by the Borg C10 scores and the physical NASA-TLX subscale which takes into account both Task 1 and Task 2, we can see that, generally, the *VR Mover* should reduce the user's arm fatigue. It should be further noted that fatigue may be perceived differently by different users. Some users may not consider the tasks to be particularly fatiguing. However, even in that case, the interface's level of comfort should also be considered, and the result discussed here indicates that *VR Mover*, aside from reducing fatigue, can also improve the level of perceived comfort for those users.

8.3 Workload

Workload is an important consideration: if the workload is reduced, the user will be able to use the interface for a longer period. As reported, the mean NASA-TLX score of *VR Mover* significantly lowered compared to *Voice Command* and *Control*. This result shows that embedding the LLM to the interface can significantly reduce workload overall. We have already mentioned the physical workload when discussing the reduced arm fatigue of *VR Mover*; in the following, we will discuss other subscales in NASA-TLX.

Foremost, we can observe that the mental demand of *VR Mover* is significantly lower, compared to *Voice Command* and *Control*. It is believed that this lower mental demand is linked to the asynchronous multi-object manipulation feature and coarse-to-fine design of *VR Mover*. As discussed earlier, a human's visual working memory tends to group objects to process and manipulate visual information in a coarse-to-fine manner. By allowing users to manipulate multiple objects together and place them first in a general placement and then later fine-tune it, *VR Mover* fits the cognitive process of a human's visual working memory. Thus, the lower mental demand and the considerable effect size validate this belief. When compared to *Voice Command*, another important advantage of *VR Mover* is that the user need not memorise the grammar structure and keywords. This is reflected by several users (N=6) with comments such as

"(*VR Mover*) is the best: no grammar requirement and quick, convenient"(P23) and that *Voice Command* "must use a specific word,"(P6) and "need to remember the syntax"(P7).

For other subscales (temporal, performance, effort, and frustration), *VR Mover* has significance when compared to *Control* but not *Voice Command*. Overall, the existing evidence suggests that an LLM-based interface can enhance a traditional object manipulation interface by reducing the workload.

8.4 User Experience

User experience is an important aspect of any interface [41]. As shown in the result section, *VR Mover* achieves a significantly higher overall score in UEQ-S. In addition, *VR Mover* also has a higher score for hedonic and practical measures; this may indicate that a design that echoes how we convey spatial manipulation in real life is both enjoyable and practical. Some of the users have commented that the interface is "very fun to play with", "very interesting" and "amazing". The preference ranking of *VR Mover* is also encouraging for an LLM-based interface. For both Task 1 and Task 2, the majority of the users picked *VR Mover* as the first choice. Together, the result shows that *VR Mover* is a competitive interface for object manipulation. A few users (N=3) also commented on the freedom they felt while using the interface: "Users have more freedom in expressing themselves."(P20)

8.5 Naturalness

A natural user interface should be intuitive to use; that is, the user is able to quickly tell how to interact via the interface [46]. This kind of interface is ideal because the user can dedicate less time to the "entry-level" learning curve. As mentioned before, how *VR Mover* aims to achieve this is to mimic a real mover that can understand natural spatial manipulation communication. Qualitatively, it seems that we have achieved our goal, some of the users (N=3) have commented on the intuitiveness of *VR Mover* "It is intuitive to instruct the system, I tell it what I want and it will automatically do it for me"(P18), and that it feels like a conversation (N=3), "It's like talking to a person, which makes it easy to use."(P20) The SUS score also seems to reflect this as *VR Mover* is rated significantly higher compared to *Control*. The two-item SUS is particularly suitable for reporting ease of use as its two questions are "I thought this software technique was easy to use" and "I found the software very cumbersome/awkward to use". Inversely, one user commented that the *Control* technique has a "steep learning curve."(P10)

It was hoped that an intuitive interface would in turn improve presence as controller transparency impacts presence [12]. However, in our experiment, *VR Mover* did not show significance when compared with both *Voice Command* and *Control*. It is believed that this could be caused by the design that the user needs to press A when "pointing". Thus, the user still needs to remember to press the button to signal the selection point or area of interest, interrupting the feeling of presence. We believe the result does, however, inversely show that an interface involving speech or LLM will not reduce immersion and, subsequently, the feeling of presence.

8.6 Interactive Behaviour

As discussed, one of our beliefs is that users would prefer to follow a coarse-to-fine process for object manipulation. We have observed that this is indeed quite a common behaviour. Some of the users (N=4) have explicitly described this process during the interview, "When you ask someone to do the thing for you, afterwards, you may still want to adjust it by yourself."(P10) "(I) do the rough work in (*VR Mover*) then switch to hand for precise control."(P6) "(*VR Mover*) is the best. Firstly, use it to put things into a rough position, then use hand control for fine-tuning."(P13)

8.7 Summary

Here, we summarise the findings above. We can identify several ***benefits and drawbacks of an LLM-based interface compared to classic interfaces*** (objective **R3**). Foremost, it can be seen that compared to a classic interface (*Control*), the LLM-based interface (*VR Mover*) is able to achieve a **lower mental workload**. This benefit can be attributed to the fact that the LLM interface's ability to replicate real-world interactions and therefore is **natural** for the user to use. On the other hand, classic interfaces often require the user to more or less memorise abstract 3D representation (e.g. gizmos) and buttons' configuration, which can be challenging and take time to adapt. It has been known that abstraction can be detrimental to the naturalness of an interface [46]. Further, we believe the LLM interface is also more **intuitive**, as the results have shown it offers **better ease of use**. Users have explicitly commented that the interface feels intuitive and requires only natural instructions to achieve the desired outcome. Thus, it is expected that users are able to take much less time to understand how to operate compared to classic interfaces. In addition, the lower mental workload, higher usability, and naturalness of an LLM-based interface can also lead to better **learnability**, thereby lowering the **entry level** and making VR applications easier to operate. However, a notable drawback of utilizing an LLM is the **delay in response**. While our approach achieves a response time of ~2 seconds and ongoing LLM advancements are expected to improve processing speed, the user must still anticipate the inherent delay associated with running an LLM. In addition, with LLM, it is also **difficult to perform fine manipulation**. It is relatively suboptimal to use pointing and speech to specify a very particular positional and rotational transformation. However, we find that users are mostly concerned with coarse placement, as suggested in our earlier theoretical discussion. Regardless, in general, despite some of the drawbacks, the higher usability and intuitive control, along with other benefits, can result in an **overall better user experience** for an LLM interface.

Another important question is what exactly is the ***benefit and necessity of an LLM*** (objective **R4**). To answer this question, we have explicitly introduced a rule-based variant (*Voice Command*) in the user study to compare with the LLM interface. This pair of interfaces can isolate the LLM as a factor: they both allow the user to use speech and multimodal signals (e.g. pointing) for manipulating, while the only major difference is that *Voice Command* uses a rule-based algorithm to determine the action, whereas *VR Mover* uses the LLM. Echoing the discussion above, one of the major benefits of introducing an LLM is **lower mental workload**. This can be attributed to the fact that with an LLM-removed variant, the user

will need to memorise the command and grammar of the intended manipulation. In contrast, given an LLM, the user **need not memorise command words and grammar**, they can simply **naturally converse** to describe their goal, and then the **LLM infers the intent** of the user. This is reflected not only in the mental workload score but also explicitly by the users, who report that they can speak naturally without a heavy burden on their memory. In addition, the LLM is also an important component in **driving multi-object manipulation**. It seems that typical interfaces or voice command interfaces can only uniformly perform manipulation on multiple objects. Along with the ability to **conveniently refer to multiple objects** at once, the LLM interface also **enables asynchronous manipulations**. As an LLM-based interface is more optimal for the manipulation of multiple objects, ultimately, it allows the user to "do less for more". Thus, it can also be seen that the LLM will lead to **lower fatigue** and **lower overall workload**, even when compared to the LLM-absent variant. Furthermore, the intelligence offered by the LLM also **unlocks flexible and novel manipulations** (e.g. undo feature). These benefits together show a strong impact in **improving user experience**, which indicates that an LLM-based interface can improve VR experience.

9 Design Recommendation

Driven by the results presented, we derived a set of design implications for future LLM-based interfaces to address **R5** (subsection 9.1). Then, based on the design findings, we further develop some explorative interactions to showcase how some of the design implications can be implemented (subsection 9.2).

9.1 Design Implications

(D1) **Leverage both LLM and classic interfaces:** While an LLM-based interface can enable natural interaction, the fact that it may rely on speech and multi-modal signals (e.g. pointing) can make it less optimal for precise control. Thus, we suggest that it should be mixed with a classic interface as a complementary control method for fine-tuning. We further recommend that the LLM-based and classic interface should be seamlessly switched and integrated. For example, after the user has asked the LLM to move an object, the moving gizmos from the classical interface can pop-up immediately afterwards to enable the user to quickly adjust afterwards. Another possible direction is that the LLM can directly control the classic interface based on user instruction, thereby blurring the distinction between the LLM-based and classic interface, and reaching a middle ground between coarse and fine control.

(D2) **Coarse placement:** Although this may depend on the specific scenario or the user's preference, our experiment suggests that most users are not particularly concerned with fine control for manipulation. Instead, they generally focus more on ensuring that object placements are roughly where they want them to be. Thus, an implication is that the LLM interface should prioritize optimizing effective coarse control rather than refining its fine control capacity.

(D3) **Focus on asynchronous multi-object interaction:** While an LLM-based interface can perform single-object manipulation similarly to traditional methods, its main strength lies in multiple-object manipulation. The LLM intelligence enables contextual awareness, filtering, and a deeper understanding of the manipulations and objects that are involved. As such, future interfaces should explore how the LLM intelligence can be leveraged for more complex, asynchronous multi-target operations.

(D4) **Develop an optimal set of atomic functions:** Previous LLM research focused on code generation to achieve editing. Although valid, it leads to errors and longer processing times (>15 seconds). In contrast, as discussed in subsection 5.5, our approach utilises a set of functions to achieve our advanced manipulation goals while effectively mitigating errors and significantly reducing response times. We believe this implies that an LLM-based interface should focus on the design of an optimal atomic function set, thus enabling the LLM to address user requests optimally.

(D5) **Interaction discovery and tuning:** As mentioned, an LLM-based interface can make use of atomic functions to achieve a broader manipulation goal. There are many ways for the LLM to conjugate manipulation methods — further experimentation is necessary to fully explore the LLM-enabled interaction space. However, the effectiveness of some experimental interactions may be limited or relatively unreliable. If such an interaction is deemed important and beneficial, it can be easily "taught" to the LLM by providing a singular exemplar case within the prompt. Although this prompt engineering modification still requires some preparation, it shows that an LLM-based interface is much easier to alter than a classic interface, which requires programming.

(D6) **Mimic real-world behaviour:** Another important finding is that replicating interactions in reality is an effective design strategy to generate natural interactions in VR. This aligns with previous research on natural hardware interfaces, which suggests that, so long as the user is able to transfer their embodied knowledge from a classic interface (e.g. steering wheel) to a novel one that replicates the former, they will be able to naturally use it without much extra training [46]. Our work here shows that this transfer of embodied knowledge is not limited to tangible interfaces but also to interactions (i.e. requesting a mover to assist with object movement). It is worth considering integrating an LLM-based interface that replicates real-world interactions to reduce the learning curve and enhance usability in VR applications.

9.2 Explorative Interactions

Based on the design implications just discussed, this section explores interactions that we believe are particularly interesting and can further showcase the application potential of *VR Mover*. Additional examples and presentations of these explorative interactions are provided in Appendix *A.2*. They have all been implemented and were used for a garden scene editing scenario (see Appendix *A.3*).

• **Asynchronous Multi-object Undo:** First, we explore design implication D3. Aside from the undo operation presented earlier,

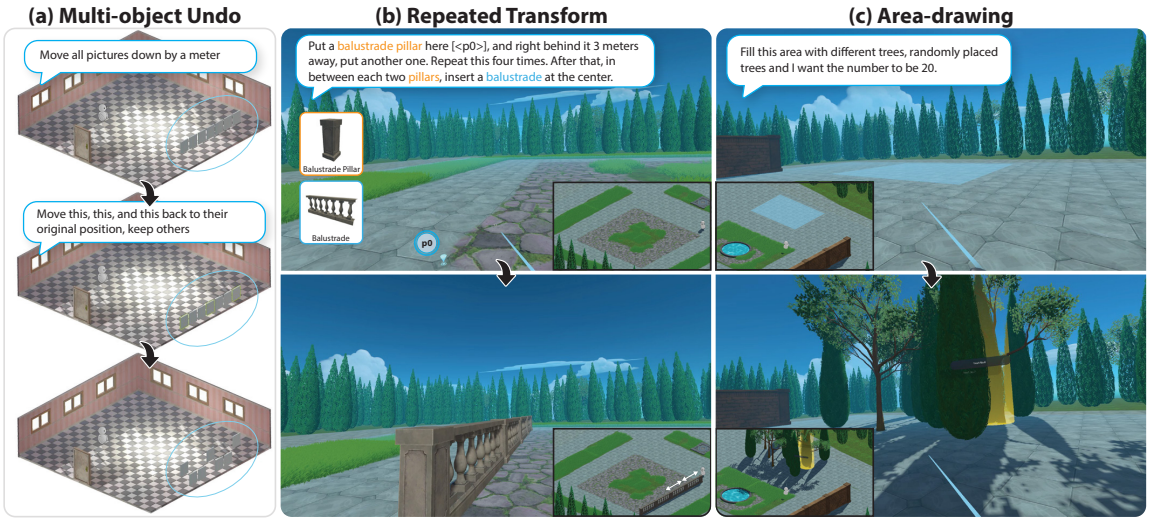

Figure 16: *VR Mover* can also enable (a) asynchronous multi-object undo, (b) repeated transform and (c) area-drawing.

we also show that it is possible to perform asynchronous multi-object undo when we reverse some (not all) objects' manipulation (Figure 16*a*).

• **Repeated Transform:** Then, we experiment with some novel interactions based on D5. Reminiscent of 3D modelling software (e.g. Blender), it is possible to apply a repetitive transform to build patterns (Figure 16*b*).

• **Area-drawing:** We have shown how points and lines can be used for multimodal signalling, but other formats are possible as well. Noting that users prefer coarse placement (D2) and that people tend to draw an area to indicate an area-of-interest (D6), in Figure 16*c*, we show that the user can draw an area to indicate where to manipulate objects.

• **Filtered Selection:** Last, we further explore more interactions (D5). It is also possible to specify an area and then only select some of the objects in the area for manipulation. Assuming there is an area with yellow flowers, red flowers and bushes, the user can say "I only want the yellow flowers in this area to be larger."

10 Limitations and Future Works

VR Mover has several limitations. For one, it is currently not optimal for mid-air object manipulation as it is difficult to specify where to place an object in mid-air. It is possible with lining, but more effective strategies could be investigated. In addition, improving LLM-empowered natural interfaces for precise manipulations could be a future direction. Second, pointing requires the user to press the A button to indicate, while in daily life, people tend to use fingers to refer to points of interest without extra actions. A more faithful pointing interface may improve user experience and results. Third, currently, we use bounding boxes to describe the general shape of objects; this limits the model's ability to handle concave objects or objects with complex geometry. Currently, visual prompting is not used for *VR Mover* as it hinders response time, but future work could consider it to improve the interface's performance. Lastly, we did not incorporate gaze integration in our work. The integration of additional signals, such as gaze, visual cues, and auditory inputs, however, may open new avenues to enhance the overall interaction experience and thus warrants further investigation.

11 Conclusion

Inspired both by how movers, in reality, assist us with moving objects and that we have a natural capacity to convey spatial manipulation instructions to each other, we propose *VR Mover*, an LLM-based interface that supports object manipulation. Previous interfaces generally require users to perform all actions to reach a specific outcome; on the other hand, *VR Mover* uses a user- and spatial-aware LLM to support object manipulation tasks based on multimodal user instruction. To realise *VR Mover*, we proposed an atomic object manipulation function set to facilitate the quick generation of manipulation commands and a user-centric augmentation module that enables quick integration of multimodal signals, including user perception and gestural cues.

To the best of our knowledge, we are the first to conduct a user study that compares an LLM-based interface to classic interfaces. Results show that compared to gizmos-only and virtual hand control, *VR Mover* is more performant in multi-object manipulation tasks, has better user experience and usability, and less workload and fatigue. A similar result is shown when comparing the *Voice Command* LLM-removed variant, which shows the importance of an LLM that can understand natural user instructions. The LLM can allow the user to express their manipulation requests more freely. At the same time, it also enables the LLM to complete manipulation tasks that are beyond the original scope. We believe our interface's better usability and less workload can particularly benefit novice users. We also envision that people with limited hand motion ability may particularly benefit from VRMover as well. In addition, the interface can also generally help (advanced) users with complex tasks and use cases. The last part of our work provides design implications for future LLM-based interfaces; we suggest

future research to focus on complex operations that cannot be intuitively controlled by a classic interface or explicitly implemented via a rule-based approach, and enables the user to quickly switch between an LLM-based interface with a complementary interface for fine-tuning.

Acknowledgments

The authors thank Hong-Va Leong and Pangjing Wu for their useful discussions and feedback.

References

[1] Setareh Aghel Manesh, Tianyi Zhang, Yuki Onishi, Kotaro Hara, Scott Bateman, Jiannan Li, and Anthony Tang. 2024. How People Prompt Generative AI to Create Interactive VR Scenes. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference (DIS '24)*. 2319–2340.

[2] Elizabeth E Austin and Naomi Sweller. 2014. Presentation and production: The role of gesture in spatial communication. *Journal of experimental child psychology* 122 (2014), 92–103.

[3] Edward Awh, Brian Barton, and Edward K. Vogel. 2007. Visual Working Memory Represents a Fixed Number of Items Regardless of Complexity. *Psychological Science* 18, 7 (2007), 622–628.

[4] Farkhandah Aziz, Chris Creed, Sayan Sarcar, Maite Frutos-Pascual, and Ian Williams. 2022. Voice Snapping: Inclusive Speech Interaction Techniques for Creative Object Manipulation. In *Proceedings of the 2022 ACM Designing Interactive Systems Conference (DIS '22)*. 1486–1496.

[5] Alan Baddeley. 2003. Working memory: looking back and looking forward. *Nature reviews neuroscience* 4, 10 (2003), 829–839.

[6] Yiwei Bao, Jiaxi Wang, Zhimin Wang, and Feng Lu. 2023. Exploring 3D Interaction with Gaze Guidance in Augmented Reality. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR '23)*. 22–32.

[7] Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, et al. 2022. GAUDI: A Neural Architect for Immersive 3D Scene Generation. *Advances in Neural Information Processing Systems* 35 (2022), 25102–25116.

[8] Rojin Bayat, Elios De Maio, Jacopo Fiorenza, Massimo Migliorini, and Fabrizio Lamberti. 2024. Exploring Methodologies to Create a Unified VR User-Experience in the Field of Virtual Museum Experiences. In *2024 IEEE Gaming, Entertainment, and Media Conference*. 1–4.

[9] Richard A. Bolt. 1980. "Put-that-there": Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '80)*. 262–270.

[10] Gunnar A Borg. 1982. Psychophysical bases of perceived exertion. *Medicine and science in sports and exercise* 14, 5 (1982), 377–381.

[11] J Brooke. 1996. SUS: A quick and dirty usability scale. *Usability Evaluation in Industry* (1996).

[12] Erin A. Carroll, Celine Latulipe, Richard Fung, and Michael Terry. 2009. Creativity factor evaluation: towards a standardized survey metric for creativity support. In *Proceedings of the Seventh ACM Conference on Creativity and Cognition*. 127–136.

[13] Alan Y Cheng, Meng Guo, Melissa Ran, Arpit Ranasaria, Arjun Sharma, Anthony Xie, Khuyen N Le, Bala Vinaithirthan, Shihe Luan, David Thomas Henry Wright, et al. 2024. Scientific and Fantastical: Creating Immersive, Culturally Relevant Learning Experiences with Augmented Reality and Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. 1–23.

[14] Sharon Rose Clay and Jane Wilhelms. 1996. Put: Language-based interactive manipulation of objects. *IEEE Computer Graphics and Applications* 16, 2 (1996), 31–39.

[15] Tor-Salve Dalsgaard, Jarrod Knibbe, and Joanna Bergström. 2021. Modeling Pointing for 3D Target Selection in VR. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology (VRST '21)*. 1–10.

[16] Fernanda De La Torre, Cathy Mengying Fang, Han Huang, Andrzej Banburski-Fahey, Judith Amores Fernandez, and Jaron Lanier. 2024. LLMR: Real-time Prompting of Interactive Worlds using Large Language Models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. Article 600, 22 pages.

[17] Giuseppe Desolda, Andrea Esposito, Florian Müller, and Sebastian Feger. 2023. Digital Modeling for Everyone: Exploring How Novices Approach Voice-Based 3D Modeling. In *IFIP Conference on Human-Computer Interaction*. Springer, 133–155.

[18] Tobias Drey, Michael Montag, Andrea Vogt, Nico Rixen, Tina Seufert, Steffi Zander, Michael Rietzler, and Enrico Rukzio. 2023. Investigating the Effects of Individual Spatial Abilities on Virtual Reality Object Manipulation. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. 1–24.

[19] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2024. LayoutGPT: Compositional Visual Planning and Generation with Large Language Models. *Advances in Neural Information Processing Systems* 36 (2024).

[20] Jorge Askur Vazquez Fernandez, Jae Joong Lee, Santiago Andrés Serrano Vacca, Alejandra Magana, Bedrich Benes, and Voicu Popescu. 2024. Hands-Free VR. *arXiv preprint arXiv:2402.15083* (2024).

[21] Rachel L. Franz, Sasa Junuzovic, and Martez Mott. 2024. A Virtual Reality Scene Taxonomy: Identifying and Designing Accessible Scene-Viewing Techniques. *ACM Trans. Comput.-Hum. Interact.* 31, 2, Article 23 (2024), 44 pages.

[22] Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. 2024. AnyHome: Open-Vocabulary Generation of Structured and Textured 3D Homes. In *European Conference on Computer Vision (ECCV '24)*. 52–70.

[23] Zaifeng Gao and Shlomo Bentin. 2011. Coarse-to-fine encoding of spatial frequency information into visual short-term memory for faces but impartial decay. *Journal of Experimental Psychology: Human Perception and Performance* 37, 4 (2011), 1051.

[24] Zaifeng Gao, Xiaowei Ding, Tong Yang, Junying Liang, and Rende Shui. 2013. Coarse-to-fine construction for high-resolution representation in visual working memory. *PLoS One* 8, 2 (2013), e57913.

[25] Daniele Giunchi, Nels Numan, Elia Gatti, and Anthony Steed. 2024. DreamCodeVR: Towards Democratizing Behavior Design in Virtual Reality with Speech-Driven Programming. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR '24)*. 579–589.

[26] P Christopher Gloumeau, Wolfgang Stuerzlinger, and JungHyun Han. 2020. PinNPivot: Object Manipulation Using Pins in Immersive Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics* 27, 4 (2020), 2488–2494.

[27] Barbara Göbl, Simone Kriglstein, and Helmut Hlavacs. 2021. Conversational Interfaces in Serious Games: Identifying Potentials and Future Research Directions based on a Systematic Literature Review. *CSEDU (1)* (2021), 108–115.

[28] Ankur Handa, Viorica Pătrăucean, Simon Stent, and Roberto Cipolla. 2016. Scenenet: An annotated model generator for indoor scene understanding. In *2016 IEEE International Conference on Robotics and Automation*. 5737–5743.

[29] Jeffrey T Hansberger, Chao Peng, Shannon L Mathis, Vaidyanath Areyur Shanthakumar, Sarah C Meacham, Lizhou Cao, and Victoria R Blakely. 2017. Dispelling the gorilla arm syndrome: the viability of prolonged gesture interactions. In *Virtual, Augmented and Mixed Reality: 9th International Conference, Held as Part of HCI International 2017*. Springer, 505–520.

[30] Sandra G Hart. 2006. NASA-task load index (NASA-TLX); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 50. 904–908.

[31] Daniel Hepperle, Yannick Weiß, Andreas Siess, and Matthias Wölfel. 2019. 2D, 3D or speech? A case study on which user interface is preferable for what kind of object interaction in immersive virtual reality. *Computers & Graphics* 82 (2019), 321–331.

[32] Kenneth P. Herndon, Andries van Dam, and Michael Gleicher. 1994. The challenges of 3D interaction: a CHI '94 workshop. *SIGCHI Bull.* 26, 4 (Oct. 1994), 36–43.

[33] Jan Hombeck, Henrik Voigt, Timo Heggemann, Rabi R Datta, and Kai Lawonn. 2023. Tell Me Where To Go: Voice-Controlled Hands-Free Locomotion for Virtual Reality Systems. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR '23)*. 123–134.

[34] Xiyun Hu, Dizhi Ma, Fengming He, Zhengzhe Zhu, Shao-Kang Hsia, Chenfei Zhu, Ziyi Liu, and Karthik Ramani. 2025. GesPrompt: Leveraging Co-Speech Gestures to Augment LLM-Based Interaction in Virtual Reality. *arXiv preprint arXiv:2505.05441* (2025).

[35] Robert JK Jacob. 1990. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '90)*. 11–18.

[36] Sujin Jang, Wolfgang Stuerzlinger, Satyajit Ambike, and Karthik Ramani. 2017. Modeling Cumulative Arm Fatigue in Mid-Air Interaction based on Perceived Exertion and Kinetics of Arm Motion. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. 3328–3339.

[37] Kevin Sujith John, G Abin Roy, and PS Bindhya. 2024. LLM Based 3D Avatar Assistant. In *2024 1st International Conference on Trends in Engineering Systems and Technologies*. 1–5.

[38] Majeed Kazemitabaar, Xinying Hou, Austin Henley, Barbara Jane Ericson, David Weintrop, and Tovi Grossman. 2023. How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. 1–12.

[39] Daehwa Kim, Vimal Mollyn, and Chris Harrison. 2023. WorldPoint: Finger Pointing as a Rapid and Natural Trigger for In-the-Wild Mobile Interactions. *Proceedings of the ACM on Human-Computer Interaction* 7, ISS (2023), 357–375.

[40] David Kirk, Tom Rodden, and Danaë Stanton Fraser. 2007. Turn it this way: grounding collaborative action with remote gestures. In *Proceedings of the SIGCHI*

conference on Human Factors in Computing Systems (CHI '07). 1039–1048.
[41] Ahmet Baki Kocaballi, Liliana Laranjo, and Enrico Coiera. 2019. Understanding and measuring user experience in conversational interfaces. *Interacting with Computers* 31, 2 (2019), 192–207.
[42] Joseph J LaViola Jr, Ernst Kruijff, Ryan P McMahan, Doug Bowman, and Ivan P Poupyrev. 2017. *3D user interfaces: theory and practice*. Addison-Wesley Professional.
[43] Jaewook Lee, Jun Wang, Elizabeth Brown, Liam Chu, Sebastian S Rodriguez, and Jon E Froehlich. 2024. GazePointAR: A Context-Aware Multimodal Voice Assistant for Pronoun Disambiguation in Wearable Augmented Reality. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. 1–20.
[44] Xiaolong Liu, Lili Wang, Wei Ke, and Sio-Kei Im. 2024. Object manipulation based on the head manipulation space in VR. *International Journal of Human-Computer Studies* (2024), 103346.
[45] Robert H Logie. 2003. Spatial and visual working memory: A mental workspace. In *Psychology of Learning and Motivation*. Vol. 42. 37–78.
[46] Mitchell W McEwan, Alethea L Blackler, Daniel M Johnson, and Peta A Wyeth. 2014. Natural mapping and intuitive interaction in videogames. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play (CHI PLAY '14)*. 191–200.
[47] Daniel Mendes, Fabio Marco Caputo, Andrea Giachetti, Alfredo Ferreira, and Joaquim Jorge. 2019. A Survey on 3D Virtual Object Manipulation: From the Desktop to Immersive Virtual Environments. *Computer Graphics Forum* 38, 1, 21–45.
[48] Daniel Mendes, Maurício Sousa, Rodrigo Lorena, Alfredo Ferreira, and Joaquim Jorge. 2017. Using custom transformation axes for mid-air manipulation of 3D virtual objects. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology (VRST '17)*. 1–8.
[49] Meta. 2024. Llama 3.1. https://llama.meta.com/ Accessed: 2024-9-13.
[50] Microsoft. 2024. How to generate reproducible output with Azure OpenAI Service - Azure OpenAI. https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/reproducible-output Accessed: 2024-09-11.
[51] George A Miller. 1956. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review* 63, 2 (1956), 81.
[52] Elena Morotti, Lorenzo Donatiello, and Gustavo Marfia. 2020. Fostering fashion retail experiences through virtual reality and voice assistants. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW '20)*. 338–342.
[53] Chelsea Myers, Anushay Furqan, Jessica Nebolsky, Karina Caro, and Jichen Zhu. 2018. Patterns for How Users Overcome Obstacles in Voice User Interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. 1–7.
[54] Jakob Nielsen. 1994. *Usability Engineering*. Morgan Kaufmann Publishers Inc.
[55] Başak Melis Öcal, Maxim Tatarchenko, Sezer Karaoglu, and Theo Gevers. 2024. SceneTeller: Language-to-3D Scene Generation. *arXiv preprint arXiv:2407.20727* (2024).
[56] OpenAI. 2024. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/ Accessed: 2024-9-13.
[57] Jason Orlosky, Chang Liu, Kenya Sakamoto, Ludwig Sidenmark, and Adam Mansour. 2024. EyeShadows: Peripheral Virtual Copies for Rapid Gaze Selection and Interaction. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR '24)*. 681–689.
[58] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *Proceedings of the 9th annual ACM symposium on User interface software and technology (UIST '96)*. 79–80.
[59] Jesse Sargent, Stephen Dopkins, John Philbeck, and David Chichka. 2010. Chunking in spatial memory. *Journal of Experimental Psychology: Learning, memory, and cognition* 36, 3 (2010), 576.
[60] Jeff Sauro. 2018. *Can You Use a Single Item to Predict SUS Scores?* Retrieved Sep 13, 2024 from https://measuringu.com/single-item-sus/
[61] Philip J. Schneider and David Eberly. 2003. Intersection in 3D. In *Geometric Tools for Computer Graphics*. Morgan Kaufmann, 481–662.
[62] Martin Schrepp, Andreas Hinderks, et al. 2017. Design and evaluation of a short version of the user experience questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence* 4, 6 (2017), 103–108.
[63] Alon Shoa, Ramon Oliva, Mel Slater, and Doron Friedman. 2023. Sushi with Einstein: Enhancing Hybrid Live Events with LLM-Based Virtual Humans. In *Proceedings of the 23rd ACM International Conference on Intelligent Virtual Agents*. 1–6.
[64] Yasuko Sugase, Shigeru Yamane, Shoogo Ueno, and Kenji Kawano. 1999. Global and fine information coded by single neurons in the temporal visual cortex. *Nature* 400, 6747 (1999), 869–873.
[65] Hemant Bhaskar Surale, Aakar Gupta, Mark Hancock, and Daniel Vogel. 2019. TabletInVR: Exploring the Design Space for Using a Multi-Touch Tablet in Virtual Reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. 1–13.
[66] Mirko Thalmann, Alessandra S Souza, and Klaus Oberauer. 2019. How does chunking help working memory? *Journal of Experimental Psychology: Learning, Memory, and Cognition* 45, 1 (2019), 37.
[67] Leslie G Ungerleider, Susan M Courtney, and James V Haxby. 1998. A neural system for human visual working memory. *Proceedings of the National Academy of Sciences* 95, 3 (1998), 883–890.
[68] Hongyu Wan, Jinda Zhang, Abdulaziz Arif Suria, Bingsheng Yao, Dakuo Wang, Yvonne Coady, and Mirjana Prpa. 2024. Building LLM-based AI Agents in Social Virtual Reality. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '24)*. 1–7.
[69] Can Wang, Hongliang Zhong, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. 2024. Chat2Layout: Interactive 3D Furniture Layout with a Multimodal LLM. *arXiv preprint arXiv:2407.21333* (2024).
[70] Keru Wang, Zhu Wang, Ken Nakagaki, and Ken Perlin. 2024. "Push-That-There": Tabletop Multi-robot Object Manipulation via Multimodal 'Object-level Instruction'. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference (DIS '24)*. 2497–2513.
[71] Zhan Wang, Lin-Ping Yuan, Liangwei Wang, Bingchuan Jiang, and Wei Zeng. 2024. VirtuWander: Enhancing Multi-modal Interaction for Virtual Tour Guidance through Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. 1–20.
[72] Yuxi Wei, Zi Wang, Yifan Lu, Chenxin Xu, Changxing Liu, Hao Zhao, Siheng Chen, and Yanfeng Wang. 2024. Editable Scene Simulation for Autonomous Driving via Collaborative LLM-Agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '24)*. 15077–15087.
[73] Matt Whitlock, Ethan Harnner, Jed R Brubaker, Shaun Kane, and Danielle Albers Szafir. 2018. Interacting with Distant Objects in Augmented Reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR '18)*. 41–48.
[74] Bob G Witmer, Christian J Jerome, and Michael J Singer. 2005. The factor structure of the presence questionnaire. *Presence: Teleoperators & Virtual Environments* 14, 3 (2005), 298–312.
[75] Jackie Yang, Yingtian Shi, Yuhan Zhang, Karina Li, Daniel Wan Rosli, Anisha Jain, Shuning Zhang, Tianshi Li, James A Landay, and Monica S Lam. 2024. ReactGenie: A Development Framework for Complex Multimodal Interactions Using Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. 1–23.
[76] Chaoxiong Ye, Tengfei Liang, Yin Zhang, Qianru Xu, Yongjie Zhu, and Qiang Liu. 2020. The two-stage process in visual working memory consolidation. *Scientific reports* 10, 1 (2020), 13564.
[77] Difeng Yu, Xueshi Lu, Rongkai Shi, Hai-Ning Liang, Tilman Dingler, Eduardo Velloso, and Jorge Goncalves. 2021. Gaze-Supported 3D Object Manipulation in Virtual Reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. 1–13.
[78] Lei Zhang, Jin Pan, Jacob Gettig, Steve Oney, and Anhong Guo. 2024. VRCopilot: Authoring 3D Layouts with Generative AI Models in VR. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. 1–13.
[79] Xiaoyan Zhou, Adam Sinclair Williams, and Francisco Raul Ortega. 2022. Eliciting Multimodal Gesture+Speech Interactions in a Multi-Object Augmented Reality Environment. In *Proceedings of the 28th ACM Symposium on Virtual Reality Software and Technology (VRST '22)*. 1–10.