ImoGenXR: Investigating the Impact on Creativity Support in a Generative-AI Assisted Immersive Authoring Workflow

George Limbert
School of Computing and Communications
Lancaster University
Lancaster, UK
g.limbert@lancaster.ac.uk

Abhijit Karnik
School of Computing and Communications
Lancaster University
Lancaster, UK
a.karnik@lancaster.ac.uk

***Abstract*—Creating interactive extended reality experiences is currently a complex activity with a high entry barrier of developer skills. Immersive authoring and no/low code techniques are a promising direction towards democratizing access to novice users as they lower the entry barrier while users develop advanced skills. Similarly, generative-AI has shown potential for template code generation, 3D model creation and retrieval. In this paper, we present ImoGenXR, an immersive workflow integrated with Unity and scaffolded by existing generative-AI. Users can build scenes within the immersive environment by inserting 3D models using voice commands. They can also attach AI-generated code to invoke desired behaviours without leaving the immersive environment. We use ImoGenXR's workflow to explore the effect of creativity support using generative AI on an existing tool like Unity. The results show significant improvement in user experience for novice users. ImoGenXR shows the value of including creativity support through generative AI for existing XR development tools.**

***Index Terms*—Immersive authoring, Interaction design, Generative AI, User studies, Creativity Support**

I. Introduction

Virtual Reality (VR) and Augmented Reality (AR) experiences are becoming prevalent since general public are able to access content using relatively affordable head-mounted devices (HMDs). Users expect high levels of transformation in everyday interactions in domains like education, travel and retail through use of XR experiences. The current tools available for developing extended reality (XR) are powerful, but require a substantial degree of domain expertise, such as programming, 3D modeling and extensive knowledge of a specific development framework or toolkit. Specifically in the domain of education, these tools present a high barrier for entry for topic experts wishing to use the medium of XR experiences to scaffold pedagogical approaches [1]. Nebeling and Speicher highlight similar issues across a broader class of XR development tools [2]. The current limitation with most off-the-shelf XR development tools is that they do not allow the novice user to start creating simple interaction-rich XR experiences whilst the user builds up experience and expertise to fully leverage the power of the development tools. The development workflow of the tools can be versatile and powerful, but at the same time, intimidating and non-intuitive to a beginner.

Creativity support has been attempted through no/low code paradigm aiming to abstract the underlying programming language, or through alternative interfaces that help add interactivity to an XR experience [3], [4]. This helps users focus on the creative aspects without requiring the programming knowledge needed to achieve similar results through traditional programming methods. In XR experience development, the immersive approach is gaining traction. The immersive approach allows users to edit the environment immersed in the XR experience and without exiting to the development tool's default interface. This allows the user to avoid constant context switching between the development environment and testing which normally involves a HMD or a smart device. However, currently such systems have constraints and limitations to customisation that make them difficult to integrate into a creative task workflow.

This paper explores creativity support for users considered as novices of the XR development tools they wish to use as a supportive technology. The motivation for this is based in Schneiderman's argument that *"Supportive technologies can become the potter's wheel and mandolin of creativity"* [5]. Innovation in creativity support opens up new means for users to express their ideas, enabling them to create and disseminate ideas more freely. To this effect, we present ImoGenXR (Immersive object Generation in eXtended Reality), as a immersive no-code workflow that is integrated on top of an existing XR development tool. ImoGenXR allows users to build experiences by requesting custom object models, via a natural language voice interface, that can then be placed into the virtual environment. The same interface can also be used to dynamically generate and import code to be attached to any object without any further effort from the user. Imported code can either be run as a standalone action or combined with a trigger to form an interactive behaviour. Akin to the evolution of photographic slides to slideshow presentation

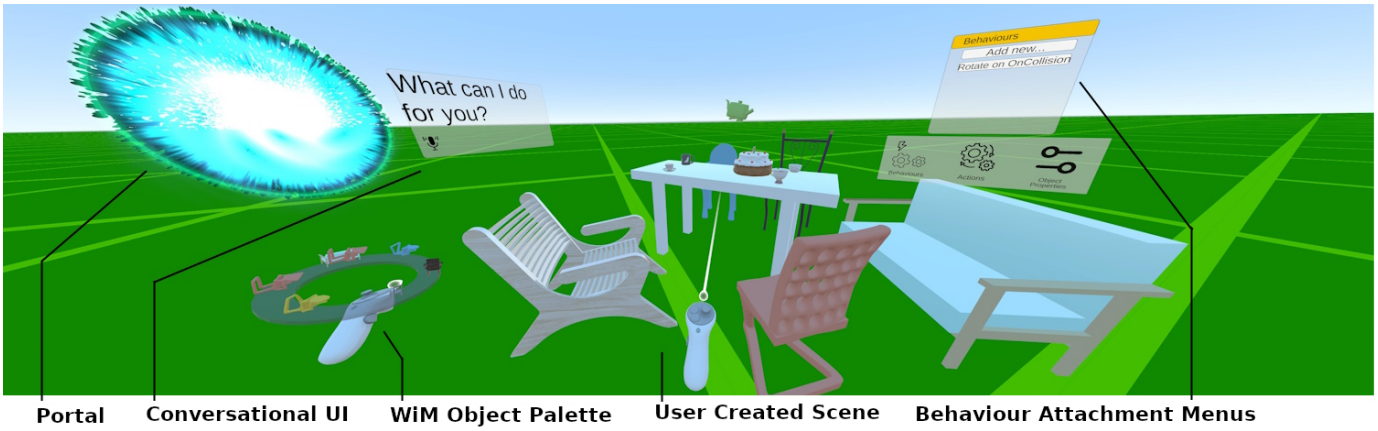

Fig. 1: User view of ImoGenXR's workflow showing the key visual elements of the system

programs, and multiplane cameras to 3D animation software, this work proposes to transform the current time and resource expensive means of XR content creation into an immersive no-code experience. ImoGenXR takes the recent advances of GenAI and existing XR toolkits to implement a system that supports novice users in immersively creating interaction-rich XR experiences. ImoGenXR contributes a proof-of-concept immersive authoring workflow that utilises existing development features provided by the base Unity engine. Our work builds on existing literature around immersive authoring tools, such as LLMR [6], to propose a workflow that enhances the user experience during such activities through AI-enabled creativity support.

II. Background

Below we discuss existing applications that enable virtual content authoring, as well as the current state of the art in GenAI relevant to the context of ImoGenXR.

A. Immersive Authoring Tools

Authoring tools can be defined as a development environment that facilitates the creation of new VR content through means such as reuse of assets [7]. These tools can be classified by skill and resources required, and fidelity provided [2]. Steed et al. were among the first to suggest a system which allowed immersive authoring of VR content through visual dataflow representations [8]. Immersive authoring as a concept was later formalised by Lee at al. [9] who define the approach as one which supports the development of a VR application whilst inside said application. Since, immersive authoring has only expanded as a focus of research [10], [11]. Such authoring tools are often limited by predefined sets of available objects or simplistic interactive behaviours. Various commercial products have also incorporated immersive techniques for VR authoring, such as Neos Metaverse[1] and Zoe [2] which allow end users to create simple interactive scenes immersively, and without writing any code. Currently, game engines such as Unity and Unreal are the most popular development environments used to create VR experiences, despite many developers encountering steep learning curves [12]. This work aims to contribute to the sixth category of authoring tools as identified by Nebeling and Speicher [2] by supporting users in immersively creating interactive VR experiences without requiring domain-specific expertise.

B. Visual Programming

Early visual programming languages were created as less complicated programming interfaces with novice computer users in mind [13]. For example, Scratch has become a popular teaching tool due to its easy-to-understand block-based interface and abstracted programming concepts. Modern day XR authoring tools like Unity and Unreal Engine have adapted visual programming as alternative interfaces to writing code. Both Unreal's Blueprint system and Unity's visual scripting interface offer developers with a node-based scripting graphs that allow creation of game-play logic. These interfaces reduce the coding effort but require specialized understanding to be used effectively. Visual programming has been explored in VR to help users interact with complex functionality that would otherwise necessitate domain-specific knowledge [14]. The challenge with visual programming approaches is that their low-code paradigm does not always translate into low-complexity of use or understanding required to use effectively. Many of these visual programming languages designed to be low-code expose complex concepts that necessitate extra learning time, despite being low-code [15]. Thus, there is an open gap in the implementation of the low-code paradigm for XR development with the explicit intention to simultaneously reduce the knowledge and experience requirements.

C. Generative AI

Generative artificial intelligence (GenAI) has seen rapid development in recent years due to continuous innovation in computer hardware. GenAI provides an alternative approach to support the low-code paradigm by reducing the code-writing effort on the part of the user. The proficiency of

[1]Neos Metaverse: https://neos.com/ Last accessed: 01-Sep-2024

[2]Zoe: https://www.meta.com/en-gb/experiences/5464660053555314/ Last accessed: 01-Sep-2024

GenAI to produce correct code [16] has seen its incorporation into numerous programming support tools such as Github's Copilot and Microsoft's Intellicode. Large language models (LLMs) have revolutionised the fields of natural language processing and GenAI, the most notable of these models being OpenAI's GPT (Generative Pre-Trained Transformers). LLMs require a prompt as input, from which intent is predicted and content generated accordingly [17]. Users can provide a prompt to an LLM and acquire adequate results. However, prompt engineering can be employed to fine-tune a more desirable output [18]. Such a process often requires contextual information related to the broader task in order to prime the LLM towards generating relevant responses. Intent extraction, the process of eliciting intention from a given utterance, gives rise to virtual assistive approaches [19] that can enable newer forms of low-code paradigms.

GenAI has also found use in creation of visual artefacts and models. The generative process for images relies on diffusion models (e.g. DALL-E). For 3D models, advances in Neural radiance fields (NeRFs) have unlocked the ability to produce 3D models and scenes, complete with textures, from a single text prompt or input image. Gaussian splatting, has shown better result quality in less time than its predecessor (NeRFs). This is of particular interest when it comes to content authoring in XR where prompt-based generation or retrieval of models can alleviate the bottleneck on creativity that emerges from constraints on availability of models.

D. Integrating AI into XR Development

Given the continuous development of XR technologies, and the concurrent advancements of AI, the use of AI within XR applications is a reasonable next step. Hirzle et al. [20] reviewed 311 articles that explore the use of AI in the context of XR, showing active interest in integrating AI with XR. Within the diverse and distinct set of problems tackled, only one output targets generation of environments or scenes [21]. More recent research focuses on using AI to interpret the intent of the user and assist them in using an existing XR development tool. For e.g., BroomRocket explores object placement support within a scene in Blender using an open-source NLP model [22]. Similarly, DreamCodeVR demonstrates how speech driven prompts can allow no-code development through Unity [23] and LLMR showcases the capability of AI to dynamically generate XR scenes in real-time [6]. These developments show that XR development can be improved through the use of AI. However, we identify that there is still a gap in research regarding the interactions that occur between AI systems and the user within AI-enabled immersive development environments and the creativity support offered by such tools. Thus, ImoGenXR addresses this gap by expanding upon existing research and proposes a workflow that is built on top of the Unity game engine and uses GenAI to facilitate creativity support during immersive authoring tasks.

E. Conversational User Interfaces

Traditional application development has relied on code creation by using an integrated development environment (IDE) to write it. Even within an immersive experience, certain tasks require text entry. For immersive authoring, this manifests as situations where users need to create code within the immersive environment. Whilst a large body of work exploring text entry in VR exists [24], this modality is often inefficient and error prone [25], [26]. Conversational user interfaces (CUIs) offer an alternative method of input. The widespread availability of CUI-enabled devices like Google Home and Amazon Echo has made users more comfortable using them to access information or control smart devices. Weiß et al. found that a speech interface was easier to learn, simpler to use and allowed for greater efficiency [25] than 2D and 3D UIs. Hombeck et al. expand this finding by showing that voice UIs in VR increased user satisfaction and task efficiency [26]. We use this to inform our design decisions related to text entry within the virtual environment.

III. Design

A. Design Motivation

ImoGenXR's motivation is based in improving creativity support for authoring tasks within existing XR development tools. Proper creativity support can make it easier for beginners to focus on exploring their creative vision instead of struggling with the use of the tool.

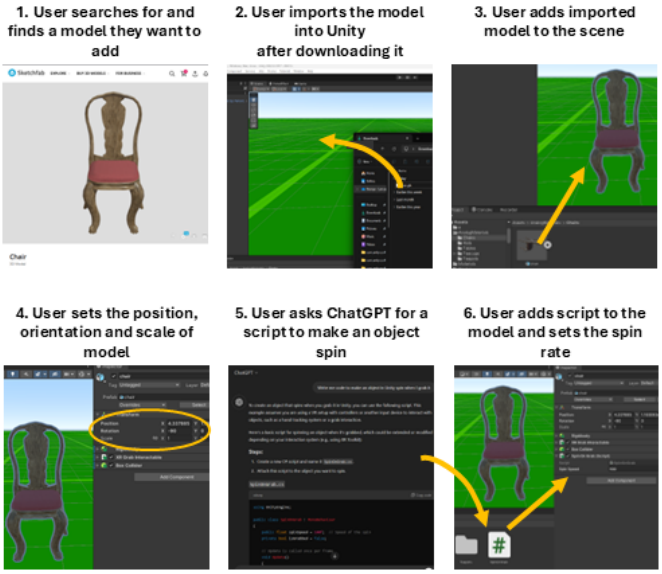

Fig. 2: Simplified Unity workflow for creating the scene described in the Design scenario

1) Design Scenario: To flesh out what could be important for creativity support, we use a design scenario [27]. Consider the case of an educator discussing the visuals from Alice in Wonderland, for example, the setup of the Mad Hatter's tea party similar to Figure 1. To engage the students and

to trigger their imagination, the educator wishes to create a simple VR experience that can be viewed using their school's Quest headsets. The scene is that of a tea party which includes objects such as a table, chairs, a teapot and teacups. Further, the educator would like to make the scene interactive by making the objects behave in interesting and unexpected ways. Assuming that this educator is not an expert in Unity development, this task presents a high entry barrier for success.

Experienced developers would be able to breakdown the task into smaller steps or actions they need to take to complete the task. This includes identifying 3D model resources, importing these into Unity, positioning them, deciding on what interaction-based animations are needed, creating or generating scripts for said animations and finally connecting these to the correct models in the scene. These tasks constitute steps in a workflow as shown in Figure 2. The educator from our design scenario above is more likely to struggle and finally abandon the idea. Even if they persist, the repeated context switches and the overall complexity of the tool will lead to higher cognitive load and hinder their creative process. This leads us to consider an alternative workflow set in an immersive environment which is discussed next.

2) Design of Workflow: Building upon the scenario of the educator creating a VR experience, we consider how the workflow design can make the task simpler for the user, through direct interaction with the scene and with minimal context switches out of it. The user could start with the placement of furniture; verbalising their thoughts directly into the system. For e.g., "I need furniture, let's add a table". The system then generates a set of tables and returns these to the educator for them to select from and place into the virtual environment. The user can repeat this to populate the scene with more furniture as desired. The user then considers making objects in the scene interactive; verbalising a trigger for the interaction ("Now I want the teapot to respond to touch") as well as the action to perform ("It would be nice if it floated away"). Building on the low-code paradigm, the user is presented with an abstract representation of the code which they can tack onto the teapot. They would want to test if the teapot reacts as expected and the system allows them to interact with options that serve as input parameters to the 'float' script. From the perspective of the user, they can focus on the creative aspect of the process while leaving the immersive authoring workflow to address the code and asset requirements.

B. Workflow Components

The above mentioned design scenario and the subsequent visualisation of the user-centric workflow allows us tobegin identifying the components that need to be combined to produce ImoGenXR.

1) Portal and CUI: We intend to build the immersive workflow on top of an existing XR development tool like Unity. This allows us to focus on the immersion and creativity support. It also removes the need to build a compilation pipeline that Unity has already implemented. A naive approach would first focus on presenting and logically arranging, within the immersive environment, the 2D UI menus that the development tool like Unity already provides. Instead we propose that the immersive workflow should follow a minimalist approach to inclusion of workflow UI elements within the immersive environment. This leads us to consider using the CUI whenever possible. However, in the simplest form of implementation CUIs inherently present a hidden affordance. Even when active, the CUI requires a signifier to provide feedback on the current state of the system.

We chose gesture input to activate the CUI and present a visual signifier to show the start of the CUI interaction. This allows a deliberate start of the interaction The signifier is shown as a 'portal' with the intention to maintain immersion while presenting an abstract interactive element that the user manipulates to obtain the desired objects and code. The portal (see Figure 3 top-1) is chosen over anthropomorphic characters as the portal signifies a rift in the existing environment and a connection to a source external to the user's virtual world. This supports the desired functionality of interactively accessing models and code without breaking the immersion of concurrently experiencing the environment being built. Simple commands can be issued through the CUI to allow the system to identify the user's intention. For example, "Get me a table" will return a collection of tables for the user to select from. The user should be given an opportunity to confirm that the command was processed correctly or, alternatively, correct the system's transcription to minimise errors and frustration that may result from the errors. Once the interaction is complete and the user is satisfied with the result, the virtual presence should be hidden from view to reduce visual clutter and misinterpretation of system state such as the user mistaking the system as waiting for additional input.

2) Generative AI: The CUI provides an interactive interface for requesting objects and actions. As shown in Figure 2, the user is responsible for switching context away from the development tool to obtain these objects and code that supports the actions. We avoid this context switch and resulting break in immersion by using GenAI to fill the gap. For model generation, we decided to utilise a large 3D object dataset such as ShapeNet [28] or Objaverse [29]; such databases contain enough samples for the purposes of the prototype application. Future iterations of the system could see the use of text-to-3D generation pipelines, once technically feasible. Interactivity of in-scene object requires scripts being attached to them. The workflow can satisfy the low-code paradigm by using GenAI to generate the code that is attached to the objects but only showing the user an abstract representation to interact with. While the user verbalises commands that describe what the 'object should do', we can utilize an LLM to extract the user intent and action. For example, the command "Get me a chair" would be broken into a "get object" intent, and a "chair" entity. The component that communicates with the LLM for code generation will also need prompt engineering to prime it for producing code in the relevant language and returning additional meta-data like input parameters in a usable format.

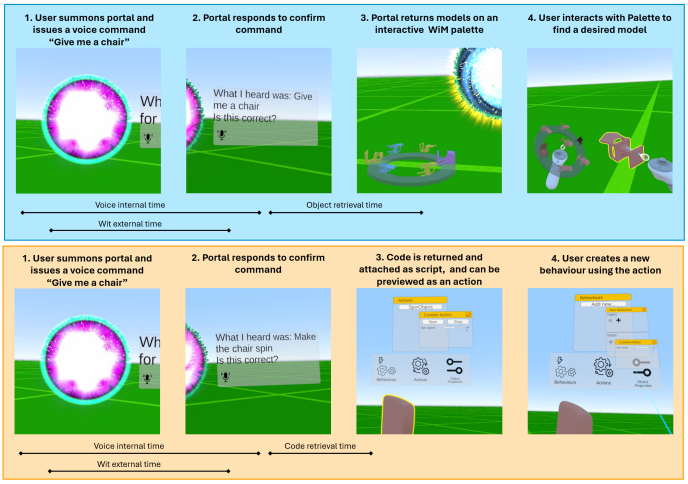

Fig. 3: Workflows of requesting objects (blue) and actions (orange) whilst using ImoGenXR. Time intervals show where the system is waiting on external systems to finish processing.

3) Object Palette: When the GenAI returns 3d models, the user should be able to interact with them meaningfully. An obvious approach is to display the objects through 2D-based menus and images. However, to maintain the immersion, we explored alternative approaches. We settled on the 'worlds in miniature' approach proposed by Stoakley et al. [30] which situates well within the 3D nature of the immersive environment. Combining this with a carousel concept that builds upon Belga et al. [31], we arrive at the design we refer to as the "object palette" (see Figure 3 top-3).

To keep the object palette's interactions familiar to users, we elected for a dunk technique in which the user will be able to grab their chosen object and place it into the virtual environment. The middle of the palette displays a larger preview of the object currently selected by the user. Users can also attach the object palette to their virtual hand to support additional manipulation through gestures. The user could tilt it right to fetch more object results or tilt it left to view previous set of objects. They could chuck it to disconnect or discard it once it is no longer needed.

4) Enabling Interactivity: Objects can be given interactivity by attaching custom code to them within the immersive environment. Typically, an experienced user would have knowledge of the required components to able to write a script to achieve this. As discussed in Section III-B2, the user can use a prompt such as, "Provide code for Unity to make an object spin when I grab it". The prompt naturally splits itself into both a trigger ("when I grab it") and the action to be executed ("spin"). Once the script is returned by the LLM, the user can then customise the input parameters of the script, as well as its trigger. The user accesses these through a series of nested interactive 2D menus consisting of behaviours, actions and triggers. While we seem to choose a naive approach here, we intentionally chose it to allow us to test if users showed an explicit preference towards the use of 2D menus. The low-code paradigm is further supported in the workflow by allowing the returned script to be tested immediately, without needing to switch between the immersive environment and the development environment.

C. Overall Workflow

A typical usage scenario would resemble the following: the user summons the portal through a gesture, a voice command for the desired object is made through the CUI, the system confirms the command, a selection of 3D models is made available to the user via the object palette, the user chooses and adds an object into the virtual environment and is able to immediately interact with it (see Figure 3 top-4). Additionally, the user can request an action to be added to the object through the CUI following a similar interaction workflow (see Figure 3 bottom). Actions can then be previewed, or added to a behaviour to make the object interactive. As objects are added into the virtual environment, the data needed to represent the object (e.g. mesh, position, rotation etc) will be stored alongside any actions and behaviours the user adds during runtime. Creating a fully self-contained environment in this way allows for the virtual environment to be saved and reloaded at a later date, enabling users to edit scenes they have previously created or share their scenes with others for entertainment or educational purposes.

IV. Implementation

Based on the design discussion of an immersive workflow, we have the functional requirements to implement the prototype system for ImoGenXR. Due to technical limitations, we were unable to implement dynamic object generation as we had originally designed due to the reasons discussed in Section III-B. We instead use the Objaverse 1.0 3D object dataset [29], which boasts a collection large enough for the purposes of the prototype implementation. The Objaverse 1.0 API has support for LVIS categories, allowing for keyword retrieval of objects annotated with the requested category (e.g. apple, cup) from the dataset. The architecture remains otherwise the same, but we provide an updated system architecture that can be divided into 5 primary components, these being: the conversational agent, the visual "portal" effect, object selection and placement via the "object palette", accessing object properties and behaviours through the "interaction menu" and the backend server.

The prototype was developed using Unity (ver. 2022.3.13f1) and XR Interaction Toolkit (ver. 3.0.5). Voice support uses the Meta Voice SDK with Wit.AI for voice recognition. Models and coded were generated accessed via Objaverse API and OpenAI's gpt-4o API.

A. Portal CUI

Users are able to activate (or "summon") the portal by performing a clockwise circular motion using one of their controllers. Similarly, a counter-clockwise circle will remove the palette from the user's view. Gesture recognition was

accomplished via a Unity-specific implementation[3] of the $1 gesture recogniser [32]. When a user presses the trigger to begin drawing a gesture, a raycast from the controller collides with a transparent plane directly in front of the user. These points of collision produce an array of 2D coordinates that are passed into the gesture recogniser.

After the user has performed the clockwise gesture, the portal will open and begin listening to the microphone. At this stage, a choice of two commands can be given: "Give me a [object]" or "Make the [object] [action]". Voice input is processed using Wit.Ai, which allows for pre-training of an NLP model that is able to extract pre-defined intents (whether a user's utterance was requesting an object or an action) and entities (the particular object or action that was requested). Users are then able to confirm that the system heard them correctly via a "yes" command that will use the returned intent and entities to either request an object from the Objaverse API, or code from the ChatGPT API. If the system misheard, a "no" command will revert to the start of the workflow, permitting the user to give make the request again. Once objects are imported into the runtime environment, the palette becomes accessible to the user, allowing them to choose and place objects into the immersive environment. Similarly, once the code for a requested action has finished compiling, it will appear in the object interaction menu.

B. Object and Code Retrieval

As mentioned previously, we retrieve objects from the Objaverse API and code for requested actions from OpenAI's ChatGPT API. For each "get object" command, we request a random selection of six objects from those that match the keyword in the user's request. The objects are received as glTF files, cached locally and subsequently imported into the runtime environment using Unity's glTFast package. The models are scaled, positioned and interactive child components are added. When the user requests an action script, we employ prompt engineering to coerce the GenAI into returning responses in a pre-defined format. The prompt requests a Unity script that performs the specified action and customized to operate on the interactive components of the target object. All variables are exposed as float-based properties with human-readable names, ensuring correct type conversions. The prompt also states that final response should be a formatted JSON. Once the response JSON is received, the code part is extracted and saved to a C# file inside the Unity project's Assets directory. Unity triggers a domain reload (an internal Unity process where all script files are reset and re-compiled), making the assets and code available for use without needing to restart the game. Serialization is used to maintain selected instance states across domain reloads.

C. Backend

The backend server was written in Python using the Flask framework, acts as middleware to handle requests made to the Objaverse and OpenAI APIs from the Unity application. This architecture also allows us to easily replace APIs in the future, such as substituting Objaverse for a cloud-based text-to-3D model provider (when one becomes feasibly available). We also use the server to store a small amount of state such as the last requested object tag and the object IDs that were returned, which supports the re-request and backwards functionality of the object palette.

[3]Unity $1 Gesture Recogniser: https://github.com/SteBeeGizmo/DollarUnity Last accessed: 01-Sep-2024

V. Experiment

A. Study Design

1) Research Question: The main aim of developing ImoGenXR is to see if a generative-AI supported immersive workflow has any impact on creativity support for existing XR development tools. We ran a user study which compares ImoGenXR with the baseline workflow from the existing system i.e. Unity (ver. 2022.3.13f1) to evaluate the perception of creativity support for both workflows.

2) Task: The task has to provide adequate opportunity for the users to express their creativity while retaining comparability across different users as well as across the two conditions. We used a task scenario similar to the design scenario described in Section III-A1. The participants were asked to recreate a scene resembling the Mad Hatter's Tea Party from Alice in Wonderland containing furniture and cutlery that would react to being interacted with. The creativity aspect was maintained by allowing the participants to decide the furniture layout, choose furniture style and configure the interactive behaviour of the objects as they saw fit. While using ImoGenXR, they retrieved the furniture and attached AI-generated scripts as behaviours using ImoGenXR's interface. For the Unity-only workflow (UoWF), we provided a selection of scripts and furniture assets grouped into categories (e.g. tables, chairs) that they could choose from. We did not allow participants to import additional assets into the project during the UoWF to ensure only the workflow within the Unity engine (and not the participant's ability to retrieve external content) was being evaluated. Similarly, content retrieved via ImoGenXR could not be modified with use of external tools. In both conditions, there was no time limit on the task and they were allowed to stop when they were happy with the generated scene. Additionally, both tasks utilised the same base sandbox environment including a grass-textured floor and skybox.

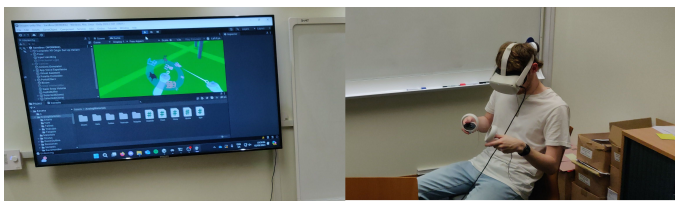

Fig. 4: Experimental setup, with participant using ImoGenXR on the right and the researcher's view on the left

3) Apparatus: We used Unity (ver. 2022.3.13f1) as the XR development tool for both ImoGenXR and UoWF. ChatGPT 4o was used to generate the code for ImoGenXR as well as pre-generating the scripts used by UoWF. The versions are reported for completeness and there is no version-specific feature of these software that underpins the functionality. The participants used a Windows 11 desktop (Intel i7-14700F, NVIDIA GeForce RTX 4060, 16GB RAM). A Meta Quest 2 configured in developer mode and connected via USB-C to the computer (through the Quest Link app) was used to view the task scene in both workflow conditions. The experimental setup used in the study is shown in Figure 4.

4) Participants: Participants were recruited from the staff and student cohort of Lancaster University via the snowball sampling. The participants were not offered any compensation for the study. In total, 20 participants participated in the study. We recorded generic participant demographics like age-range (18-20: two, 21-25: nine; 26-30: six; >30: three), gender (M:11; F:7; Other:2), English language proficiency (Native speakers: 10; CEFR C2: 8; CEFR C1: 2). We also recorded context specific participant demographics like experience with VR applications or games (Yes: 19; No: 1), propensity for motion sickness (Yes: 4; No: 16). We asked the participants to self-report their expertise with using game engines and to select the ones they had used. Ten participants reported themselves to be knowledgeable or experts while the rest mentioned they had passing or no knowledge. The participants selected the engines they had used. This showed a prevalent awareness of Unity (11) and Unreal (4) with some awareness of others (7).

5) Metrics: The study design is a mixed-methods design consisting of qualitative as well as quantitative metrics. To measure the creativity support, we used the Creativity Support Index (CSI) [33] questionnaire which provides a CSI score (out of 100) for each workflow within the context of the task. We also used the System Usability Score (SUS) questionnaire to evaluate the overall usability of both workflows. For the post-study questionnaire, we used Plutchik's emotion wheel [34] to capture affect emerging from use of each workflow and finally included open-ended feedback questions. Qualtrics XM was used to administer the questionnaires and log all the participant responses. We also used screen recording to capture the user interactions with both workflows. For ImoGenXR we also used additional logging to capture time intervals between different events within the system.

6) Procedure: The study was set up in quiet room with only the researcher and participant present. Participants filled out a consent form followed by a pre-study questionnaire. Then the participants received a tutorial on how to use the first workflow. The choice of the first workflow as either ImoGenXR or UoWF was counterbalanced to alleviate learning-order effects. The participants then completed the task using the first workflow. After this, they filled up the questionnaire for the first tool. They were then given a tutorial on the second workflow and again asked to complete the same task with the second tool. Once they completed the task, they filled up the questionnaire for the second tool. Finally, they filled up a post-study questionnaire. For each workflow, we allowed the participants to decide when they were happy enough with the scene to stop and we made it clear that there was no time-pressure to complete the task quickly. The study was run after getting ethics approval from the University Research Ethics Committee.

The UoWF tutorial consisted of an overview of adding objects into the scene, basic camera controls and UI navigation, components required to make objects interactive and variable customisation within added scripts. The participants were also shown how to deploy the scene and view it in the HMD. Similarly, for ImoGenXR, users were shown the basic locomotion controls and introduced to the key mechanics of the system such as accessing the CUI, retrieving objects and actions, use of the object palette, adding behaviours to objects and customising object properties. Participants received no other training besides these tutorials.

B. Results

1) Processing: Qualtrics was setup to pre-compute the CSI and SUS scores for each participant. We analysed these scores using IBM SPSS 29. System logs were processed using Python to produce aggregated results.

2) Comparative: CSI and SUS Scores: We used repeated measures ANOVA on both SUS and CSI scores, and found a significant difference between the SUS scores for the UoWF (μ*=46.00*) and ImoGenXR (μ*=65.50*), $F_{(1,19)} = 9.56$, $p<.01$. A significant difference was also found between the CSI scores for the UoWF (μ*=42.27*) and ImoGenXR (μ*=65.67*), $F_{(1,19)} = 10.28$, $p<.005$. We also analysed individual CSI factors (Effort, Exploration, Immersion, Expressiveness and Enjoyment) as per the protocol [33] and ran the tests on the weighted scores. The results of these tests are presented in Table I. We see significant difference for Effort, Immersion and Enjoyment with ImoGenXR scoring higher than UoWF. Users rated Effort as the most important factor. Since the task and the workflows don't have a collaborative component, it scores zero but is still reported as per the reporting protocol.

3) Comparative: Workflow Usage Logs: We processed the system logs and screen-recordings to extract the task completion time (TCT), counts of models explored, models inserted and finally actions attached to objects. The TCT for ImoGenXR (μ*=16m 22s*) is similar to that of UoWF (μ*=19m 06s*). Users explored nearly similar number of objects in ImoGenXR (μ*=5.10*) versus UoWF (μ*=4.75*). Scenes created in ImoGenXR (μ*=13.15*) had some more objects than UoWF (μ*=11.55*). The number of actions added for UoWF (μ*=4.75*) is also similar to ImoGenXR (μ*=1.55*). None of these results could be considered statistically significant differences with $p>.05$ in all cases. The implication of these results in discussed in Section VI-A.

4) Comparative: Affect and User Responses: Plutchik's Emotion Wheel [34] is widely used for sentiment analysis but lacks a standard method for comparing affect between systems. We developed a strategy based on prior work [35],

TABLE I: Statistical analysis of workflow scores with mean values and standard deviation (s.d.)

Metric	**Workflow ⇒**	**UoWF μ (s.d.)**		**ImoGenXR μ (s.d.)**		**F-val**	**Sig.**
SUS		46.00 (23.98)		65.50 (18.68)		9.56	<.01
CSI		42.27 (25.73)		65.67 (23.09)		10.28	<.005
CSI Factors	**Counts μ (s.d.)**	**Score μ (s.d.)**	**Wt. Score μ (s.d.)**	**Score μ (s.d.)**	**Wt. Score μ (s.d.)**	**Wt. Score Tests**	
Reward for Effort	3.80 (1.24)	8.05 (5.34)	29.00 (21.03)	12.85 (5.44)	48.15 (25.65)	8.82	<.01
Exploration	3.20 (.95)	9.00 (5.47)	29.35 (22.85)	12.35 (4.30)	39.25 (17.69)	3.24	N.S.
Collaboration*	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	N/A	N/A
Immersion	2.30 (1.49)	7.15 (4.43)	16.7 (18.99)	13.10 (4.98)	34.15 (28.25)	11.57	<.005
Expressiveness	2.50 (1.47)	9.05 (6.10)	21.45 (20.70)	13.20 (4.87)	31.45 (22.45)	3.03	N.S.
Enjoyment	2.95 (1.15)	9.80 (5.61)	30.30 (23.89)	13.95 (4.93)	43.10 (26.61)	7.89	<.05

* Participants marked collaboration questions as "Not Applicable" and ranked it lowest as a factor because no collaboration was required.

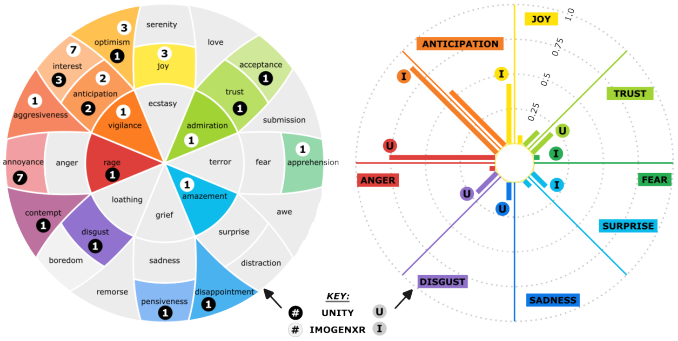

Fig. 5: Analysis of affect: (Left) Response counts for each emotion. Emotions that were not selected are greyed out. (Right) Emotion vector visualization across the 8 dimensions. Labels (U/I) denote which workflow has higher values for the associated dimension

where participants select a single emotion per workflow in a questionnaire (Figure 5-left). Each selection is converted into scores for the eight basic emotions from the wheel's middle ring. The strongest emotions (innermost ring) score three, while the weakest (outermost ring) score one. Unselected emotions score zero. Dyad emotions split their score (1 each) between adjacent basic emotions (e.g., 'Optimism' gives 'Anticipation' and 'Joy' a score of one each). We then compute an 8-dimensional PEW vector (Joy, Anticipation, Surprise, Trust, Anger, Disgust, Sadness, Fear) by summing weighted emotion scores across participants and normalizing. This unit vector represents affect per workflow. For UoWF, the PEW vector is (0.07, 0.55, 0.07, 0.21, 0.76, 0.21, 0.14, 0.00), while for ImoGenXR it is (0.44, 0.87, 0.15, 0.15, 0.05, 0.00, 0.00, 0.05). The histogram and vector comparison of participant responses is shown in Figure 5-left. ImoGenXR scores higher on three of four positive emotions (Joy, Anticipation, Surprise, Trust), while UoWF scores significantly higher (i.e., worse) on three of four negative emotions (Anger, Disgust, Sadness, Fear).

5) ImoGenXR-specific Results: We asked participants to choose one feature that they felt was most exciting and also the one they felt was least important. The participants chose the Portal (8), CUI (6) and Palette (6) as the most important. No one chose the menus, which incidentally were the most selected option (9) as the least important feature. We also analysed the system logs from ImoGenXR to see how the external features affected the total playtime. As seen in Figure 3, the voice command is processed externally, but the voice external (Wit) processing (μ=*3.12s*) happens in parallel with the voice internal time (μ=*6.29s*) spent listening. Thus its impact on overall playtime is low. However the external time for object retrieval (μ=*3.64s*) and the code retrieval (μ=*2.60s*) do affect the overall playtime. The voice transcription error rates (where the user confirmed that the system had heard them wrong) were (μ=*2.55*) for objects and (μ=*1.10*) for actions per participant session.

VI. Discussion

A. Study Results

1) Questionnaires: With a significant difference in CSI scores (μ=65.67 versus μ=42.27 for ImoGenXR and UoWF respectively), we can see that participant found that ImoGenXR provided better creativity support than UoWF. The SUS scores also show similar results. Thus, we can say that ImoGenXR was successful in achieving its objective of providing better creativity support for the tool itself. Individual weighted factor comparison showed that participants found ImoGenXR to be more enjoyable, immersive and more rewarding for the amount of effort expended. The CSI score for ImoGenXR is still not very high and shows that there is further scope for improvement.

2) Affect: The PEW vector puts ImoGenXR higher than UoWF for three of the four positive emotions. UoWF scores higher for three of the four negative emotions too. Thus, for the affect part of creativity support, ImoGenXR scores better than UoWF on six out of the eight dimensions of the PEW vector. Novice users and even more experienced users all seem to have a negative affect towards Unity. This relates to the impression that the tool is difficult to learn and sometimes difficult to use. ImoGenXR, on the other hand, was able to elicit mainly positive affect. This is encouraging for research focusing on immersive authoring.

3) Workflow Usage Logs: We did not find significant differences for the system usage logs between ImoGenXR and UoWF. We intentionally avoided context switches out

of UoWF. The users did not have to search and retrieve a model or ask ChatGPT to generate code. These were provided, conveniently organized, in the assets area of Unity. We undercount the UoWF task completion time due to this; were these not provided, additional time would be added onto the task completion time that is already accounted for in our analysis of ImoGenXR.

4) Participant #13: This participant presented as an outlier, giving ImoGenXR a CSI score of 22.0 while rating UoWF as 76.3. We looked at their comments and workflow videos to understand what resulted in this score. The individual reported that their expertise with Unity as knowledgeable. They liked the placement controls but found Unity's alternatives better due to the lack of fine-grained placement control. This observation feeds into our expectation that ImoGenXR is used as a workflow rather than a replacement for the interface Unity offers. As users of ImoGenXR gain experience and establish their expertise, they are likely to customize their use of and switching between the workflows to what suits their needs best. This also shows that creativity support is more nuanced and needs to be tailored to suit the expertise of the user.

B. Workflow not Tool

Our study compared ImoGenXR against the standard Unity-based workflow showing negative trends for creativity support and a generally negative affect towards Unity. However, we stress that we do not consider this as a negative assessment of Unity. ImoGenXR is built on top of Unity. Unity provides a powerful set of features that can be exploited well by an expert user and ImoGenXR benefits from these features. For developers of XR development tools, our study highlights how tools like Unity can benefit from providing immersive workflows like ImoGenXR as a part of their offering.

VII. Applications

We believe that the strengths of ImoGenXR fully emerge in progressing the democratization of VR content authoring for novice users, explained through three scenarios.

Developing immersive story-telling experiences
Interactive story-telling is a creative activity that parents and children can use to support various learning goals [36], [37]. ImoGenXR can be used by a parent to recreate a scene based on a story, and to reinforce a particular learning objective or moral in a more engaging way. An expanded system could also process more complex prompts, directly taken from the story to retrieve or visualize a scene further as the story progresses.

Simple experiences for educators
ImoGenXR can allow educators to build simple interactive experiences that engage the students' attention and allow the educators to develop pedagogically sound experiences. ImoGenXR would also alleviate the challenges of learning how to use Unity, and allow the educators to scaffold their expertise in creating VR experiences in Unity.

Rapid prototyping of experiences
Many novice users have a desire to create a VR experience, but lack the knowledge to fully execute the design and development. ImoGenXR enables users to be able to rapidly explore multiple alternatives, without having to write complex code. In effect, the system could also allow XR developers to discuss and explore multiple scenarios with clients by treating the system as a 3D wireframing tool.

VIII. Limitations and Future Work

ImoGenXR has some limitations, which we discuss next. In our current implementation, asset generation is limited to the Objaverse library. However, we believe that future advances—such as NeRFs or Gaussian Splatting—will enable the generation of more complex assets, including animated or rigged models, thus broadening the scope of what systems like ImoGenXR can achieve. For the code scripts, we do not perform any error checking or verification that the output is correct. If script compilation fails, we request another one immediately. More complex actions, and scripts, would likely require more thorough error handling strategies such as those utilised in DreamCodeVR [23] and LLMR [6]. Additionally, the use of GenAI has well-documented sustainability and ethical concerns, which we do not discuss in detail here but should remain a focus of future research.

While the above limitations emerge due to integration with external systems, we also identified an issue with how Unity handles hot reloading; once a script is imported, Unity recompiles the entire project, resulting in the HMD view becoming temporarily static. This can be jarring, and a workaround would enhance the experience. Most participants ranked the interaction menu as the least important feature, highlighting the potential for further research into comparisons between alternative methods of displaying information and interaction designs that more easily support users in customising object interactivity. Multiple participants noted that the interaction with objects created from the palette was difficult at times. Future work on ImoGenXR will look into selective use of gravity, toggling rigid-body collisions and making fine control of placement easier when working with the objects.

IX. Conclusion

In this paper we propose ImoGenXR, a novel immersive authoring workflow that allows novice users to create interactive XR scenes through dynamic loading of objects and code produced by generative AI. We also conduct a user study that compares creativity support offered by ImoGenXR and a popular alternative authoring tool Unity. The results show that users greatly prefer the creativity support offered by ImoGenXR. The work contributes to and motivates future research to enhance creative support in novice content authoring, while finding immediate use in a variety of creative scenarios such as storytelling and creation of XR-based educational material.

References

[1] R. Sims and A. Karnik, "Opportunities and challenges for vr-mediated educational resources: An educator's perspective," in *EDULEARN22 Proceedings*, ser. 14th International Conference on Education and New Learning Technologies. IATED, 4-6 July, 2022 2022, pp. 8810–8819. [Online]. Available: https://doi.org/10.21125/edulearn.2022.2109

[2] M. Nebeling and M. Speicher, "The trouble with augmented reality/virtual reality authoring tools," in *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, Oct 2018, pp. 333–337.

[3] M. Nebeling, K. Lewis, Y.-C. Chang, L. Zhu, M. Chung, P. Wang, and J. Nebeling, "Xrdirector: A role-based collaborative immersive authoring system," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20. ACM, 2020, p. 1–12. [Online]. Available: https://doi.org/10.1145/3313831.3376637

[4] V. Frau, L. D. Spano, V. Artizzu, and M. Nebeling, "Xrspotlight: Example-based programming of xr interactions using a rule-based approach," *Proc. ACM Hum.-Comput. Interact.*, vol. 7, no. EICS, jun 2023. [Online]. Available: https://doi.org/10.1145/3593237

[5] B. Shneiderman, "Creativity support tools," *Commun. ACM*, vol. 45, no. 10, p. 116–120, oct 2002. [Online]. Available: https://doi.org/10.1145/570907.570945

[6] F. De La Torre, C. M. Fang, H. Huang, A. Banburski-Fahey, J. Amores Fernandez, and J. Lanier, "Llmr: Real-time prompting of interactive worlds using large language models," in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, ser. CHI '24. ACM, 2024. [Online]. Available: https://doi.org/10.1145/3613904.3642579

[7] H. Coelho, P. Monteiro, G. Gonçalves, M. Melo, and M. Bessa, "Authoring tools for virtual reality experiences: a systematic review," *Multimedia Tools and Applications*, vol. 81, no. 19, pp. 28 037–28 060, 3 2022. [Online]. Available: http://dx.doi.org/10.1007/s11042-022-12829-9

[8] A. Steed and M. Slater, "A dataflow representation for defining behaviours within virtual environments," in *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, 1996, pp. 163–167.

[9] G. Lee, C. Nelles, M. Billinghurst, and G. Kim, "Immersive authoring of tangible augmented reality applications," in *International Symposium on Mixed and Augmented Reality (ISMAR'04)*, 2004, pp. 172–181.

[10] L. Zhang and S. Oney, "Flowmatic: An immersive authoring tool for creating interactive scenes in virtual reality," *ACM Symposium on User Interface Software and Technology*, 2020.

[11] V. Krauß, M. Nebeling, F. Jasche, and A. Boden, "Elements of xr prototyping: Characterizing the role and use of prototypes in augmented and virtual reality design," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22. ACM, 2022. [Online]. Available: https://doi.org/10.1145/3491102.3517714

[12] N. Ashtari, A. Bunt, J. McGrenere, M. Nebeling, and P. K. Chilana, "Creating augmented and virtual reality applications: Current practices, challenges, and opportunities," *International Conference on Human Factors in Computing Systems*, 2020.

[13] T. D. Kimura, J. W. Choi, and J. M. Mack, *A visual language for keyboardless programming*. Washington University, Department of Computer Science, 1986.

[14] J. Vincur, M. Konopka, J. Tvarozek, M. Hoang, and P. Navrat, "Cubely," VRST '17: 23rd ACM Symposium on Virtual Reality Software and Technology. ACM, 11 2017. [Online]. Available: http://dx.doi.org/10.1145/3139131.3141785

[15] Y. Luo, P. Liang, C. Wang, M. Shahin, and J. Zhan, "Characteristics and challenges of low-code development: The practitioners' perspective," in *Proceedings of International Symposium on Empirical Software Engineering and Measurement (ESEM '21)*. ACM, 2021. [Online]. Available: https://doi.org/10.1145/3475716.3475782

[16] C. E. A. Coello, M. N. Alimam, and R. Kouatly, "Effectiveness of chatgpt in coding: A comparative analysis of popular large language models," *Digital*, vol. 4, no. 1, pp. 114–125, 2024. [Online]. Available: https://www.mdpi.com/2673-6470/4/1/5

[17] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt," 2023. [Online]. Available: https://arxiv.org/abs/2303.04226

[18] B. Chen, Z. Zhang, N. Langrené, and S. Zhu, "Unleashing the potential of prompt engineering in large language models: a comprehensive review," 2024. [Online]. Available: https://arxiv.org/abs/2310.14735

[19] H. Weld, X. Huang, S. Long, J. Poon, and S. C. Han, "A survey of joint intent detection and slot filling models in natural language understanding," *ACM Comput. Surv.*, vol. 55, no. 8, dec 2022. [Online]. Available: https://doi.org/10.1145/3547138

[20] T. Hirzle, F. Müller, F. Draxler, M. Schmitz, P. Knierim, and K. Hornbæk, "When xr and ai meet - a scoping review on extended reality and artificial intelligence," CHI '23: CHI Conference on Human Factors in Computing Systems. ACM, 4 2023, pp. 1–45. [Online]. Available: http://dx.doi.org/10.1145/3544548.3581072

[21] M. Sra, P. Maes, P. Vijayaraghavan, and D. Roy, "Auris," Symposium on Virtual Reality Software and Technology (VRST'17). ACM, 11 2017. [Online]. Available: http://dx.doi.org/10.1145/3139131.3139139

[22] S. Bonic, J. Bonic, and S. Schmid, "Broomrocket: Open source text-to-3d algorithm for 3d object placement," *ACM Games*, vol. 2, no. 3, aug 2024. [Online]. Available: https://doi.org/10.1145/3648233

[23] D. Giunchi, N. Numan, E. Gatti, and A. Steed, "Dreamcodevr: Towards democratizing behavior design in virtual reality with speech-driven programming," 2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR). IEEE, 3 2024. [Online]. Available: http://dx.doi.org/10.1109/VR58804.2024.00078

[24] T. J. Dube and A. S. Arif, "Text entry in virtual reality: A comprehensive review of the literature," in *Human-Computer Interaction. Recognition and Interaction Technologies*, M. Kurosu, Ed. Cham: Springer International Publishing, 2019, pp. 419–437.

[25] Y. WeiB, D. Hepperle, A. SieB, and M. Wolfel, "What user interface to use for virtual reality? 2d, 3d or speech–a user study," 2018 International Conference on Cyberworlds (CW). IEEE, 10 2018. [Online]. Available: http://dx.doi.org/10.1109/CW.2018.00021

[26] J. Hombeck, H. Voigt, and K. Lawonn, "Voice user interfaces for effortless navigation in medical virtual reality environments," *Computers & Graphics*, p. 104069, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849324002048

[27] J. M. Carroll, *Scenario-Based Design*, 2003, pp. 45–70.

[28] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[29] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A universe of annotated 3d objects," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[30] R. Stoakley, M. J. Conway, and R. Pausch, "Virtual reality on a wim: interactive worlds in miniature," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1995, pp. 265–272.

[31] J. Belga, T. D. Do, R. Ghamandi, R. P. McMahan, and J. J. LaViola, "Carousel: Improving the accuracy of virtual reality assessments for inspection training tasks," in *Proceedings of the 28th ACM Symposium on Virtual Reality Software and Technology*, ser. VRST '22. ACM, 2022. [Online]. Available: https://doi.org/10.1145/3562939.3565618

[32] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," in *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '07. ACM, 2007, p. 159–168. [Online]. Available: https://doi.org/10.1145/1294211.1294238

[33] E. Cherry and C. Latulipe, "Quantifying the creativity support of digital tools through the creativity support index," *ACM Trans. Comput.-Hum. Interact.*, vol. 21, no. 4, jun 2014. [Online]. Available: https://doi.org/10.1145/2617588

[34] R. Plutchik, "A psychoevolutionary theory of emotions," *Social Science Information*, vol. 21, no. 4-5, pp. 529–553, 1982. [Online]. Available: https://doi.org/10.1177/053901882021004003

[35] S. Wang, A. Maoliniyazi, X. Wu, and X. Meng, "Emo2vec: Learning emotional embeddings via multi-emotion category," *ACM Trans. Internet Technol.*, vol. 20, no. 2, apr 2020. [Online]. Available: https://doi.org/10.1145/3372152

[36] Z. Zhang, Y. Xu, Y. Wang, B. Yao, D. Ritchie, T. Wu, M. Yu, D. Wang, and T. J.-J. Li, "Storybuddy: A human-ai collaborative chatbot for parent-child interactive storytelling with flexible parental involvement," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22. ACM, 2022. [Online]. Available: https://doi.org/10.1145/3491102.3517479

[37] H. Dang, F. Brudy, G. Fitzmaurice, and F. Anderson, "Worldsmith: Iterative and expressive prompting for world building with a generative ai," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '23. ACM, 2023. [Online]. Available: https://doi.org/10.1145/3586183.3606772